

KNOWLEDGE REPRESENTATION AND REASONING

KNOWLEDGE REPRESENTATION AND REASONING

COGNITIVE ROBOTICS

A Logical Account of Causal and Topological Maps

Emilio Remolina and Benjamin Kuipers*

Computer Science Department
The University of Texas at Austin
Austin, TX 78712, USA

e-mail: {eremolin,kuipers}@cs.utexas.edu

Abstract

We consider the problem of how an agent creates a discrete spatial representation from its continuous interactions with the environment. Such representation will be the *minimal* one that explains the experiences of the agent in the environment. In this paper we take the Spatial Semantic Hierarchy as the agent's target spatial representation, and use a circumscriptive theory to specify the minimal models associated with this representation. We provide a logic program to calculate the models of the proposed theory. We also illustrate how the different levels of the representation assume different spatial properties about both the environment and the actions performed by the agent. These spatial properties play the role of "filters" the agent applies in order to distinguish the different environment states it has visited.

1 Introduction

The problem of map building –how an agent creates a discrete spatial representation from its continuous interactions with the environment– can be stated formally as an abduction task where the actions and observations of the agent are explained by connectivity relations among places in the environment [Shanahan, 1996, Shanahan, 1997, Remolina and Kuipers, 1998]. In this paper we consider the Spatial Semantic Hierarchy (SSH) [Kuipers, 2000, Kuipers and Byun, 1988, Kuipers and Byun, 1991] as the agent's target spatial representation. The SSH is a set of distinct representations for large scale space, each with its own ontology and each abstracted from the levels below it. The SSH describes the different states of knowledge that an agent uses in order to organize its sensorimotor experiences and create a spatial representation (i.e. a map). Using the SSH representation, navigation among places is not dependent on the accuracy, or even the existence, of metrical knowledge of the environment.

*This work has taken place in the Intelligent Robotics Lab at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Intelligent Robotics lab is supported in part by NSF grants IRI-9504138 and CDA 9617327, and by funding from Tivoli Corporation.

In order to define the *preferred models* associated with the experiences of the agent, we use a circumscriptive theory to specify the SSH's (minimal) models. Different models can exist that explain the same set of experiences. This occurs because the agent could associate the same sensory description to different environment states, or because the agent has not completely explored the environment. The different SSH levels assume different spatial properties about the environment and the actions performed by the agent. These spatial properties play the role of "filters" the agent applies in order to distinguish the different environment states it has visited. For instance, at the SSH causal level two environment states are considered the same if any sequence of actions started at these states renders the same sequence of observations. At the SSH topological level, two environment states are considered the same if they are at the same place along the same paths. Finally, at the SSH metrical level, two environment states are the same, if it is possible to assign to them the same coordinates in any frame of reference available to the agent. In sections 3 and 4 we make precise the claims above.

Finally, we use the SSH circumscriptive theory as the specification for a logic program used to implement the abduction task. In the paper we provide the logic program for the SSH causal level theory and illustrate how to encode the minimality condition associated with this theory. We have implemented the program using Smodels [Niemelä and Simons, 1997] and confirm that the theory yields the intended models.

2 Related Work

The SSH grew out of the TOUR model proposed in [Kuipers, 1977, Kuipers, 1978]. Other computational theories of the cognitive map have been proposed: [Kortenkamp *et al.*, 1995, McDermott and Davis, 1984, Leiser and Zilbershatz, 1989, Yeap, 1988]. These theories share the same basic principles: the use of multiple frames of reference, qualitative representation of metrical information, and connectivity relations among landmarks. They differ in how they define what a landmark is, or the description (view, local 2D geometry) associated with a landmark. Except for McDermott and Davis, none of the theories above has a formal account like the one presented in this paper for the SSH.

Considering map building as a formal abduction task has been proposed by Shanahan [Shanahan, 1996, Shanahan, 1997]. He proposes a logic-based framework (based on the

circumscriptive event calculus) in which a robot constructs a model of the world through an abductive process whereby sensor data is explained by hypothesizing the existence, locations, and shapes of objects. In Shanahan’s work, space is considered a real-valued coordinate system. As pointed out in [Shanahan, 1997], a problem of Shanahan’s approach is the existence of many minimal models (maps) that explain the agent’s experiences. We have alleviated this problem by considering the SSH topological map instead of an Euclidean space as the agent’s target spatial representation.

The problem of distinguishing environment states by outputs (views) and inputs (actions) has been studied in the framework of automata theory [Basye *et al.*, 1995]. In this framework, the problem we address here is the one of finding the smallest automaton (w.r.t. the number of states) consistent with a given set of input/output pairs. Without any particular assumptions about the environment or the agent’s perceptual abilities, the problem of finding this smallest automaton is NP-complete [Basye *et al.*, 1995].

The SSH [Kuipers, 2000, Kuipers and Byun, 1988, Kuipers and Byun, 1991] abstracts the structure of an agent’s spatial knowledge in a way that is relatively independent of its sensorimotor apparatus and the environment within which it moves. At the *SSH control level*, the agent and its environment are modeled as continuous dynamical systems whose equilibrium points are abstracted to a discrete set of *distinctive states*. A distinctive state has associated a *view* describing the sensory input obtained at that distinctive state. The control laws, whose executions define trajectories linking these distinctive states, are abstracted to *actions*, giving a discrete causal graph representation for the state space. The causal graph of states and actions can in turn be abstracted to a topological network of *places*, *paths* and *regions* (i.e. the *topological map*). Local metrical models, such as occupancy grids, of neighborhoods of places and paths can then be built on the framework of the topological network while avoiding global metrical consistency problems. In the next sections we formally describe the SSH causal and topological levels.

3 SSH Causal level

We use a first order sorted language in order to describe the SSH causal level. The sorts of this language include *distinctive states*, *views*, *actions* and *schemas*. The sort of distinctive states corresponds to the names given by the agent to the fix-points of hill-climbing control strategies. It is possible for the agent to associate different distinctive state names with the same environment state. This is the case since the agent might not know at which of several environment states it is currently located. A distinctive state has an associated view. We use the predicate $View(ds, v)$ to represent the fact that v is a *view* associated with *distinctive state* ds . We assume that a distinctive state has a unique view. However, we do **not** assume that views uniquely determine distinctive states (i.e. $View(ds, v) \wedge View(ds', v) \not\rightarrow ds = ds'$). This is the case since the sensory capabilities of an agent may not be sufficient to distinguish distinctive states.

An action has a unique type, either *travel* or *turn*, associated with it. We use the predicate $Action_type(a, type)$

to represent the fact that the type of action a is *type*. Turn actions have associated a unique turn description, either *turnLeft*, *turnRight* or *turnAround*. We use the predicate $Turn_desc(a, desc)$ to indicate that $desc$ is the turn description associated with the turn action a .

A schema represents an action execution performed by the agent in the environment. An action execution is characterized in terms of the distinctive states the agent was at before and after the action was performed.¹ We use the predicate $CS(s, ds, a, ds')$ to denote the fact that according to schema s , action a was performed starting at distinctive state ds and ending at distinctive state ds' . While schemas are explicit objects of our theory, most of the time it is convenient to leave them implicit. We introduce the following convenient notation:

$$\begin{aligned} \langle ds, a, ds' \rangle &\equiv_{def} \exists s CS(s, ds, a, ds') \\ \langle ds, type, ds' \rangle &\equiv_{def} \exists a \{ \langle ds, a, ds' \rangle \wedge Action_type(a, type) \} \\ \langle ds, desc, ds' \rangle &\equiv_{def} \exists a \{ \langle ds, a, ds' \rangle \wedge Turn_desc(a, desc) \} \end{aligned}$$

Example 1

Consider a robot moving in the environment depicted in figure 1. The robot moves from distinctive state a to distinctive state b by performing a follow-midline action, ml . Then the robot performs the same action to move to distinctive state c . We assume that all corridor intersections look alike ($v+$). This set of experiences can be described by the formulae:

$$Action_type(ml, travel), CS(s1, a, ml, b), CS(s2, b, ml, c), View(a, v+), View(b, v+), View(c, v+).$$

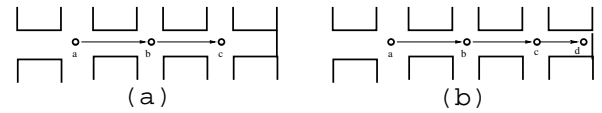


Figure 1: (a) Distinctive states a , b and c are not distinguishable at the causal level. Topological information is needed in order to distinguish them. (b) All distinctive states are distinguished at the causal level given the new information $\langle c, travel, d \rangle$.

Given this set of experiences, at the SSH causal level distinctive states a , b and c are not distinguishable. Any known sequence of actions renders the same set of views. However, at the SSH topological level all these distinctive states are distinguishable since the robot has traveled from a to b and then to c following the same *path* (see example 3). Should the robot continue the exploration and visit distinctive state d , with view \sqsupset , then by relying just on known actions and views the agent can distinguish all distinctive states it has visited (see example 2). *{end of example}*

The agent’s experiences in the environment are described in terms of CS , $View$, $Action_type$ and $Turn_desc$ atomic formulae. Hereafter we use \mathbf{E} to denote a particular agent’s experience formulae. By $\mathbf{HS}(\mathbf{E})$ we denote the formulae

¹An action execution also has metrical information associated with it. This metrical information represents an estimate of, for example, the distance or the angle between the distinctive states associated with the action execution.

stating that the sorts of schemas, distinctive states, views and actions are completely defined by the sets of *schema*, *distinctive states*, *view* and *action* constant symbols occurring in E respectively.² By **DT** we denote our domain theory, the formulae stating that: (-) the sets {turn, travel}, {turnLeft, turnRight, turnAround}, completely define the sorts of *action_types* and *turn_descriptions*; (-) an action has associated a unique action type ; (-) distinctive states have associated a unique view; (-) the description associated with an action is unique; (-) turn actions have associated a turn description; (-) the type of actions as well as the qualitative description of turn actions is the one specified in E . The SSH causal theory **CT(E)** defines when two distinctive states are indistinguishable at the SSH causal level. We use the predicate $ceq(ds, ds')$ to denote this fact. We will assume that actions are *deterministic*:³

$$\langle ds, a, ds' \rangle \wedge \langle ds, a, ds'' \rangle \rightarrow ds' = ds'' . \quad (1)$$

CT(E) is the following nested abnormality theory [Lifschitz, 1995]:

$$CT(E) = E, HS(E), DT, Axiom1, CEQ_block$$

where **CEQ_block** is defined as

$$\begin{aligned} &\{ \text{max } ceq : \\ &ceq(ds, ds') \rightarrow ceq(ds', ds), \\ &ceq(ds, ds') \wedge ceq(ds', ds'') \rightarrow ceq(ds, ds''), \\ &ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v), \quad (2) \\ &ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds'_1 \rangle \wedge \langle ds_2, a, ds'_2 \rangle \rightarrow ceq(ds'_1, ds'_2) \quad (3) \\ &\} \end{aligned}$$

It can be proved that the predicate ceq defines an equivalence relation on the sort of distinctive states. Axiom 2 states that indistinguishable distinctive states have the same view. Axiom 3 states that if distinctive states ds and ds' are indistinguishable and action a has been performed for both ds and ds' , then the action links these states with indistinguishable states. By maximizing ceq we identify distinctive states that cannot be distinguished by actions and/or views, and thereby minimize the set of states represented by the model.

Axioms 2 and 3 allow us to prove the following useful lemma:

Lemma 1 *Let A denote a sequence of action symbols. Let $A(ds)$ denote the distinctive state symbol resulting of starting the sequence A at distinctive state ds or \perp if A is not defined for ds .⁴ Then,*

$$\begin{aligned} &ceq(ds, ds') \wedge A(ds) \neq \perp \wedge A(ds') \neq \perp \\ &\rightarrow View(A(ds), v) \equiv View(A(ds'), v) . \end{aligned}$$

Example 2

²That *sort* is completely defined by the constant symbols s_1, \dots, s_n means that an interpretation for *sort* is the *Herbrand* interpretation defined by the set $\{s_1, \dots, s_n\}$.

³Throughout this paper we assume that formulas are universally quantified.

⁴Given an action symbol A and distinctive state ds , $A(ds) = ds'$ if the schema $\langle ds, A, ds' \rangle$ has been observed, otherwise, $A(ds) = \perp$. Moreover, $A(\perp) = \perp$. The definition is then extended to action sequences in the standard way. Notice that $A(ds)$ being well-defined relies on our assumption that actions are deterministic (Axiom 1).

Consider the situation depicted in Figure 1b, with the corresponding schemas and views as in example 1. Using lemma 1 one can conclude that all distinctive states a , b and c are distinguishable by actions and views alone. For instance, $\{ml, ml\}(a) = c$, $\{ml, ml\}(b) = d$, $View(\{ml, ml\}(a), v+) = View(\{ml, ml\}(b), \perp)$, and consequently, $\neg ceq(a, b)$. *{end of example}*

The Herbrand models of $CT(E)$ are in a one to one correspondence with the answer sets [Gelfond and Lifschitz, 1991] of the logic program in Figure 2.⁵ In this program, the X and Y variables range over distinctive states and the variable V ranges over views in E . The sets of rules 4 and 5 are the facts corresponding to the agent's experiences. Rules 6-8 require ceq to be an equivalence class. Rules 8 and 9 are the counterpart of axiom 2. Rule 11 is the counterpart of axiom 3. In order to define the maximality condition of ceq , the auxiliary predicate $p(X, Y, X1, Y1)$ is introduced. This predicate reads as "If X and Y were the same, then $X1$ and $Y1$ would be the same". The predicate $dist(X, Y)$ defines when distinctive states X and Y are distinguishable. Constraint 12 establishes the maximality condition on ceq : $ceq(X, Y)$ should be the case unless X and Y are distinguishable.⁶

$$\begin{aligned} &\{cs(ds, a, ds') \leftarrow . : cs(ds, a, ds') \in E\} \quad (4) \\ &\{view(ds, v) \leftarrow . : view(ds, v) \in E\} \quad (5) \\ &ceq(X, Y), \neg ceq(X, Y) \leftarrow . \\ &p(X, Y, X, Y) \leftarrow . \\ &p(X, Y, X2, Y1) \leftarrow p(X, Y, X1, Y1), ceq(X1, X2). \\ &p(X, Y, X1, Y2) \leftarrow p(X, Y, X1, Y1), ceq(Y1, Y2). \\ &p(X, Y, X2, Y2) \leftarrow p(X, Y, X1, Y1), cs(X1, A, X2), cs(Y1, A, Y2). \\ &p(X, Y, Y1, X1) \leftarrow p(X, Y, X1, Y1). \\ &p(X, Y, X1, Y2) \leftarrow p(X, Y, X1, Y1), p(X, Y, Y1, Y2). \\ \\ &dist(X, Y) \leftarrow p(X, Y, X1, Y1), view(X1, V), not view(Y1, V). \\ &dist(X, Y) \leftarrow p(X, Y, X1, Y1), not view(X1, V), view(Y1, V). \\ &\leftarrow not ceq(X, X). \quad (6) \\ &\leftarrow ceq(X, Y), not ceq(Y, X). \quad (7) \\ &\leftarrow ceq(X, Y), ceq(Y, Z), not ceq(X, Z). \quad (8) \\ &\leftarrow ceq(X, Y), view(X, V), not view(Y, V). \quad (9) \\ &\leftarrow ceq(X, Y), not view(X, V), view(Y, V). \quad (10) \\ &\leftarrow not ceq(X1, Y1), ceq(X, Y), cs(X, A, X1), cs(Y, A, Y1). \quad (11) \\ \\ &\leftarrow not ceq(X, Y), not dist(X, Y). \quad (12) \end{aligned}$$

Figure 2: Logic program associated with CT(E).

⁵See extended version of this paper [Remolina and Kuipers, 2001] for a proof.

⁶We have implemented this logic program in Smodels [Niemelä and Simons, 1997]. In the implementation, one has to add variable domain restrictions to the different rules. For example, rule

$$ceq(X, Y), \neg ceq(X, Y) \leftarrow .$$

becomes

$$ceq(X, Y), \neg ceq(X, Y) \leftarrow dstate(X), dstate(Y)$$

where $dstate$ is our predicate to identify the sort of distinctive states.

4 SSH Topological Level

We are to define the SSH topological theory, $\mathbf{TT}(\mathbf{E})$, associated with a set of experiences E . The language of this theory is a sorted language with sorts for *places*, *paths* and *path directions*.⁷ The main purpose of $\mathbf{TT}(E)$ is to minimize the set of paths and places consistent with the given experiences E . A place can be a *topological place* (hereafter place) or a *region*. A place is a set of distinctive states linked by turn actions. A region is a set of places. We use the predicates *tplace* and *is_region* to identify these subsorts. A path defines an order relation among places connected by travel with no turn actions. They play the role of streets in a city layout. We use the predicate *tpath* to identify the sort of paths. By minimizing the extent of *tplace*, *is_region* and *tpath* we minimize the sort of places and paths respectively.⁸ The language of the SSH topological level includes the following other predicates: *teq(ds, ds')* – distinctive states ds and ds' are *topologically indistinguishable*; *at(ds, p)* – distinctive state ds is at place p ; *along(ds, pa, dir)* – distinctive state ds is along path pa in direction dir ; *OnPath(pa, p)* – place p is on path pa ; *PO(pa, dir, p, q)* – place p is before place q when facing direction dir on path pa (PO stands for Path Order).

$\mathbf{TT}(\mathbf{E})$, is the following nested abnormality theory:

$$\begin{aligned} \forall p, tplace(p) \equiv \neg is_region(p), \forall pa, tpath(pa), & (13) \\ \{min\ is_region : & \\ CT(E), T_block, AT_block \} & \end{aligned}$$

The first line in Axioms 13 says that topological places and regions are the two subsorts of places, and that the predicate *tpath* represents the sort of paths. The block $\mathbf{CT}(\mathbf{E})$ is the one defined in the previous section. The block $\mathbf{T_block}$ defines the predicates \widehat{turn} , \widehat{travel} , and \overline{travel} such that \widehat{turn} is the equivalence closure of the schemas $\langle \cdot, turn, \cdot \rangle$; \widehat{travel} and \overline{travel} are the equivalence and transitive closure of the schemas $\langle \cdot, travel, \cdot \rangle$.

The block $\mathbf{AT_block}$ (Figure 3) is the heart of our theory.⁹ The purpose of this block is to define the extent of the predicates *tpath*, *tplace*, *at*, *along*, *PO* and *teq*, while identifying a minimum set of places and paths that explain E . The block has associated the circumscription policy¹⁰

$\mathbf{circ\ tpath} \succ \mathbf{along} \succ \mathbf{PO} \succ \mathbf{OnPath} \succ \mathbf{tplace\ var\ SSHpred}$

where $\mathbf{SSHpred}$ stands for the tuple of predicates *at*, *teq*, *travel_eq*, and *turn_eq*.¹¹ This circumscription policy states

⁷The sort of directions is completely defined by the symbols *pos* and *neg*.

⁸Notice that our logic has sorts for *places* and *paths* but in order to minimize these sorts we have to explicitly have predicates representing them.

⁹Notice that the predicate *is_region* is not mentioned in the theory of figure 3. In the next section we will add to this theory axioms dealing with regions. For the purpose of this section, the minimization of *is_region* in conjunction with $\forall p, tplace(p) \equiv \neg is_region(p)$ implies (the default) $\forall p\ tplace(p)$.

¹⁰The symbol \succ indicates prioritized circumscription (see [Lifschitz, 1994] section 7.2).

¹¹Block 19 in Figure 3 states that the predicate *turn_eq* corresponds to the relation \widehat{turn} modulo *teq*. Block 31 defines *travel_eq* to be the relation \overline{travel} modulo *teq*.

(among others) that a minimum set of paths is preferred over a minimum set of places. Next we discuss the axioms in $\mathbf{AT_block}$.

$$\begin{aligned} \{ : & \\ teq(ds, ds') \equiv \exists p \{ ceq(ds, ds') \wedge at(ds, p) \wedge at(ds', p) \}, & (14) \\ at(ds, p) \rightarrow tplace(p), & (15) \\ \exists! pat(ds, p), & (16) \\ \langle ds, turn, ds' \rangle \wedge at(ds, p) \rightarrow at(ds', p), & (17) \\ at(ds, p) \wedge at(ds', p) \rightarrow turn_eq(ds, ds'), & (18) \\ \{min\ turn_eq : & (19) \\ teq(ds, ds') \wedge teq(dr, dr') \wedge \widehat{turn}(ds', dr') \rightarrow turn_eq(ds, dr), & \\ turn_eq(ds, ds') \wedge turn_eq(ds', ds'') \rightarrow turn_eq(ds, ds'') \} & \\ along(ds, pa, dir) \rightarrow tpath(pa), & (20) \\ at(ds, p) \wedge at(ds', q) \wedge \overline{travel}(ds, ds') \rightarrow & (21) \\ \exists pa, dir \{ PO(pa, dir, p, q) \wedge along(ds, pa, dir) \wedge along(ds', pa, dir) \}, & \\ along(ds, pa, dir) \wedge along(ds, pa1, dir1) \rightarrow pa = pa1, & (22) \\ at(ds, p) \wedge at(ds', p) \wedge along(ds, pa, dir) \wedge & (23) \\ along(ds', pa, dir) \rightarrow teq(ds, ds'), & \\ \{ \langle ds, turn_desc, ds' \rangle \wedge turn_desc \neq turnAround \wedge & (24) \\ along(ds, pa, dir) \wedge along(ds', pa1, dir1) \} \rightarrow pa \neq pa1, & \\ \langle ds, turnAround, ds' \rangle \rightarrow along(ds, pa, dir) \equiv along(ds', pa, -dir), & (25) \\ PO(pa, pos, p, q) \equiv PO(pa, neg, q, p), & (26) \\ \neg PO(pa, dir, p, p), & (27) \\ PO(pa, dir, p, q) \wedge PO(pa, dir, q, r) \rightarrow PO(pa, dir, p, r), & (28) \\ PO(pa, dir, p, q) \rightarrow OnPath(pa, p) & (29) \\ OnPath(pa, p) \wedge OnPath(pa, q) \wedge tpath(pa) \rightarrow & (30) \\ \exists ds, ds' \{ at(ds, p) \wedge at(ds', q) \wedge travel_eq(ds, ds') \}, & \\ \{min\ travel_eq : & (31) \\ teq(ds, ds') \wedge teq(dr, dr') \wedge \widehat{travel}(ds', dr') \rightarrow travel_eq(ds, dr), & \\ travel_eq(ds, ds') \wedge travel_eq(ds', ds'') \rightarrow travel_eq(ds, ds'') \} & \\ \mathbf{circ\ tpath} \succ \mathbf{along} \succ \mathbf{PO} \succ \mathbf{OnPath} \succ \mathbf{tplace\ var\ SSHpred} & \\ \} & \end{aligned}$$

Figure 3: $\mathbf{AT_block}$.

Predicate *teq* is the equivalence relation defined by axiom 14. $teq(ds, ds')$ is the case whenever ds and ds' cannot be distinguished by views and actions (i.e. $ceq(ds, ds')$) and it is consistent to group ds and ds' into the same place. If we assume that views uniquely identify distinctive states (e.g. $View(ds, V) \wedge View(ds', V) \rightarrow ds = ds'$), then predicates *ceq* and *teq* will reduce to equality. This is expected since all that is required to identify a distinctive state is its view.

Every distinctive state is at a unique place (Axiom 16). Whenever the agent *turns*, it stays at the same place (Axiom 17). Distinctive states grouped into a topological place should be *turn* connected (modulo *teq*) (Axiom 18). *Travel* actions among distinctive states are abstracted to topological paths connecting the places associated with those distinctive states (Axiom 21). A distinctive state is along at most one path (Axiom 22). At each place there is at most one distinctive state along a given path direction (Axiom 23). Turn actions other

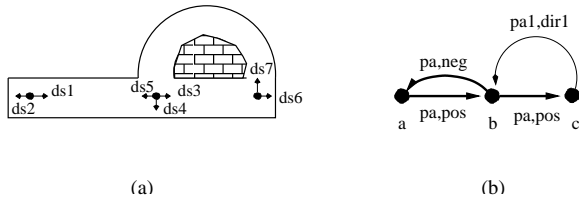


Figure 4: The environment in (a) illustrates a case where different paths intersect at more than one place. (b) depicts the topological map associated with this environment.

than *turnAround* change the path the initial and final distinctive states are at (Axiom 24). *TurnAround* actions relate distinctive states being in the same path but opposite directions (Axiom 25). The order of places in a given path direction is the inverse of the order of places in the other path direction (Axiom 26). Axioms 27 and 28 require $PO(pa, dir, \cdot, \cdot)$ to be a non-reflexive transitive order for the places on pa . Places ordered by a path should belong to that path (Axiom 29). Axiom 30 requires the agent to have traveled among the places on a same path.

Our theory does not assume a “rectilinear” environment where paths intersect at most in one place. It is possible for different paths to have the same order of places (see Figure 4). Topological information can distinguish distinctive states not distinguishable by view and actions.

Example 3

Consider the scenario of example 1. Since the same view is experienced at a , b and c , the extent of ceq is maximized by declaring $ceq = true$. Using the topological theory, from axiom 16 we conclude that there exist places P and Q , such that $at(a, P)$ and $at(c, Q)$. Since it is the case that $\overline{travel}(a, c)$, from axioms 21 and 27 we conclude, for instance, that $P \neq Q$. Distinctive states a and c are topologically distinguishable though they are “causally indistinguishable” (i.e. $ceq(a, c) \wedge \neg teq(a, c)$). $\{end\ of\ example\}$

Given a minimal model M of $TT(E)$, the SSH topological map is defined by the extent in M of $tpath$, $tplace$, $along$, PO and at . Since the positive and negative direction of a path are chosen arbitrarily (Axiom 21), there is not a unique minimal model for $TT(E)$. We will consider these “up to path direction isomorphic” models to be the same. However, it is still the case that the theory $TT(E)$ has minimal models that are not isomorphic up to path direction (see Figure 5).

5 SSH Boundary Regions

In addition to connectivity and order among places and paths, the topological map includes topological boundary relations: assertions that a place lies to the right of, to the left of, or on a path. In order to determine boundary relations we formally state the following default heuristic. Suppose the agent is at an intersection on a given path, and it then turns right. If the agent now travels, any place it finds while traveling with no turns will be on the right of the starting path. When conflicting information exists about whether a place is to the right or

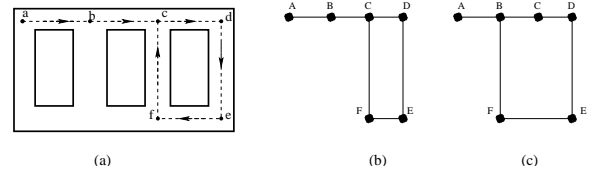


Figure 5: (a) The robot goes around the block visiting places A, \dots, F, C in the order suggested in the figure. Intersections B and C look alike to the agent. Two minimal models can be associated with the set of experiences in (a) (see (b) and (c)). Topological information is not enough to decide whether the agent is back to B or C . Notice that if the agent accumulates more information, by turning at c and traveling to d , then it can deduce that the topology of the environment is the one in (b). In addition, when available, metrical information can be used to refute the incorrect topology.

left of a path, we deduce no boundary relation (see Figure 6).

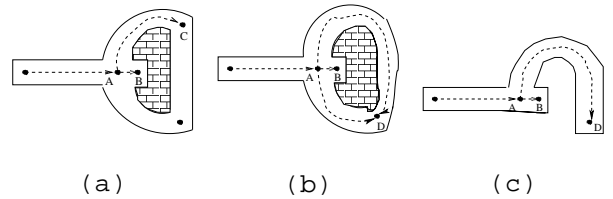


Figure 6: Different environments illustrating how our default to determine boundary relations works. In (a), we conclude by default that place C is to the left of the path from A to B . In (b) we conclude nothing about the location of place D with respect to the path from A to B . In (c), we conclude that place D is to the left of the path from A to B . This is the case since there is no information to conclude otherwise.

We use the predicates $TotheRightOf/TotheLeftOf(p1, pa, dir, pa1, dir1)$ to represent the facts that (i) $p1$ is a place on both paths, pa and $pa1$, and (ii) when the agent is at place $p1$ facing in the direction dir of pa , after executing a turn right (left) action, the agent will be facing on the direction $dir1$ of $pa1$ (see Figure 7). The predicates $TotheLeftOf$ and $TotheRightOf$ are derived from the actions performed by the agent at a place:

$$\langle ds, turnRight, ds1 \rangle \wedge at(ds, p) \wedge along(ds, pa, dir) \wedge along(ds1, pa1, dir1) \rightarrow TotheRightOf(p, pa, dir, pa1, dir1) \quad (32)$$

We use the predicates $LeftOf(pa, dir, lr)$ and $RightOf(pa, dir, rr)$ to denote that region lr (rr) is the left (right) region of path pa with respect to the path’s direction dir . The left/right regions of a path are unique, disjoint, and related when changing the path direction (i.e. $LeftOf(pa, dir, r) \equiv RightOf(pa, -dir, r)$). From the relative orientation between paths at a place, we deduce the relative location of places with respect to a path (see Figure 7):¹²

$$TotheRightOf(p1, pa, dir, pa1, dir1) \wedge PO(pa1, dir1, p1, p) \wedge RightOf(pa, dir, rr) \wedge \neg Ab(pa, p) \rightarrow in_region(p, rr) \quad (33)$$

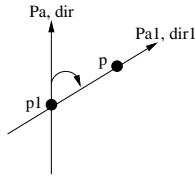


Figure 7: Path $pa1$ is to the right of path pa at place $p1$. Place p is after place $p1$ on path $pa1$. By default, we conclude that place p is to the right of path pa .

The predicate **Ab** is the standard “abnormality” predicate used to represent defaults in circumscriptive theories [Lifschitz, 1994]. Axiom 33 states that “normally”, if at place $p1$ path $pa1$ is to the right of path pa , and place p is after $p1$ on path $pa1$, then it should be the case that p is on the right of pa (Figure 7). In order to capture this default, boundary regions domain theory axioms¹³ are added to the block **AT_block** (see Figure 3). Since we are interested in the extent of the new predicates *in_region*, *LeftOf*, *RightOf*, *ToTheLeftOf* and *ToTheRightOf*, we allow them to vary in the circumscription policy. The new circumscription policy becomes

$$\text{circ } tpath \succ \text{along} \succ PO \succ \text{Onpath} \succ \text{Ab} \succ \text{is_region} \succ \\ \text{in_region} \succ tplace \text{ var } newSSHpred$$

where $newSSHpred$ stands for the tuple of predicates *at*, *along*, *teq*, *travel_eq*, *turn_eq*, **LeftOf**, **RightOf**, **ToTheLeftOf**, and **ToTheRightOf**. The circumscription policy states that boundary relations should be established even at the expense of having more places on the map. In addition, by minimizing the predicates *is_region* and *in_region*, we require the models of our theory to have only the regions that are explicitly created by the agent, and not arbitrary ones.

Example 4

Boundary relations determine distinctions among environment states that could not be derived from the connectivity of places alone. Consider an agent visiting the different corners of a square room in the order suggested by Figure 8a. In addition, suppose the agent defines *views* by characterizing the direction of walls and open space. Accordingly, the agent experiences *four* different views, $v1-v4$, in this environment.

The set of experiences E in the environment are:

$$\begin{array}{lll} View(ds1, v1) & View(ds2, v2) & View(ds3, v1) \\ View(ds4, v2) & View(ds5, v1) & \langle ds1, \text{turnRight}, ds2 \rangle \\ \langle ds2, \text{travel}, ds3 \rangle & \langle ds4, \text{travel}, ds5 \rangle & \langle ds3, \text{turnRight}, ds4 \rangle \end{array}$$

Suppose that the agent does not use boundary regions when building the topological map. Then the minimal topological model associated with E has two paths¹⁴ and two places. In this model, $teq(ds1, ds5)$ is the case. The environment looks perfectly symmetric to the agent (Figure 8b).!!

Suppose now that the agent relies on boundary regions. Let P , Q , R , be the topological places associated with

¹²The predicate $in_region(p, r)$ states that *place* p is in *region* r .

¹³In the spirit of axioms 32-33.

¹⁴Notice that from $\langle ds3, \text{turnRight}, ds4 \rangle$ and Axiom 24 we can deduce that $Pa \neq Pb$ in Figure 8b.

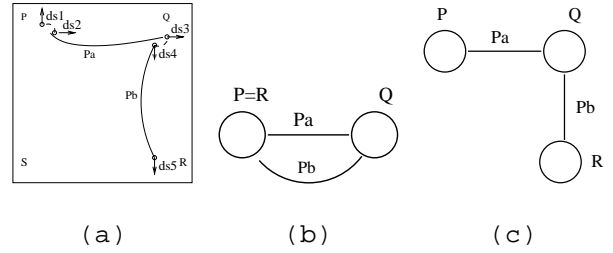


Figure 8: (a) The figure shows the sequence of actions followed by an agent while navigating a square room. Starting at distinctive state $ds1$, distinctive states are visited in the order suggested by their number. Dashed lines indicate Turn actions. Solid lines indicate Travel actions. (b) and (c) depict the topological map associated with the environment in (a) without and using boundary regions, respectively.

$ds1$, $ds3$ and $ds5$ respectively. From Axiom 21, let Pa , Pb , dir_a and dir_b be such that $PO(Pa, dir_a, P, Q)$, $along(ds2, Pa, dir_a)$, $along(ds3, Pa, dir_a)$, $PO(Pb, dir_b, Q, R)$, $along(ds4, Pb, dir_b)$, and $along(ds5, Pb, dir_b)$ hold. From Axiom 32 we can conclude then $ToTheRightOf(Q, Pa, dir_a, Pb, dir_b)$. In the proposed model, the extent of Ab is minimized by declaring $Ab = false$ and consequently from Axiom 33 we conclude $in_region(R, right(Pa, dir_a))$ where $right(Pa, dir_a)$ denotes the right region of Pa when facing dir_a . Finally, since a path and its regions are disjoint, and $OnPath(Pa, P)$ is the case, we conclude $P \neq R$ and so $\neq teq(ds1, ds5)$. The resulting topological map is depicted in Figure 8c. {end of example}

If the agent’s sensory capabilities are so impoverished that many distinctive states are perceived to be similar, then metrical information could be used to distinguish different environment states. Figure 9 summarizes different representations an agent could build depending on the spatial properties it relies on.

6 Conclusions

Starting with an informal description of the SSH we have formally specified its intended models. These models correspond to the models of the circumscriptive theory $TT(E)$. The formal account of the theory allows us to illustrate the deductive power of the different SSH ontologies. For instance, example 4 shows how the use of boundary relations allows the agent to determine distinctions among environment states that could not be derived from the connectivity of places and paths alone.

The theory $TT(E)$ is rather complex so it may be difficult to determine the effect of the different defaults in combination. However, it is possible to translate this theory into a logic program whose answer sets determine the models of $TT(E)$. We have illustrated the case for the SSH causal theory $CT(E)$, but the same techniques apply for $TT(E)$. The major subtleties in the translation are the minimality and maximality conditions associated with the theory. We have used Smodels to calculate the models of $TT(E)$ and confirm that the theory yields

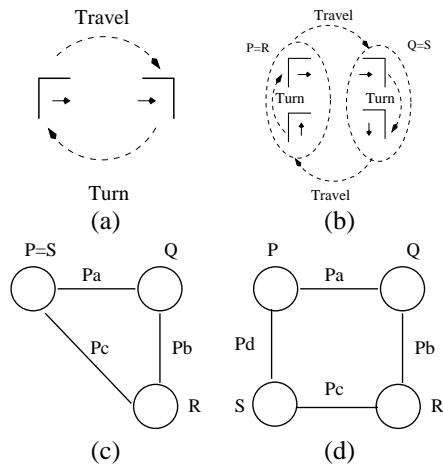


Figure 9: Consider the same environment and agent as in figure 8. Assumes the agent keeps turning right and following the left wall until it is back to distinctive state $ds1$, at place P . Only two kind of views \models and \Rightarrow are observed by the agent. Next we summarizes different maps the agent could build depending on the spatial properties it relies on. (a) If the agent only relies on causal information, the map consists of two states. (b) When topological information is used, but without boundary relations, the map consists of four states and two places. (c) When boundary relations are used, the map consists of six states and three places. There is no fixed correspondence between the three places in the map and the four indistinguishable places in the real world. (d) If metrical information is accurate enough to refute the hypothesis $P = S$, the map will consist of eight states and four places.

the intended models. However, when the number of distinctive states is big, Smodels may not be able to ground the theory as the number of rules associated with the program grows exponentially. We are still working on solving this problem.

Acknowledgments

We are grateful to Vladimir Lifschitz for his valuable feedback during this work and for suggesting the use of Smodels to implement the ideas proposed here. We also thank the anonymous referees for their valuable comments on this paper.

References

[Basye *et al.*, 1995] K. Basye, T. Dean, and L. P. Kaelbling. Learning dynamics: System identification for perceptually challenged agents. *Artificial Intelligence*, 72(1):139–171, 1995.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Kortenkamp *et al.*, 1995] D. Kortenkamp, E. Chown, and S. Kaplan. Prototypes, locations, and associative networks (PLAN): towards a unified theory of cognitive mapping. *Cognitive Science*, 19:1–51, 1995.

[Kuipers and Byun, 1988] B. Kuipers and Y. T. Byun. A robust qualitative method for spatial learning in unknown environments. In Morgan Kaufmann, editor, *AAAI-88*, 1988.

[Kuipers and Byun, 1991] B. J. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[Kuipers, 1977] B. Kuipers. *Representing Knowledge of Large-Scale Space*. PhD thesis, Artificial Intelligence Laboratory, MIT, 1977.

[Kuipers, 1978] B. J. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[Kuipers, 2000] B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[Leiser and Zilbershatz, 1989] D. Leiser and A. Zilbershatz. THE TRAVELLER: a computational model of spatial network learning. *Environment and Behavior*, 21(4):435–463, 1989.

[Lifschitz, 1994] V. Lifschitz. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 297–352. Oxford University Press, 1994.

[Lifschitz, 1995] V. Lifschitz. Nested abnormality theories. *Artificial Intelligence*, (74):351–365, 1995.

[McDermott and Davis, 1984] D. V. McDermott and E. Davis. Planning routes through uncertain territory. *Artificial Intelligence*, 22:107–156, 1984.

[Niemelä and Simons, 1997] I. Niemelä and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *4th International Conference on Logic Programming and Non-monotonic Reasoning*, number 1265 in LNCS, pages 420–429. Springer-Verlag, 1997.

[Remolina and Kuipers, 1998] E. Remolina and B. Kuipers. Towards a formalization of the Spatial Semantic Hierarchy. In *Fourth Symposium on Logical Formalizations of Commonsense Reasoning, London*, January 1998.

[Remolina and Kuipers, 2001] E. Remolina and B. Kuipers. A logical account of causal and topological maps. Technical Report AI01-288, The University of Texas at Austin, <http://www.cs.utexas.edu/users/qr>, April 2001.

[Shanahan, 1996] M. Shanahan. Noise and the common sense informatic situation for a mobile robot. In *AAAI-96*, pages 1098–1103, 1996.

[Shanahan, 1997] M. Shanahan. Noise, non-determinism and spatial uncertainty. In *AAAI-97*, pages 153–158, 1997.

[Yeap, 1988] W. K. Yeap. Towards a computational theory of cognitive maps. *Artificial Intelligence*, 34:297–360, 1988.

On-Line Execution of cc-Golog Plans

Henrik Grosskreutz and Gerhard Lakemeyer

Department of Computer Science V

Aachen University of Technology

52056 Aachen, Germany

{grosskreutz,gerhard}@cs.rwth-aachen.de

Abstract

Previously, the plan language *cc-Golog* was introduced for the purpose of specifying event-driven behavior typically found in robot controllers. So far, however, *cc-Golog* is usable only for projecting the outcome of a plan and it is unclear how to actually execute plans on-line on a robot. In this paper, we provide such an execution model for *cc-Golog* and, in addition, show how to interleave execution with a new kind of time-bounded projection. Along the way we also demonstrate how a typical robot control architecture where a high-level controller communicates with low-level processes via messages can be directly modelled in *cc-Golog*.

1 Introduction

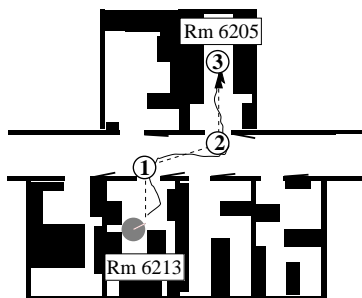


Figure 1: Actual Trajectory and Approximation

Consider a robot which, while engaged in a delivery task, is to move from Room 6213 to Room 6205 (see Figure 1). In a modern robot architecture like XAVIER [Simmons *et al.*, 1997] or RHINO [Burgard *et al.*, 2000], the controller would issue a message to a low-level navigation process telling it the new destination (Rm6205), and the navigation process would then move the robot to its desired location, making use of intermediate goal points like those outside doorways (nodes 1 and 2 in the figure) for robustness.

Besides executing such plans, an intelligent robot controller should be able to reason about its actions, that is, consider, for example, whether it is better to first charge the batteries or to continue the delivery, which would involve projecting the outcome of the delivery task and checking whether the batteries would still have enough power af-

terwards. Moreover, projection is also valuable for the designer of robot controllers to find out whether a program under development is executable or whether it would satisfy certain goals after execution. In [Grosskreutz and Lakemeyer, 2000a], we proposed such a projection mechanism for *cc-Golog* an extension of *GOLOG* [Levesque *et al.*, 1997] featuring a model of time, continuous change, and the ability to wait for conditions (like arriving in Rm6205) to become true. While tasks like the above can be modelled quite naturally in *cc-Golog*, it so far remains unclear how to actually execute a *cc-Golog* plan or program (in this paper we use plan and program interchangeably). To get a sense of the problem, consider the actions of initiating moving just outside of Room 6213 and waiting to arrive there. In *cc-Golog* these actions would have the effect of setting the location of the robot to a continuous linear function of time to approximate the robot's trajectory and to advance time to the point when the goal location is reached, respectively. While this seems fine for the purpose of projection, which is sometimes also called off-line execution [de Giacomo and Levesque, 1999], those "effects" are simply inappropriate during actual on-line execution. For one, the robot has no control over the passage of time and, for another, the actual trajectory often follows a function quite different from the idealized approximation (see the curved vs. the straight (dotted) line in Figure 1). What is needed instead, it seems, are frequent sensor readings telling the robot about the current time and location, which should be used instead of the models of how time passes or how the robot moves. A waiting action would then simply reduce to a test whether the goal has been reached. Another minor complication is that actions that are part of the model of a low-level process such as navigation need to be ignored during execution except for an action that activates the process. Finally, while projection in *cc-Golog* so far is limited to a complete program starting in the initial state, one would often like to project on the fly during execution, similar to the search operator of [de Giacomo and Levesque, 1999]. However, for reasons of efficiency, we would like to go beyond that and allow for a restricted projection of a program, which only searches up to a (temporally) limited horizon. For instance, if the robot is in the middle of a delivery but near the docking station, we want to enable it to find out whether the coming activities would allow it to operate for at least another 5 minutes and, if not, decide to charge the batteries first.

In this paper, we show how all this can be done in *cc-Golog*. For that we first show how a robot control architec-

ture where a high-level controller communicates with low-level processes via messages can be modelled directly in **cc-Golog**. The main advantage is that there is a clear separation of the actions of the high-level controller from those of the low-level processes. Then we discuss the changes necessary to use the same **cc-Golog** program both for on-line execution and projection. Finally, we show how a time-bounded projection mechanism can be defined and interleaved with execution.

The rest of the paper is organized as follows. In Section 2, we briefly review **cc-Golog** and the situation calculus on which **cc-Golog** is based. Then we describe how to model the robot control architecture and give an example of a model of a low-level navigation process. In Sections 4 and 5 we discuss the changes needed to allow on-line execution of **cc-Golog** programs and how on-line execution and projection can be interleaved. We end with a brief summary and discussion of related work.

2 Preliminaries

The Situation Calculus The semantics of **cc-Golog** is based on an extension of the situation calculus [McCarthy, 1963]. We will only go briefly over the basic situation calculus: all terms in the language are one of the following sorts: ordinary objects, actions, or situations; there is a special constant S_0 used to denote the *initial situation*, namely that situation in which no actions have yet occurred; there is a distinguished binary function symbol *do* where $do(a, s)$ denotes the successor situation to s resulting from performing the action a ; relations whose truth values vary from situation to situation are called relational *fluents*, and are denoted by predicate symbols taking a situation term as their last argument; similarly, functions varying across situations are called functional fluents and are denoted analogously; finally, there is a special predicate $Poss(a, s)$ used to state that action a is executable in situation s .

Within this language, we can formulate theories which describe how the world changes as the result of the available actions. One possibility is a *basic action theory* of the following form [Levesque *et al.*, 1998]:

- Axioms describing the initial situation, S_0 .
- Action precondition axioms, one for each primitive action a , characterizing $Poss(a, s)$.
- Successor state axioms (SSAs), one for each fluent F , stating under what conditions $F(\vec{x}, do(a, s))$ holds as a function of what holds in situation s . These take the place of the so-called effect axioms, but also provide a solution to the frame problem.
- Domain closure and unique name axioms for the actions.
- Foundational, domain independent axioms.

Let us now consider extending this basic situation calculus by time and continuous change according to [Grosskreutz and Lakemeyer, 2000a]. Note that these extensions are intended for the purpose of projections only. In Section 4 we will discuss the necessary changes to allow on-line executions as well. We begin by adding two new sorts real and time.¹ Similar to Pinto and Reiter [Pinto, 1997; Reiter, 1996], we

¹For simplicity, the reals are not axiomatized and we assume their standard interpretations together with the usual operations and ordering relations. The sort time ranges over the reals as well.

introduce a special unary functional fluent *start* connecting situations and time. The intuition is that $start(s)$ denotes the time when situation s begins. (We defer the formal definition of *start* until after the discussion of the passage of time.)

As in [Pinto, 1997], continuous change is modelled based on the idea that a fluent like the position of a robot takes as value a *function of time*. For example, in the situation where a robot starts moving, its 2-dimensional location can be characterized (in a somewhat idealized fashion) by a linear function of time, starting at its actual position and moving toward its destination. We call functional fluents whose values are continuous functions *continuous fluents*. In order to represent the value of continuous fluents, we add to the language another sort t-function, whose elements are meant to be functions of time. We assume that there are only finitely many function symbols of type t-function and we require *domain closure and unique names axioms* for them, just as in the case of primitive actions. Furthermore, one needs to specify what values these functions have at any particular time t . This is done with the help of the special binary function *val*. The following axiom illustrates the use of *val*, where the values of constant functions and a special kind of linear functions are defined.

$$\begin{aligned} val(const(x, y), t) &= \langle x, y \rangle; \\ val(linear(x, y, v_x, v_y, t_0), t) &= \langle x', y' \rangle \equiv \\ & x = x + v_x * (t - t_0) \wedge y' = y + v_y * (t - t_0) \end{aligned}$$

While $const(x, y)$ always evaluates to the tuple $\langle x, y \rangle$, $linear(x, y, v_x, v_y, t_0)$ is a linear function intended to approximate the movement of a robot in 2-dimensional space (we ignore the details of representing tuples of reals in logic).

Motivated by the fact that during the execution of plans time passes merely when the high-level controller is waiting, the fluent *start* changes its value only as a result of the special primitive action $waitFor(\phi)$. The intuition is as follows. Normally, every action happens immediately, that is, the starting time of the situation after doing a in s is the same as the starting time of s . The only exception is $waitFor(\phi)$: whenever this action occurs, the starting time of the resulting situation is advanced to the earliest time in the future when ϕ becomes true.

The arguments of $waitFor$ are restricted to so-called t-forms, which are special formulas involving continuous fluents with the situation argument suppressed (see [Grosskreutz and Lakemeyer, 2000a] for details). An example is $(battLevel \leq 46)$. Given a t-form ϕ , $\phi[s, t]$ denotes ϕ with every continuous fluent evaluated in situation s at time t . For example, $(battLevel \leq 46)[s, t]$ becomes $(val(battLevel(s), t) \leq 46)$. The following axiom defines the least time point after the start of s where ϕ becomes true:

$$\begin{aligned} ltp(\phi, s, t) &\equiv \phi[s, t] \wedge t \geq start(s) \wedge \\ &\forall t'. start(s) \leq t' < t \supset \neg \phi[s, t'] \end{aligned}$$

A $waitFor(\phi)$ -action is possible iff ϕ has a least time point:

$$Poss(waitFor(\phi), s) \equiv \exists t. ltp(\phi, s, t).$$

Finally, the following successor state axiom for *start* captures the intuition that the starting time of a situation changes only as a result of a $waitFor(\phi)$, in which case it advances to the earliest time in the future when ϕ holds.

$$\begin{aligned} Poss(a, s) \supset [start(do(a, s)) = t \equiv \\ \exists \phi. a = waitFor(\phi) \wedge ltp(\phi, s, t) \vee \\ \forall \phi. a \neq waitFor(\phi) \wedge t = start(s)]. \end{aligned}$$

Example To illustrate the use of these definitions, let us model how a robot’s location changes as a result of primitive actions. For simplicity, we ignore the robot’s orientation and represent its position by the continuous fluent *robotLoc*, which evaluates to a tuple of reals. There are two types of actions that affect the location of the robot: *startGo*($\langle x, y \rangle$), which initiates moving the robot towards position $\langle x, y \rangle$,² and *endGo* which stops the movement of the robot. We assume that *startGo* is only possible if the destination differs from the robot’s current location. Then we obtain the following successor state axiom for *robotLoc*.

$$\begin{aligned} Poss(a, s) \supset [robotLoc(do(a, s)) = f \equiv & \\ \exists t, x, y. t = start(s) \wedge val(robotLoc(s), t) = \langle x, y \rangle \wedge & \\ [\exists x', y', v_x, v_y. a = startGo(\langle x', y' \rangle) \wedge v_x = (x' - x)/\nu & \\ \wedge v_y = (y' - y)/\nu \wedge f = linear(x, y, v_x, v_y, t) \vee & \\ a = endGo \wedge f = const(x, y) \vee & \\ \forall x', y'. a \neq startGo(x', y') \wedge a \neq endGo \wedge & \\ f = robotLoc(s)] & \end{aligned}$$

where, $\nu \doteq \sqrt{(x' - x)^2 + (y' - y)^2}$ is a normalizing factor needed in order to ensure that the total 2-dimensional velocity does not exceed 1.

The variables x and y refer to the coordinates of the robot. After *startGo*($\langle x', y' \rangle$), *robotLoc* has as value a linear t-function starting at the current position and moving toward $\langle x', y' \rangle$. After *endGo*, it is *const*($\langle x, y \rangle$). Finally, if a is neither a *startGo* nor a *endGo* action, *robotLoc* remains unchanged.

cc-Golog We now turn to **cc-Golog**, which offers constructs such as sequences, iteration and procedures to define complex actions. In addition, parallel actions are introduced with a conventional interleaving semantics.

α	primitive action
<i>waitFor</i> (ϕ)	wait until condition ϕ becomes true
$\phi?$	test action
$[\sigma_1, \sigma_2]$	sequence
<i>if</i> (ϕ, σ_1, σ_2)	conditional
<i>while</i> (ϕ, σ)	loop
<i>withPol</i> (σ_1, σ_2)	prioritized execution until σ_2 ends ³
<i>proc</i> $\beta(x)\sigma$	procedure definition

Using **cc-Golog**, it is possible to specify event-driven behavior quite naturally. For example, the following program specifies that the robot is to travel to Room 6205 and say “In Hallway” if it ever enters the hallway on its way.

prg1 \doteq *withPol*([*waitFor*(*inHallway*), *say*(“In Hallway”)], [*send*(*destRm*, 6205), *reg*(*reached*) = 6205?])

Here the *send* action indicates that the robot assigns the room number to a register, which leads to an activation of the navigation process to head to that room, and then blocks execution until the register *reached* signals that the navigation process has completed successfully. The details of this architecture are found in Section 3.

The semantics of **cc-Golog** is specified similar to that of **ConGolog** [Giacomo *et al.*, 2000]. Its main ingredients are a predicate *Trans*(σ, s, δ, s'), which defines single steps of

²For simplicity, we model the robot as always traveling at speed 1m/s. In a more realistic model, we would also consider different velocities.

³Unprioritized concurrency can be defined as well, but is omitted since we do not use it in the paper.

computation, and another predicate *Final*(σ, s). *Final* specifies which configurations (σ, s) are final, meaning that the computation can be considered completed. This is the case, roughly, when the remaining program is *nil*, but not if there is still a primitive action or test action to be executed. We leave out the details for reasons of space.

Intuitively, the predicate *Trans*(σ, s, δ, s') associates with a given program σ and situation s a new situation s' that results from executing a primitive action in s , and a new program δ that represents what remains of σ after having performed that action. For space reasons, we only list a few of the axioms for *Trans*.

$$Trans(\alpha, s, \delta, s') \equiv Poss(\alpha, s) \wedge \delta = nil \wedge s' = do(\alpha, s)$$

$$Trans(nil, s, \delta, s') \equiv FALSE$$

$$Trans(\phi?, s, \delta, s') \equiv \phi[s] \wedge \delta = nil \wedge s' = s^4$$

$$Trans(seq(\sigma_1, \sigma_2), s, \delta, s') \equiv$$

$$Final(\sigma_1, s) \wedge Trans(\sigma_2, s, \delta, s') \vee$$

$$\exists \delta'. Trans(\sigma_1, s, \delta', s') \wedge \delta = seq(\delta', \sigma_2)$$

$$Trans(withPol(\sigma_1, \sigma_2), s, \delta, s') \equiv \neg Final(\sigma_2, s) \wedge$$

$$\exists \delta_1. Trans(\sigma_1, s, \delta_1, s') \wedge \delta = withPol(\delta_1, \sigma_2) \wedge$$

$$\forall \delta_2, s_2. Trans(\sigma_2, s, \delta_2, s_2) \supset start(s') \leq start(s_2)]$$

$$\vee \exists \delta_2. Trans(\sigma_2, s, \delta_2, s') \wedge \delta = withPol(\sigma_1, \delta_2) \wedge$$

$$\forall \delta_1, s_1. Trans(\sigma_1, s, \delta_1, s_1) \supset start(s') < start(s_1)]$$

A transition of a primitive action requires it to be possible in the current situation, leaving nothing (*nil*) to be done afterwards. The execution of σ_2 with policy σ_1 means that one action of one of the programs is performed, whereby actions which can be executed earlier are always preferred. If both σ_1 and σ_2 are about to execute an action at the same time, the policy σ_1 takes precedence.⁵

A final situation s' reachable after a finite number of transitions from a starting situation is identified with the situation resulting from a possible execution trace of program σ , starting in situation s ; this is captured by the predicate *Do*(σ, s, s'), which is defined in terms of *Trans**, the transitive closure of *Trans*:

$$Do(\delta, s, s') \equiv \exists \delta'. Trans^*(\delta, s, \delta', s') \wedge Final(\delta', s')$$

$$Trans^*(\delta, s, \delta', s') \equiv \forall T[... \supset T(\delta, s, \delta', s')]$$

where the ellipsis stands for the universal closure of the conjunction of the following formulas:

$$T(\delta, s, \delta, s)$$

$$Trans(\delta, s, \delta'', s'') \wedge T(\delta'', s'', \delta', s') \supset T(\delta, s, \delta', s')$$

Given a program δ , proving that δ is executable in the initial situation then amounts to proving $\Sigma \models \exists s Do(\delta, S_0, s)$, where Σ consists of the above axioms for **cc-Golog** together with a basic action theory in the situation calculus.

3 A Robot Control Architecture

As indicated in the introduction, in modern robot architectures like XAVIER [Simmons *et al.*, 1997] and RHINO [Burgard *et al.*, 2000], the high-level controller does not directly operate the robot’s physical sensors and effectors. Instead, it

⁴Here, ϕ stands for a situation calculus formula with all situation arguments suppressed. $\phi[s]$ will denote the formula obtained by restoring situation variable s to all fluents appearing in ϕ .

⁵Note that *Trans* requires a reification of formulas and programs in the logical language. See [Giacomo *et al.*, 2000] for how this can be done.

activates and deactivates specialized processes like a navigation process, to which we will refer as *low-level processes*. The job of the high-level controller is then to combine the activation and deactivation of these routines in a way to fulfill the overall goal. It turns out that cc-Golog allows a logical reconstruction of this type of architecture in a fairly natural way. The overall architecture is illustrated in Figure 2.

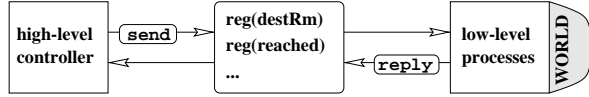


Figure 2: Robot Control Architecture

The communication between the low-level processes and the high-level plan is achieved through the special fluent *reg*. The high-level interpreter can affect the value of *reg* by means of the special action *send(id, val)* which assigns *reg(id)* the value *val*. The intuition is that in order to activate a low-level process, the high-level controller executes a *send* action. For example, the execution of *send(destRm, 6205)* would tell the navigation process to start moving toward Room 6205.

On the other hand, the low-level processes can provide the high-level controller with information by means of the (exogenous) action *reply(id, val)*. For example, in order to inform the high-level controller that it has reached its destination, the low-level process would cause an exogenous *reply(reached, 6205)* action. The following successor state axiom specifies how *reg* is affected by *send* and *reply*.

$$\begin{aligned} Poss(a, s) \supset [reg(id, do(a, s)) = val \equiv \\ a = send(id, val) \vee a = reply(id, val) \vee reg(id, s) = val \\ \wedge \neg(\exists r, v. a = send(r, v) \vee a = reply(r, v))] \end{aligned}$$

3.1 Modeling Low-level Processes as cc-Golog Procedures

To describe complex low-level processes like a navigation process, we model low-level processes as cc-Golog *procedures*.⁶ Given a faithful characterization of the low-level processes in terms of cc-Golog procedures, we can then *project* the effect of the activation of these processes using their corresponding cc-Golog models. We stress that these procedures are not meant to be executed, but rather represent a model of the effects of the corresponding low-level process.

As an example, let us consider a low-level navigation process. It is activated through a *send(destRm, R)* action, which assigns *reg(destRm)* the value *R* and tells the process to travel to room *R*. The process causes the robot to move towards *R* until the destination is reached, where it informs the high-level controller of the arrival by means of an *reply(reached, R)* action.

We model this behavior by the cc-Golog procedure *navProc*. *navProc* makes use of the following functions: *exitOf(R)*, *inside(R)* and *currentRoom*. *exitOf* maps a room name *R* to a location near the exit of room *R*, and *inside* to location within *R*. *currentRoom* is a function whose value in *s* is the name of the room the robot is in at the beginning of *s*. We write *gotoLoc(d)* as an abbreviation for *[startGo(d), waitFor(near(d))]* and *loop(σ)* for *while(TRUE, σ)*.

⁶This is similar to [Grosskreutz and Lakemeyer, 2000b], where a probabilistic programming language is used to model processes with uncertain outcome.

$$\begin{aligned} navProc \doteq loop([reg(destRm) \neq currentRoom?, \\ if(currentRoom \neq HALLWAY, \\ gotoLoc(exitOf(currentRoom))), \\ gotoLoc(exitOf(reg(destRm))), \\ gotoLoc(inside(reg(destRm))), endGo, \\ reply(reached, reg(destRm))]) \end{aligned}$$

Initially, the navigation process is blocked until *reg(destRm)* is assigned a room name different from the one the robot is actually in. Then *navProc* executes a sequence of *startGo* actions, approximating the trajectory towards the destination room by a polyline with an edge near every door the robot has to travel through (see Figure 1).

3.2 Projection

We will now describe how to project a cc-Golog plan, taking into account the cc-Golog model of the low-level processes. Let *s* be a situation, *ll_{model}* a model of the low-level processes,⁷ and *σ* a cc-Golog program. A projection *s'* can then be identified with the situation that results from the concurrent simulation of *σ* and *ll* starting in *s*.

$$Proj(s, \sigma, ll_{model}, s') \doteq Do(withPol(ll_{model}, \sigma), s, s').$$

Let *AX_{cc}* be the set of foundational axioms of Section 2 together with the domain closure and unique names axioms for t-functions, the axioms required for t-form's, the axioms defining *val*, the precondition axiom for *waitFor*, and axioms stating that the robot is in Room 6213 together with other appropriate descriptions of the environment like a definition of *inHallway* in terms of coordinates on the map. Using *AX_{cc}*, we can project the plan *prg1* from Section 2. We write *do([a₁, ..., a_n], s)* a shorthand for *do(a_n, ..., do(a₁, s) ...)*.

$$\begin{aligned} AX_{cc} \models Proj(S_0, prg1, navProc, s) \equiv \\ s = do([send(destRm, 6205), startGo(l_1) \\ waitFor(near(l_1)), startGo(l_2), \\ waitFor(inHallway), say("In Hallway") \\ waitFor(near(l_2)), startGo(l_3), \\ waitFor(near(l_3)), endGo \\ reply(reached, 6205)], S_0) \end{aligned}$$

The projected execution trace includes a *startGo(l_i)* and *waitFor(near(l_i))* action for every node on the approximating trajectory, where the *l_i* stand for coordinates corresponding to the nodes *i* in Figure 1. Additionally, it includes a *say("In Hallway")* which is executed immediately after the hallway has been reached, which is assumed to happen just after leaving *near(l₁)*. The execution trace ends with *reply(reached, 6205)*, which completes the navigation task.

4 On-Line Execution of cc-Golog plans

Projections such as the above should be understood as a way of assessing whether a program is executable *in principle*. The resulting execution trace is *not* intended as input to the execution mechanism of the robot for several reasons. For one, many of the actions like *startGo(l₁)* only serve to model the navigation process and are not meant to be executed by the high-level controller at all. For another, actions that do belong to the high-level controller like *waitFor(inHallway)* must be interpreted differently during on-line execution. This

⁷In our example, the navigation process is the only low-level process, so we can simply use *navProc*. If there were more relevant low-level processes, *ll_{model}* would amount to the concurrent execution of the individual cc-Golog models.

is because during projection *waitFor* advances time to the least point where the condition is satisfied according to an idealized model of the world (like piece-wise linear trajectories). During actual execution, however, a *waitFor* should intuitively be treated simply as a test, where a condition like being at a certain location is matched against sensor readings reflecting the actual state of affairs.

In order to see how such sensor readings can be obtained, let us briefly look at how actual robots like the RHINO system deal with it. There we find a tight update loop between the low-level system and the high-level controller. This update loop periodically provides the high-level controller with an update of the low-level processes' estimate of continuous properties like the batteries' voltage level or the robot's position (typically several times a second). The period of time between two subsequent updates is so small that for practical purposes the latest update can be regarded as providing not only the correct current time but also accurate values of the continuous fluents at the current time.

Our solution is then to represent the updates by means of a new exogenous action *ccUpdate* and to treat *waitFor*'s simply as special tests during on-line execution. Intuitively, the effect of *ccUpdate* is to set the value of the continuous fluents to the latest estimates of the low-level processes. The actual arguments of *ccUpdate* depend on the continuous fluents that are to be updated. In our example scenario, *ccUpdate* has four arguments: x, y, l and t , where x and y refer to the current position of the robot, l to the current voltage level and t to the current time. To get a sense of the effect of *ccUpdate*, let us consider a new version of the successor state axiom for *robotLoc* of Section 2 suitable for both on-line execution and projection. For simplicity, it is assumed that the actions *startGo* and *endGo* only occur during projection (as part of the model of the low-level navigation process).

$$\begin{aligned}
Poss(a, s) \supset [robotLoc(do(a, s)) = f \equiv \\
& \exists t, x, y, t = start(s) \wedge val(robotLoc(s), t) = \langle x, y \rangle \wedge \\
& [\exists x', y', v_x, v_y. a = startGo(\langle x', y' \rangle) \wedge v_x = (x' - x)/\nu \\
& \quad \wedge v_y = (y' - y)/\nu \wedge \wedge f = linear(x, y, v_x, v_y, t) \vee \\
& a = endGo \wedge f = const(x, y) \vee \\
& \exists x', y', l', t'. a = ccUpdate(x', y', l', t') \\
& \quad \wedge f = const(x', y') \vee \\
& \forall \vec{x}. a \neq startGo(\vec{x}) \wedge a \neq ccUpdate(\vec{x}) \wedge a \neq endGo \wedge \\
& f = robotLoc(s)]
\end{aligned}$$

Note that *ccUpdate* simply "assigns" f the constant function $const(x', y')$, which reflects the idea that, during on-line execution, the linear approximation of the trajectory plays no role and that the actual values $\langle x', y' \rangle$ should be used instead.

Besides updating the value of the continuous fluents, *ccUpdate* is of prime importance because it causes time or, better, the internal clock to advance during execution. Therefore, we need to modify the successor state axiom of the fluent *start*. In order to account for the fact that time advances differently in projections and during on-line execution, we need to explicitly distinguish between the two modes of operation. For that purpose, we introduce a special fluent *online*(s) which can only change its truth value by the special actions *setOnline* and *clipOnline*, respectively. (Specifying the SSA for *online* is easy and left out.) The SSA for *start* then becomes

$$\begin{aligned}
Poss(a, s) \supset [start(do(a, s)) = t \equiv \\
\quad \neg online(s) \wedge \exists \phi. a = waitFor(\phi) \wedge ltp(\phi, s, t) \vee
\end{aligned}$$

$$\begin{aligned}
online(s) \wedge \exists x_u, y_u, l_u. a = ccUpdate(x_u, y_u, l_u, t) \vee \\
t = start(s) \wedge (\neg online(s) \supset \forall \phi. a \neq waitFor(\phi)) \wedge \\
(online(s) \supset \forall \vec{x}. a \neq ccUpdate(\vec{x})).
\end{aligned}$$

In other words, $start(do(a, s))$ is assigned the least time point when the *waitFor*-condition becomes true under off-line execution, or the time value returned by *ccUpdate* under on-line execution, or it remains the same. We remark that we could equivalently write two separate successor state axioms for the on-line and off-line case using the idea of guarded successor state axioms proposed in [Giacomo and Levesque, 1999]. While this would enhance the readability and modularity of the axioms, we do not pursue this idea further here for space reasons.

The precondition axiom for *ccUpdate* ensures that the starting time of legal action sequences is monotonically non-decreasing:

$$Poss(ccUpdate(x, y, l, t), s) \equiv t \geq start(s)$$

A *waitFor*(ϕ) instruction is treated as a special kind of *test* during on-line execution: it succeeds immediately iff the condition ϕ is true *at the beginning* of the actual situation. To ensure this intended behavior, we modify the definition of *Trans* for primitive actions:

$$\begin{aligned}
Trans(\alpha, s, \delta, s') \equiv \\
& Poss(\alpha, s) \wedge \delta = nil \wedge s' = do(\alpha, s) \\
& \quad \wedge [online(s) \supset \forall \phi. \alpha \neq waitFor(\phi)] \\
& \quad \vee online(s) \wedge \exists \phi. \alpha = waitFor(\phi) \wedge \phi[s, start(s)] \\
& \quad \wedge s = s' \wedge \delta = nil
\end{aligned}$$

Finally, we remark that another difference between projection and on-line execution is that during actual execution, whenever the high-level controller executes a *send* action, the interpreter should check whether this signals an activation of a low-level process and, as a side-effect, activate the actual low-level process if necessary.

On-Line Execution The situations which can result from the on-line execution of a plan have a special structure. Basically, they are made up of the sequence of robot actions as defined by the possible transitions (similar to the execution traces defined by $Trans^*$), however with an arbitrary number of exogenous actions between any two robot actions. Formally, the situations s' which can result from the execution of a program σ in situation s can be characterized by the following predicate $Trans^{exo}$:

$$Trans^{exo}(\sigma, s, \delta, s') \equiv \forall T[\dots \supset T(\delta, s, \delta', s')]$$

where the ellipsis stands for the universal closure of the conjunction of the following formulas:

$$\begin{aligned}
& T(\delta, s, \delta, s) \\
& Trans(\delta, s, \delta'', s'') \wedge T(\delta'', s'', \delta', s') \supset T(\delta, s, \delta', s') \\
& exog(a) \supset T(\delta, s, \delta, do(a, s)), \text{ where} \\
& exog(a) \equiv \exists i, n, \vec{x}. a = reply(i, n) \vee a = ccUpdate(\vec{x})
\end{aligned}$$

Note that this differs from the original $Trans^*$ in Section 2 only in the last conjunct, which expresses that an arbitrary number of exogenous actions may occur between any ordinary transition. As an example, let us consider a possible on-line execution trace of *prg1*. *prg1*'s first action is *send(destRm, 6205)*, which results in an activation of the navigation process, which we then assume to provide the high-level controller with *ccUpdate* actions every .25 sec. The execution trace results in situation S_{exec} , where the p_i, q_i stand for appropriate $x - y$ -coordinates along the path of the

robot (we have left out the 3rd argument (voltage level) of *ccUpdate*).

$$S_{exec} \doteq do([send(destRm, 6205), \\ ccUpdate(p_1, q_1, 0.25), ccUpdate(p_2, q_2, 0.5), \\ ccUpdate(p_3, q_3, 0.75), ccUpdate(p_4, q_4, 1.0), \\ say("In Hallway"), \dots, ccUpdate(p_{30}, q_{30}, 7.5), \\ reply(reached, 6205)], S_0).$$

It is not hard to see that S_{exec} is a legal on-line execution trace of the introductory cc-Golog program $prg1$, provided the coordinates $\langle p_4, q_4 \rangle$ are the first to satisfy *inHallway* and $\langle p_{30}, q_{30} \rangle$ is near the destination (l_3). Note that *waitFor* no longer occurs in the action sequence since it is now merely a test. Let AX_{ccx} be similar to AX_{cc} but with the new definitions of this section concerning *Trans*, *start*, *ccUpdate* and *robotLoc*. Let Γ be $AX_{ccx} \cup online(S_0)$. Then

$$\Gamma \models \exists \delta. Trans^{exo}(prg1, S_0, \delta, S_{exec}) \wedge Final(\delta, S_{exec})$$

5 Interleaving Projection and On-Line Execution

[de Giacomo and Levesque, 1999] suggest that it is often useful to interleave on-line execution and projection. With our model of time we can take this idea one step further and define projection during on-line execution with the possibility to explicitly put a time-bound on the projection.

5.1 Projection in non-initial Situations

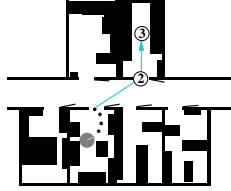


Figure 3: Projection during Execution

So far, we have only considered projection of cc-Golog in the initial situation. To see how things work in non-initial situation, let us again consider our example scenario, where projection makes use of a model of the navigation process. Suppose we are in the situation illustrated in Figure 3, resulting from the on-line execution of $prg1$. The controller begins execution by activating the navigation process through $send(destRm, 6205)$, after which the low-level process provides the high-level controller with a sequence of *ccUpdates* (the *ccUpdates* are indicated as black dots in the figure). The controller is now in situation S_1 with

$$S_1 \doteq do([send(destRm, 6205), \\ ccUpdate(p_1, q_1, 0.25), ccUpdate(p_2, q_2, 0.5), \\ ccUpdate(p_3, q_3, 0.75), ccUpdate(p_4, q_4, 1.0)], S_0).$$

First, let us consider the value of the continuous fluent *robotLoc* in S_1 . Let Γ be defined as before. Then

$$\Gamma \models robotLoc(S_1) = const(p_4, q_4).$$

We can also infer that what remains from $prg1$ in S_1 differs from the initial plan (because $send(destRm, 6205)$ has already been executed).

$$\Gamma \models \exists \delta. Trans^{exo}(prg1, S_0, \delta, S_1) \wedge \\ \delta = withPol([waitFor(inHallway), say("In Hallway")], \\ [reg(reached) = 6205])?$$

Given the updated value of *robotLoc*, we can now correctly project that the remaining plan will cause the robot to directly travel to its destination. Note the use of *clipOnline*, which forces a switch to projection mode.

$$\Gamma \models \forall \delta, s. Trans^{exo}(prg1, S_0, \delta, S_1) \supset \\ Proj(do(clipOnline, S_1), \delta, navProc, s) \equiv \\ s = do([startGo(l_2), \\ waitFor(inHallway), say("In Hallway"), \\ waitFor(near(l_2)), startGo(l_3), \\ waitFor(near(l_3)), endGo, \\ reply(reached, 6205)], do(clipOnline, S_1))]$$

The reason why projections in non-initial situations of the example scenario are fairly straightforward is that we never have to remember the state the actual navigation process is in. More precisely, for any given position, the cc-Golog model of the navigation process yields an appropriate approximation of the remaining trajectory of the robot (as illustrated in the above projection). While many low-level processes used in mobile robots seem to have this property, we remark that processes where one needs to keep track of their internal state during execution can be dealt with in a way similar to [Grosskreutz, 2000]. The details, however, are outside the scope of this paper.

5.2 Projection Tests

In order to allow for a time-bounded lookahead, we define the predicate $Proj(\phi, t, \sigma, s)$, which is true, roughly, if ϕ holds at time t in the projected execution of a plan σ in situation s . Let ll_{model} refer to an appropriate model of the low-level processes, as before.

$$Proj(\phi, t, \sigma, s) \equiv \\ \exists \sigma^*, s^*. \phi(s^*) \wedge start(s^*) \leq t \wedge \\ Trans^*(withPol(ll_{model}, \sigma), do(clipOnline, s), \sigma^*, s^*) \\ \wedge [\forall \sigma^{**}, s^{**}. Trans^*(\sigma^*, s^*, \sigma^{**}, s^{**}) \supset start(s^{**}) > t \\ \vee Final(\sigma^*, s^*)]$$

Again, we use *clipOnline* to switch to projection mode. The disjunct involving *Final* covers the case where the projected plan ends before time t . Using $Proj$ within test conditions, a cc-Golog plan can check whether a possible behavior would lead to certain conditions, and thus deliberate over different possible subplans. In addition, the lookahead can be limited to a certain amount of time t , which seems very useful in order to make quick decisions. A possible application of this test, illustrated by the following program, would be to check if the battery level is going to drop below 46V in the next 300 seconds if the robot comes close to the battery docking station.

$$withPol([waitFor(nearBattDockingStation), \\ if(Proj(deliverMail, 300, battLevel \leq 46), \\ chargeBatteries)], \\ deliverMail)$$

6 Implementation

Although the definition of cc-Golog requires second-order logic, it is easy to implement a PROLOG interpreter for cc-Golog, just as in the case of the original ConGolog.⁸ In order to deal with the constraints implied by the *waitFor* instruction

⁸The subtle differences between the second order axiomatization of ConGolog and a PROLOG implementation are discussed in [Giacomo et al., 2000].

during projection, we have made use of the ECRC Common Logic Programming System Eclipse 4.2 and its built-in constraint solver library `clpr` (similar to Reiter [Reiter, 1998]).

As for the on-line execution of `cc-Golog` plans, we have implemented a prototype runtime system that couples the `cc-Golog` interpreter to the `beeSoft` execution system [Burgard *et al.*, 2000]. The link between our PROLOG implementation and the `beeSoft` execution system is based on the High-Level Interface HLI [Hähnel *et al.*, 1998] which provides a uniform PROLOG interface to the low-level processes of `beeSoft`, along with a monitoring component which provides status update about the state of execution of the activated modules. In particular, our runtime-system handles high-level `send` actions, which are mapped to commands to the low-level processes, and periodically generates `ccUpdate` actions reflecting the latest status updates of the low-level processes.

7 Summary and Discussion

In summary, we have extended `cc-Golog` so that it becomes suitable for both projections of plans and their on-line execution. In doing so, we also incorporated a model of a robot architecture typically found on modern mobile robots. Finally, we showed how to interleave on-line execution and a form of time-bounded projection.

As mentioned before, the idea of modeling both projection and on-line execution was first explored in [de Giacomo and Levesque, 1999]. However, the authors consider neither time nor the idea of low-level processes which interact with a high-level controller in complex ways. Most importantly, there is no distinction between the effects on fluents during projection and on-line execution, a distinction we feel is necessary when it comes to modeling essential features of mobile robots such as their location at a given time. [Lesperance and Ng, 2000] have extended de Giacomo and Levesque's ideas in a direction which bears some resemblances to ours. They propose that during projection one also needs to consider a *simulated environment* which, for example, produces exogenous actions to inform the high-level controller that the destination is reached. However, their notion of a simulated environment remains fairly simple since they are not able to model temporally extended processes, and they also do not distinguish between the effects on fluents during projection and on-line execution. We feel that our proposal addresses some of these shortcomings and brings logic-based robot controllers yet another step closer to real world robotics.

The examples in this paper involving navigation tasks of a single robot were chosen mainly because of their simplicity and as such may not be too convincing as to why logic-based robot controllers with built-in projection mechanisms are beneficial or even necessary, especially at run-time. We believe that the advantages of projections will become much more apparent once robots engage in more complex tasks. Consider, for example, a multi-robot delivery scenario where a user makes a request to have a letter delivered by one of the robots. Then in order to determine which robot should deliver the letter, each robot might use projection to determine the cost that would arise if it were to carry out this job, and the task could then be assigned to the robot with a minimal cost estimate. On-the-fly projection becomes even more important when robots need to coordinate their activities. Suppose that two robots agree to meet somewhere at a certain time in the future. Until then they would probably want to carry out

as much of their other duties as possible. Since tasks may not be interruptible at arbitrary times, each robot would be well-advised to check how much of the current task can be completed before heading off to the meeting point. To do so, some form of projection seems necessary. We plan to investigate these and other multi-robot scenarios in the future.

References

- [Burgard *et al.*, 2000] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000.
- [de Giacomo and Levesque, 1999] G. de Giacomo and H.J. Levesque. An incremental interpreter for high-level programs with sensing. In H. Levesque and F. Pirri, editors, *Logical Foundations for Cognitive Agents*, pages 86–102. Springer, 1999.
- [Giacomo and Levesque, 1999] G. De Giacomo and H. Levesque. Projection using regression and sensors. In *Proc. IJCAI'99*, 1999.
- [Giacomo *et al.*, 2000] Giuseppe De Giacomo, Yves Lesperance, and Hector J Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169, 2000.
- [Grosskreutz and Lakemeyer, 2000a] H. Grosskreutz and G. Lakemeyer. `cc-golog`: Towards more realistic logic-based robot controllers. In *AAAI'2000*, 2000.
- [Grosskreutz and Lakemeyer, 2000b] H. Grosskreutz and G. Lakemeyer. Turning high-level plans into robot programs in uncertain domains. In *ECAI'2000*, 2000.
- [Grosskreutz, 2000] H. Grosskreutz. Probabilistic projection and belief update in the `pgolog` framework. In *Second International Cognitive Robotics Workshop*, 2000.
- [Hähnel *et al.*, 1998] D. Hähnel, W. Burgard, and G. Lakemeyer. Golex - bridging the gap between logic (`golog`) and a real robot. In *Proceedings of the 22st German Conference on Artificial Intelligence (KI 98)*, 1998.
- [Lesperance and Ng, 2000] Y. Lesperance and H.-K. Ng. Integrating planning into reactive high-level robot programs. In *Second International Cognitive Robotics Workshop*, 2000.
- [Levesque *et al.*, 1997] Hector J. Levesque, Raymond Reiter, Yves Lesperance, Fangzhen Lin, and Richard Scherl. `Golog`: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.
- [Levesque *et al.*, 1998] Hector Levesque, Fiora Pirri, and Ray Reiter. Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1998. URL: <http://www.ep.liu.se/ea/cis/1998/018/>.
- [McCarthy, 1963] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University. Reprinted 1968 in *Semantic Information Processing* (M.Minske ed.), MIT Press, 1963.
- [Pinto, 1997] Javier Pinto. Integrating discrete and continuous change in a logical framework. *Computational Intelligence*, 14(1), 1997.
- [Reiter, 1996] Ray Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proc. KR'96*, pages 2–13, 1996.
- [Reiter, 1998] R. Reiter. Sequential, temporal `golog`. In *Proc. KR'98*, 1998.
- [Simmons *et al.*, 1997] R.G. Simmons, R. Goodwin, K.Z. Haigh, S. Koenig, J. O'Sullivan, and M.M. Veloso. Xavier: Experience with a layered robot architecture. *ACM SIGART Bulletin Intelligence*, 8 (1–4), 1997.

An On-line Decision-Theoretic Golog Interpreter.

Mikhail Soutchanski

Dept. of Computer Science
University of Toronto
Toronto, ON M5S 3H5
mes@cs.toronto.edu

Abstract

We consider an on-line decision-theoretic interpreter and incremental execution of Golog programs. This new interpreter is intended to overcome some limitations of the off-line interpreter proposed in [Boutilier *et al.*, 2000]. We introduce two new search control operators that can be mentioned in Golog programs: the on-line interpreter takes advantage of one of them to save computational efforts. In addition to sensing actions designed to identify outcomes of stochastic actions, we consider a new representation for sensing actions that may return both binary and real valued data at the run time. Programmers may use sensing actions explicitly in Golog programs whenever results of sensing are required to evaluate tests. The representation for sensing actions that we introduce allows the use of regression, a computationally efficient mechanism for evaluation of tests. We describe an implementation of the on-line incremental decision-theoretic Golog interpreter in Prolog. The implementation was successfully tested on the B21 robot manufactured by RWI.

1 Introduction

The aim of this paper is to provide a new on-line architecture of designing controllers for autonomous agents. Specifically, we explore controllers for mobile robots programmed in DTGolog, an extension of the high-level programming language Golog. DTGolog aims to provide a seamless integration of a decision-theoretic planner based on Markov decision processes with an expressive set of program control structures available in Golog. The motivation for this research is provided in [Boutilier *et al.*, 2000], and we ask the reader to consult that paper for additional technical details and arguments why neither model-based planning, nor programming alone can manage the conceptual complexity of devising controllers for mobile robots. The DTGolog interpreter described in [Boutilier *et al.*, 2000] is an off-line interpreter that computes the optimal conditional policy π , the probability pr that π can be executed successfully and the expected utility u of the policy. The semantics of a DTGolog program p is defined by the predicate $BestDo(p, s, h, \pi, u, pr)$, where s is a starting situation and h is a given finite horizon. The policy π returned by the off-line interpreter is a Golog program consisting of the sequential composition of agent actions, $senseEffect(a)$ sensing

actions (which serve to identify a real outcome of a stochastic action a) and conditionals (**if** ϕ **then** π_1 **else** π_2), where ϕ is a situation calculus formula that provides a test condition to decide which branch of a policy the agent should take given the result of sensing. The interpreter looks ahead up to the last choice point in each branch of the program (say, between alternative primitive actions), makes the choice and then proceeds backwards recursively, deciding at each choice point what branch is optimal. It is assumed that once the off-line DTGolog interpreter has computed an optimal policy, the policy is given to the robotics software to control a real mobile robot.

However this off-line architecture has a number of limitations. Imagine that we are interested in executing a program $(p_1; p_2)$ where both sub-programs p_1 and p_2 are very large nondeterministic DTGolog programs designed to solve ‘independent’ decision-theoretic problems: p_2 is supposed to start in a certain set of states, but from the perspective of p_2 it is not important what policy will be used to reach one of those states. Intuitively, in this case we are interested in computing first an optimal policy π_1 (that corresponds to p_1), executing π_1 in the real world and then computing and executing an optimal policy π_2 . But the off-line interpreter can return only the optimal policy π that corresponds to the whole program $(p_1; p_2)$, spending on this computation more time than necessary: to compute and execute π_2 it is not relevant what decisions have to be made during the execution of π_1 . The second limitation becomes evident if in addition to $senseEffect$ sensing actions serving to identify outcomes of stochastic actions, we need to explicitly include sensing actions in the program (and those sensing actions cannot be characterized as stochastic actions with a fixed finite set of outcomes). Imagine that we are given a program

$p_1; (\pi v, t).[(now(t))?; sense(Q, v, t); \mathbf{if} \phi \mathbf{then} p_2 \mathbf{else} p_3]$, where $sense(Q, v, t)$ is a sensing action that returns at time t a measurement v of a quantity Q (e.g., the current coordinates, the battery voltage) and the condition ϕ depends on the real data v that will be returned by sensors (in the program above p_1, p_2, p_3 are sub-programs, the choice operator π binds variables v and t and the test $now(t)?$ grounds the current time). Intuitively, we want to compute an optimal policy π_1 (corresponding to p_1) off-line, execute π_1 in the real world, sense v and then compute and execute an optimal policy π_2 that corresponds to the conditional Golog program in brackets. But the off-line interpreter is not able to compute an optimal policy if a given program includes explicit sensing actions and no information is available about the possible results of sensing. Note that the nondeterministic choice operator π (it occurs in

front of the program) should not be confused with policies π_1 and π_2 computed by an interpreter.

We propose to compute and execute optimal policies on-line using a new incremental decision theoretic interpreter. It works in a step-by-step mode. Given a Golog program p and a starting situation s , it computes an optimal policy π and the program p' that remains when a first action a in π will be executed. At the next stage, the action a is executed in the real world. The interpreter gets sensory information to identify which outcome of a has actually occurred if a is a stochastic action: this may require doing a sequence of sensing actions on-line. The action a (and possibly sensing actions performed after that) results in a new situation. Then the cycle of computing an optimal policy and remaining program, executing the first action and getting sensory information (if necessary) repeats until the program terminates or execution fails. In the context of the incremental on-line execution, we define a new programming operator $optimize(p)$ that limits the scope of search performed by the interpreter. If p is the whole program, then no computational efforts are saved when the interpreter computes an optimal policy from p , but if the programmer writes $optimize(p_1); p_2$, then the on-line incremental interpreter will compute and execute step-by-step the Golog program p_1 without looking ahead to decisions that need to be made in p_2 . If the programmer knows that the sensing action $sense(Q, v, t)$ is necessary to evaluate the condition ϕ , then using the program

$$optimize(p_1); (\pi t, v).[(now(t))?; optimize(sense(Q, v, t)); \\ \text{if } \phi \text{ then } p_2 \text{ else } p_3]$$

the required information about Q will be obtained before the incremental interpreter will proceed to the execution of the conditional. If the sensing action $sense(Q, v, t)$ has many different outcomes, then this approach gives computational advantages over the off-line approach to computing an optimal policy.

Thus, the incremental interpretation of the decision-theoretic Golog programs needs an account of sensing (formulated in the situation calculus) that will satisfy several criteria which naturally arise in the robotics context. There are several accounts of sensing in the situation calculus that address different aspects of reasoning about sensory and physical actions [Bacchus *et al.*, 1995; De Giacomo and Levesque, 1999a; 1999b; Funge, 1998; Grosskreutz, 2000; Lakemeyer, 1999; Levesque, 1996; Pirri and Finzi, 1999; Scherl and Levesque, 1993].

Below we propose a new representation of sensing that simplifies reasoning about results of sensing, does not require consistency of sensory information with the domain theory, leads to a natural and sound implementation and has connections with representation of knowledge and sensing considered in [Reiter, 2000].

In Section 2 we recall the representation of the decision theoretic domain introduced in [Boutilier *et al.*, 2000]. In Section 3 we propose a representation for sensing actions and consider several examples. In section 4 we consider the on-line incremental decision-theoretic interpreter. Section 5 discusses connections with previously published papers.

2 The decision theoretic problem representation

The paper [Boutilier *et al.*, 2000] introduces the representation of problem domains that do not include sensing actions. The representation is based on the distinction between agent actions (which can be either deterministic or stochastic) and

nature's actions which correspond to separate outcomes of a stochastic agent action. Nature's actions are considered deterministic. They cannot be executed by the agent itself, therefore they never occur in policies which the agent executes. When the agent does a stochastic action a in a situation s , nature chooses one of the outcomes n of that action and the situation $do(n, s)$ is considered as one of resulting situations. In accordance with this perspective, the evolution of a stochastic transition system is specified by precondition and successor state axioms which never mention stochastic agent actions, but mention only deterministic agent actions and nature's actions. In [Boutilier *et al.*, 2000], it is suggested to characterize the DTGolog problem domain by: 1) the predicate $agentAction(a)$ which holds if a is an agent action, 2) the predicate $stochastic(a, s, n)$ meaning that nature's action n is one of outcomes of the stochastic agent action a in the situation s , 3) the function symbol $prob(n, s)$ that denotes the probability of nature's action n in situation s , and 4) the predicate $senseCond(n, \phi)$ specifying the test condition ϕ serving to identify the outcome n of the last stochastic action, 5) the function symbol $reward(do(a, s))$ denotes rewards and costs as functions of the current situation $do(a, s)$, the action a or both.

As an example, imagine a robot moving between different locations: the process of going is initiated by a deterministic action $startGo(l_1, l_2, t)$ but is terminated by a stochastic action $endGo(l_1, l_2, t)$ that may have two different outcomes: successful arrival $endGoS(l_1, l_2, t)$ and unsuccessful stop in a place different from the destination $endGoF(l_1, l_3, t)$ (the robot gets stuck in the hall or cannot enter an office because its door is closed). We represent the process of moving between locations l_1 and l_2 by the relational fluent $going(l_1, l_2, s)$ and represent a (symbolic) location of the robot by the relational fluent $robotLoc(l, s)$ meaning that l (the office of an employee, the hall or the main office) is the place where the robot is. The transitions in the stochastic dynamical system describing the robot's motion are characterized by the following precondition and successor state axioms.

$$Poss(startGo(l_1, l_2, t), s) \equiv \neg(\exists l, l')going(l, l', s) \wedge robotLoc(l_1, s),$$

$$Poss(endGoS(l_1, l_2, t), s) \equiv going(l_1, l_2, s), \\ Poss(endGoF(l_1, l_2, t), s) \equiv \exists l'.going(l_1, l', s) \wedge l' \neq l_2,$$

$$going(l, l', do(a, s)) \equiv (\exists t)a = startGo(l, l', t) \vee \\ going(l, l', s) \wedge \neg(\exists t)a = endGoS(l, l', t) \vee \\ going(l, l', s) \wedge \neg(\exists t, l'')a = endGoF(l, l'', t).$$

The real outcome n of a stochastic agent action a can be identified only from sensory information. This information has to be obtained by executing sensing actions. In the following section we propose a new representation for sensing actions providing a seamless integration with the representation considered in this section.

3 Sensing actions

In contrast to physical actions, sensing actions do not change any properties of the external world, but return values measured by sensors. Despite this difference, it is convenient to treat both physical and sensing actions equally in successor state axioms. This approach is justifiable if fluents represent what the robot 'knows' about the world (see Figure 1). More specifically, let the high-level control module of the robot be provided with the internal logical model of the world (the set of precondition and successor state axioms) and the

axiomatization of the initial situation. The programmer expresses in this axiomatization his (incomplete) knowledge of the initial situation and captures certain important effects of the robot's actions, but some other effects and actions of other agents may remain unmodeled. When the robot does an action in the real world (i.e., the high-level control module sends a command to effectors and effectors execute this command), it does not know directly and immediately what effects in reality this action will produce: the high-level control module may only compute expected effects using the internal logical model. Similarly, if the robot is informed about actions executed by external agents, the high-level control module may compute expected effects of those exogenous actions (if axioms account for them). Thus, from the point of view of the programmer who designed the axioms, the high-level control module maintains the set of beliefs that the robot has about the real world. This set of beliefs needs feedback from the real world because of possible contingencies, incompleteness of the initial information and unobserved or unmodeled actions executed by other agents. To gain the feedback, the high-level control module requests information from sensors and they return required data. We find it convenient to represent information gathered from sensors by an argument of the sensing action: a value of the argument will be instantiated at the run time when the sensing action will be executed. Then, all the high-level control module needs to know is the current situation (action log): expected effects of actions on fluents can be computed from the given sequence of physical and sensing actions.

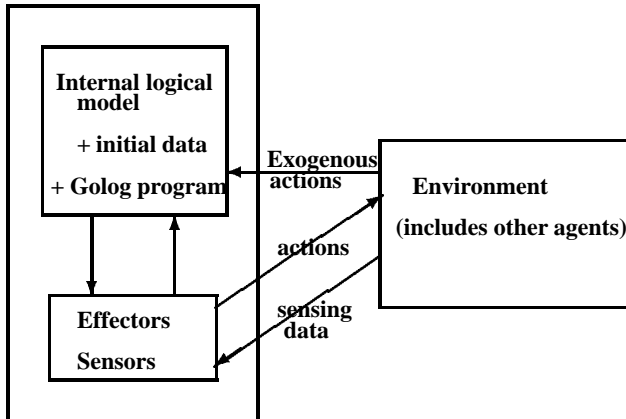


Figure 1: A high-level control module and low-level modules interacting with the environment.

In the sequel, we consider only deterministic sensing actions, but noisy sensing actions can be represented as well.

We suggest representing sensing actions by the functional symbol $sense(q, v, t)$ where q is what to sense, v is term representing a run time value returned by the sensors and t is time; the predicate $senseAction(a)$ is true whenever a is a sensing action. We proceed to consider several examples of successor state axioms that employ our representation.

1. Let $sense(Coord, l, t)$ be the sensing action that returns the pair $l = (x, y)$ of geometric coordinates of the current robot location on the two-dimensional grid and $gridLoc(x, y, s)$ be a relational fluent that is true if (x, y) are the coordinates of the robot's location in s . In this example we assume that all actions are deterministic (we do this only to simplify the exposition of this example). The process of moving (represented by the relational fluent

$moving(x_1, y_1, x_2, y_2, s)$) from the grid location (x_1, y_1) to the point (x_2, y_2) is initiated by the deterministic instantaneous action $startMove(x_1, y_1, x_2, y_2, t)$ and is terminated by the deterministic action $endMove(x_1, y_1, x_2, y_2, t)$. The robot may also change its location if it is transported by an external agent from one place to another: the exogenous actions $take(x_1, y_1, t)$ and $put(x_2, y_2, t)$ account for this (and the fluent $transported(s)$ represents the process of moving the robot by an external agent). The following successor state and precondition axioms characterize aforementioned fluents and actions.

$$\begin{aligned} gridLoc(x, y, do(a, s)) \equiv & \\ ((\exists t, l) a = sense(Coord, l, t) \wedge xCrd(l) = x \wedge yCrd(l) = y) \vee & \\ (\neg transported(s) \wedge (\exists t, x', y') a = endMove(x', y', x, y, t)) \vee & \\ (transported(s) \wedge (\exists t) a = put(x, y, t)) \vee & \\ gridLoc(x, y, s) \wedge \neg(\exists t, x', y') a = put(x', y', t) \wedge & \\ \neg(\exists l, t) a = sense(Coord, l, t) \wedge & \\ \neg(\exists x, y, x', y', t) a = endMove(x, y, x', y', t), & \end{aligned}$$

where $xCrd(l)$, $yCrd(l)$ denote, respectively, x and y components of the current robot location l .

$$\begin{aligned} moving(x_1, y_1, x_2, y_2, do(a, s)) \equiv & \\ (\exists t) a = startMove(x_1, y_1, x_2, y_2, t) \vee & \\ moving(x_1, y_1, x_2, y_2, s) \wedge & \\ \neg(\exists t) a = endMove(x_1, y_1, x_2, y_2, t). & \end{aligned}$$

$$\begin{aligned} transported(do(a, s)) \equiv & (\exists t, x, y) a = take(x, y, t) \vee \\ & transported(s) \wedge \neg(\exists t, x', y') a = put(x', y', t) \vee \\ & (\neg \exists x, y, x', y') moving(x, y, x', y') \wedge gridLoc(x, y, s) \wedge \\ & (\exists l, t) a = sense(Coord, l, t) \wedge (xCrd(l) \neq x \vee yCrd(l) \neq y). \end{aligned}$$

$$\begin{aligned} Poss(startMove(x_1, y_1, x_2, y_2, t), s) \equiv & \neg transported(s) \wedge \\ & \neg(\exists x, y, x', y') moving(x, y, x', y') \wedge gridLoc(x_1, y_1, s), \\ Poss(endMove(x_1, y_1, x_2, y_2, t), s) \equiv & moving(x_1, y_1, x_2, y_2, s), \\ Poss(take(x, y, t), s) \equiv & \neg transported(s) \wedge gridLoc(x, y, s) \wedge \\ & \neg(\exists x_1, y_1, x_2, y_2) moving(x_1, y_1, x_2, y_2, s), \\ Poss(put(x, y, t), s) \equiv & transported(s). \end{aligned}$$

Imagine that in the initial situation the robot stays at $(0,0)$; later, at time 1, it starts moving from $(0,0)$ to $(2,3)$, but when the robot stops at time 11 and senses its coordinates at time 12, its sensors tell it that it is located at $(4,4)$. The sensory information is inconsistent with the expected location of the robot, but this discrepancy can be attributed to unobserved actions of an external agent who transported the robot. Hence, the final situation is not

$$S_3 = do(sense(Coord, (4, 4), 12), do(endMove(0, 0, 2, 3, 11), do(startMove(0, 0, 2, 3, 1), S_0))),$$

as we originally expected, but the situation resulting from the execution of exogenous actions $take(2, 3, T_1)$ and $put(4, 4, T_2)$, in the situation when the robot ends moving, followed by the sensing action. The exogenous actions occurred at unknown times T_1, T_2 (we may say about the actual history only that $11 \leq T_1 < T_2 \leq 12$).¹

2. The robot can determine its location using data from sonar and laser sensors. But if the last action was not sensing coordinates (x, y) , then the current location can be determined

¹In the sequel, we do not consider how the discrepancy between results of a deterministic action and observations obtained from sensors can be explained by occurrences of unobserved exogenous actions. However, our example indicates that inconsistency between physical and sensory actions can be resolved by solving a corresponding diagnostic problem (e.g., see [McIlraith, 1998]).

from the previous location and the actions that the robot has executed. The process of going from l to l' is initiated by the deterministic action $startGo(l, l', t)$ and is terminated by the stochastic action $endGo(l, l', t)$ (axiomatized in Section 2).

$$\begin{aligned} robotLoc(l, do(a, s)) &\equiv (\exists t, v, x, y) a = sense(Coord, v, t) \\ &\wedge xCrd(v) = x \wedge yCrd(v) = y \wedge \\ &((\exists p) l = office(p) \wedge inOffice(l, x, y) \vee \\ & l = Hall \wedge \neg(\exists p) inOffice(office(p), x, y)) \vee \\ &(\exists t, l_1) a = endGoS(l_1, l, t) \vee (\exists t, l_1) a = endGoF(l_1, l, t) \vee \\ &robotLoc(l, s) \wedge \neg(\exists t, l_2, l_3, v) (a = sense(Coord, v, t) \wedge \\ & a = endGoS(l_2, l_3, t) \wedge a = endGoF(l_2, l_3, t)). \end{aligned}$$

where the predicate $inOffice(l, x, y)$ is true if the pair (x, y) is inside of the office l , the functional symbols $bottomY(l)$, $topY(l)$, $rightX(l)$ and $leftX(l)$ represent coordinates of top left and bottom right corners of a rectangle that contains the office l inside: When the robot stops and senses coordinates, it determines its real location and the high-level control module can identify the outcome of the stochastic action $endGo(l, l', t)$: whether the robot stopped successfully or failed to arrive at the intended destination.

3. Let $give(item, person, t)$ be a stochastic action that has two different outcomes: $giveS(item, person, t)$ – the robot gives successfully an *item* to *person* at time t – and $giveF(item, person, t)$ – the action of giving an *item* to *person* is unsuccessful. Let $sense(Ackn, v, t)$ be the action of sensing whether delivery of the *item* to *person* was successful or not: if it was, then delivery is acknowledged and $v=1$, if not – the result of sensing is $v=0$. The following successor state axiom characterizes how the fluent $hasCoffee(person, s)$ changes from situation to situation: whenever the robot is in the office of *person* and it senses that one of its buttons was pressed, it is assumed that the *person* pressed a button to acknowledge that she has a coffee. From this sensory information, the high-level control module can identify whether the outcome of $give(item, person, t)$ was successful or not.

$$\begin{aligned} hasCoffee(pers, do(a, s)) &\equiv \\ (\exists t) a = giveS(Coffee, pers, t) \vee hasCoffee(pers, s) \vee \\ (\exists t, v) a = sense(Ackn, v, t) \wedge v = 1 \wedge \\ robotLoc(office(pers), s). \end{aligned}$$

It is surprisingly straightforward to use regression in our setting to solve the forward projection task. Let \mathcal{D} be a background axiomatization of the domain (a set of successor state axioms, precondition axioms, unique name axioms and a set of first order sentences whose only situation term is S_0) and let $\phi(s)$ be a situation calculus formula that has the free variable s as the only situational argument. Suppose we are given a ground situational term S that may mention both physical and sensing actions. The forward projection task is to determine whether

$$\mathcal{D} \models \phi(S)$$

Regression is a computationally efficient way of solving the forward projection task [Reiter, 2000; Pirri and Reiter, 1999] when S does not mention any sensing actions. The representation introduced above allows us to use regression also in the case when S mentions sensing actions explicitly. Thus, we can use regression to evaluate tests (ϕ) ? in Golog programs with sensing actions: no modifications are required. In addition, our approach allows us to use an implementation in Prolog considered in [Reiter, 2000]. There is also an interesting connection between our representation of ‘beliefs’ and sensing and the approach to knowledge-based programming [Reiter, 2000]. In [Soutchanski, 2001] we show that his approach

and our approach to the solution of the forward projection task (with sensing actions) are equivalent.

4 The incremental on-line DTGolog interpreter

The incremental DTGolog interpreter uses the predicate $IncrBestDo(p_1, s, p_2, h, \pi, u, pr)$ and the predicate $Final(p, s, \pi, u)$. The former predicate takes as input the Golog program p_1 , starting situation s , horizon h and returns the optimal conditional policy π , its expected utility u , the probability of success pr , and the program p_2 that remains after executing the first action from the policy π . The latter predicate tells when the execution of the policy completes: $Final(p, s, \pi, u)$ is true if either the program p is *Nil* (the null program) or *Stop* (a zero cost action that takes the agent to an absorbing state meaning that the execution failed), or if the policy π is *Nil* or *Stop*. In all these cases, u is simply the reward associated with the situation s . Note that in comparison to *BestDo*, *IncrBestDo* has an additional argument p_2 representing the program that remains to be executed.

$IncrBestDo(p_1, s, p_2, h, \pi, u, pr)$ is defined inductively on the structure of a Golog program p_1 . Below we consider its definition in the case when the program p_1 begins with a deterministic agent action.

$$\begin{aligned} IncrBestDo(a; p_1, s, p_2, h, \pi, u, pr) &\stackrel{def}{=} \\ [\neg Poss(a, s) \wedge \\ p_2 = Nil \wedge \pi = Stop \wedge pr = 0 \wedge u = reward(s) \vee \\ Poss(a, s) \wedge p_2 = p_1 \wedge \exists(p', \pi', u', pr') \\ IncrBestDo(p_2, do(a, s), p', h-1, \pi', u', pr') \wedge \\ \pi = (a; \pi') \wedge u = reward(s) + u' \wedge pr = pr']. \end{aligned}$$

If a deterministic agent action a is possible in situation s , then we compute the optimal policy π' of the remaining program p_1 , its expected utility u' and the probability of successful termination pr' . Because a is a deterministic action, the probability pr' that the policy π' will complete successfully is the same as for the program itself; the expected utility u is a sum of a reward for being in s and the expected utility of continuation u' . If a is not possible, then the remaining program is *Nil*, and the policy is the *Stop* action, the expected utility is the reward in s and the probability of success is 0.

Other cases are defined similarly to *BestDo*, e.g., if the program begins with the finite nondeterministic choice $(\pi(x : \tau)p); p'$, where τ is the finite set $\{c_1, \dots, c_n\}$ and the choice binds all free occurrences of x in p to one of the elements:

$$\begin{aligned} IncrBestDo((\pi(x : \tau)p); p', s, p_r, h, pol, u, pr) &\stackrel{def}{=} \\ IncrBestDo((p|_{c_1}^x \mid \dots \mid p|_{c_n}^x); p', s, p_r, h, pol, u, pr) \end{aligned}$$

where $p|_c^x$ means substitution of c for all free occurrences of x in p . Thus, the optimal policy pol corresponds to the element c in τ that delivers the best execution. Note that the remaining program p_r is the same on the both sides of the definition.

Recall that policies are Golog programs as well. Moreover, if p is a Golog program that contains many nondeterministic choices, the optimal policy π computed from p is a conditional program that does not involve any nondeterministic choices. This observation suggests that programmers may wish to take advantage of the structure in a decision theoretic problem and use explicit search control operators that limit bounds where the search for an optimal policy has to concentrate. In addition to the standard set of Golog programming constructs,

we introduce two new operators: $local(p)$ and $optimize(p)$. Intuitively, the program $local(p_1); p_2$ means the following. First, compute the optimal policy π_1 corresponding to the sub-program p_1 , then compute the optimal policy π corresponding to the program $\pi_1; p_2$. If both sub-programs p_1 and p_2 are highly nondeterministic, then using the operator $local(p_1)$ the programmer indicates where the computational efforts can be saved: there is no need in looking ahead further than p_1 to compute π_1 . Thus, in the case that a Golog program begins with $local(p)$

$$IncrBestDo(local(p_1); p_2, s, p_r, h, \pi, u, pr) \stackrel{def}{=} (\exists p, \pi_1, u_1, pr_1) IncrBestDo(p_1; Nil, s, p, h, \pi_1, u_1, pr_1) \wedge IncrBestDo(\pi_1; p_2, s, p_r, h, \pi, u, pr).$$

The construct $local(p_1)$ limits the search, but once the policy π_1 was computed and the first action in π_1 was executed, the remaining part of the program has no indication where the search may concentrate. For this reason, the programmer may find convenient to use another search control operator that once used persists in the remaining part of the program until the program inside the scopes of that operator terminates. This operator is called $optimize(p)$ and is specified by the following abbreviation.

$$IncrBestDo(optimize(p_1); p_2, s, p_r, h, \pi, u, pr) \stackrel{def}{=} (\exists p') IncrBestDo(p_1; Nil, s, p', h, \pi, u, pr) \wedge (p' \neq Nil \wedge p_r = (optimize(p'); p_2) \vee p' = Nil \wedge p_r = p_2).$$

According to this specification, an optimal policy π can be computed without looking ahead to the program p_2 ; hence, using $optimize(p_1)$ a programmer can express a domain specific procedural knowledge to save computational efforts. Note that when p' is Nil , i.e. there will be nothing in p_1 to execute after doing the only action in π , the remaining program p_r contains only p_2 .

4.1 Implementing the on-line interpreter

Given the definitions of $IncrBestDo$ mentioned in the previous sub-section, we can consider now the on-line interpretation coupled with execution of Golog programs. The definitions of $IncrBestDo(p_1, s, p_2, h, \pi, u, pr)$ translate directly into Prolog clauses (we omit them here). The on-line interpreter calls the off-line $incrBestDo(E, S, ER, H, Poll, U1, Probl)$ interpreter to compute an optimal policy from the given program expression E , gets the first action of the optimal policy, commits to it, does it in the physical world, then repeats with the rest of the program. The following is such an interpreter implemented in Prolog:

```
online(E, S, H, Pol, U) :-
incrBestDo(E, S, ER, H, Poll, U1, Probl),
( final(ER, S, H, Poll, U1), Pol=Poll, U=U1 ;
reward(R, S),
Poll = (A : Rest),
( agentAction(A), not stochastic(A, S, L),
doReally(A), /*execute A in reality*/
!, /* commit to the result */
online(ER, do(A, S), H, PolFut, UFut),
Pol=(A : PolFut), U is R + UFut ;
senseAction(A),
doReally(A), /* do sensing */
!, /*commit to results of sensing*/
online(ER, do(A, S), H, PolFut, UFut),
Pol=(A : PolFut), U is R + UFut ;
agentAction(A), stochastic(A, S, L),
doReally(A), /*execute A in reality*/
!, /* commit to the result */
```

```
senseEffect(A, S, SEff),
diagnose(SEff, L, SN), /*What happened?*/
online(ER, SN, H, PolFut, UFut),
Pol=(A : PolF), U is R + UFut
).
```

The on-line interpreter uses the Prolog cut (!) to prevent backtracking to the predicate $doReally$: we need this because once actions have been actually performed in the physical world, the robot cannot undo them.

In addition to predicates mentioned in section 2, the on-line interpreter uses the predicate $senseEffect(A, S1, S2)$, the predicate $diffSequence(A, Seq)$ and the predicate $diagnose(S2, OutcomesList, S3)$. We describe below their meaning and show their implementation in Prolog.

Given the stochastic action A and the situation $S1$, the predicate $senseEffect(A, S1, S2)$ holds if $S2$ is the situation that results from doing a number of sensing actions necessary to differentiate between outcomes of the stochastic action A . The predicate $diffSequence(A, Seq)$ holds if Seq is the sequence of sensing actions ($a_1; a_2; \dots; a_n$) specified by the programmer in the domain problem representation: this sequence is differentiating if after doing all actions in the sequence the action chosen by ‘nature’ as the outcome of stochastic action A can be uniquely identified.

```
senseEffect(A, S, SE) :- diffSequence(A, Seq),
getSensorInput(S, Seq, SE).

getSensorInput(S, A, do(A, S)) :- senseAction(A),
doReally(A). /*connect to sensors, get data
for a free variable in A */

getSensorInput(S1, (A : Rest), SE) :-
doReally(A), /* connect to sensors,
get data */
getSensorInput(do(A, S1), Rest, SE).
```

The predicate $diagnose(S1, L, S2)$ takes as its first argument the situation resulting from getting a sensory input: it contains ‘enough’ information to disambiguate between different possible outcomes of the last stochastic action A . The second argument is the list L of all outcomes that nature may choose if the agent executes the stochastic action A in $S1$ and the third argument is the situation that is the result of nature’s action which actually occurred. We can identify which action nature has chosen using the set of mutually exclusive test conditions $senseCond(n_i, \phi_i)$, where ϕ_i is a term representing a situation calculus formula: if ϕ_i holds in the current situation, then we know that nature has chosen the action n_i (n_i belongs to the list L).

```
diagnose(SE, [N], do(N, SE)) :-
senseCond(N, C), holds(C, SE).

diagnose(SE, [N|Outcomes], SN) :- senseCond(N, C),
( holds(C, SE), SN=do(N, SE) ;
not holds(C, SE),
diagnose(SE, Outcomes, SN) ).
```

Successful tests of the implementation described here were conducted in a real office environment on a mobile robot B21 manufactured by RWI. The low-level software was initially developed in the University of Bonn to control Rhino, another B21 robot, see [Burgard *et al.*, 1999] for details. The tests of implementation demonstrated that using the expressive set of Golog operators it is straightforward to encode domain knowledge as constraints on the given large MDP problem. The operator $optimize(p)$ proved to be useful in providing heuris-

tics which allowed to compute sub-policies in real time. These preliminary tests have brought several new important issues, e.g., how the computation of a new policy off-line can proceed in parallel with executing actions from the policy on-line.

5 Discussion

The incremental Golog interpreter based on the single-step *Trans*-semantics is introduced in [De Giacomo and Levesque, 1999b]. The Golog programs considered there may include binary sensing actions. The interpreter considered in our paper is motivated by similar intuitions, but it is based on a different decision-theoretic semantics and employs more expressive representation of sensing. The paper [De Giacomo and Levesque, 1999a] introduces guarded sensed fluent axioms (GSFA) and guarded successor state axioms (GSSA) and assumes that there is a stream of sensor readings available at any time. These readings are represented by unary sensing functions (syntactically, they look like functional fluents). Because we introduce the representation for sensing actions, they can be mentioned explicitly in Golog programs or can be executed by the interpreter. The major advantage of our representation is that it does not require consistency of sensory readings with the action theory (this may prove useful in solving diagnostic tasks: [McIlraith, 1998]). The execution monitoring framework proposed in [De Giacomo *et al.*, 1998] assumes that the feedback from the environment is provided in terms of actions executed by other agents. Because in this paper we assume that the feedback is provided in terms of sensory readings, this may lead to development of a more realistic framework. An approach to integrating planning and execution in stochastic domains [Dearden and Boutilier, 1994] is an alternative to the approach proposed here.

6 Concluding remarks

Several important issues are not covered in this paper. One of them is monitoring and rescheduling of policies. Note that all actions in policies have time arguments which will be instantiated by moments of time: when the incremental interpreter computes an optimal policy, it also determines a schedule when actions have to be executed. But in realistic scenarios, when the robot is involved in ongoing processes extended over time, it may happen that a process will terminate earlier or later than it was expected.

The diagnostic task that the current version of the on-line interpreter solves is admittedly oversimplified. We expect that additional research integrating the on-line incremental interpreter with the approach proposed in [McIlraith, 1998] will allow us to formulate a more comprehensive version.

7 Acknowledgments

Thanks to Ray Reiter, Maurice Pagnucco, and anonymous reviewers for comments on preliminary versions of this paper. Sam Kaufman provided help with conducting tests on the mobile robot.

References

[Bacchus *et al.*, 1995] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. Reasoning about noisy sensors in the situation calculus. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1933–1940, Montreal, 1995.

- [Boutilier *et al.*, 2000] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level robot programming in the situation calculus. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI'00)*, Austin, Texas, 2000.
- [Burgard *et al.*, 1999] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 1999.
- [De Giacomo and Levesque, 1999a] G. De Giacomo and H. Levesque. Projection using regression and sensors. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.
- [De Giacomo and Levesque, 1999b] G. De Giacomo and H.J. Levesque. An incremental interpreter for high-level programs with sensing. In Levesque and Pirri, editors, *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*, pages 86–102. Springer, 1999.
- [De Giacomo *et al.*, 1998] G. De Giacomo, R. Reiter, and M.E. Soutchanski. Execution monitoring of high-level robot programs. In *Principles of Knowledge Representation and Reasoning: Proc. of the 6th International Conference (KR'98)*, pages 453–464, Trento, Italy, 1998.
- [Dearden and Boutilier, 1994] Richard Dearden and Craig Boutilier. Integrating planning and execution in stochastic domains. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 162–169, 1994.
- [Funge, 1998] J. Funge. *Making Them Behave: Cognitive Models for Computer Animation*, Ph.D. Thesis. Dept. of Computer Science, Univ. of Toronto, 1998.
- [Grosskreutz, 2000] H. Grosskreutz. Probabilistic projection and belief update in the pgolog framework. In *The 2nd International Cognitive Robotics Workshop, 14th European Conference on AI*, pages 34–41, Berlin, Germany, 2000.
- [Lakemeyer, 1999] G. Lakemeyer. On sensing and off-line interpreting in golog. In Levesque and Pirri, editors, *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*, pages 173–189. Springer, 1999.
- [Levesque, 1996] H.J. Levesque. What is planning in the presence of sensing? In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, volume 2, pages 1139–1145, Portland, Oregon, 1996.
- [McIlraith, 1998] S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Principles of Knowledge Representation and Reasoning: Proc. of the 6th International Conference (KR'98)*, pages 167–177, Italy, 1998.
- [Pirri and Finzi, 1999] F. Pirri and A. Finzi. An approach to perception in theory of actions: part 1. *Linköping Electronic Articles in Computer and Information Science*, 4(41), 1999.
- [Pirri and Reiter, 1999] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):261–325, 1999.
- [Reiter, 2000] R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. <http://www.cs.toronto.edu/~cogrobo/>, 2000.
- [Scherl and Levesque, 1993] R. Scherl and H.J. Levesque. The frame problem and knowledge producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 689–695, Washington, DC, 1993.
- [Soutchanski, 2001] M. Soutchanski. A correspondence between two different solutions to the projection task with sensing. In *The 5th Symposium on Logical Formalizations of Commonsense Reasoning*, New York, USA, 2001.

KNOWLEDGE REPRESENTATION AND REASONING

STRUCTURE-BASED CAUSALITY

Causes and Explanations: A Structural-Model Approach— Part II: Explanations

Joseph Y. Halpern*

Cornell University

Dept. of Computer Science

Ithaca, NY 14853

halpern@cs.cornell.edu

www.cs.cornell.edu/home/halpern

Judea Pearl†

Dept. of Computer Science

University of California, Los Angeles

Los Angeles, CA 90095

judea@cs.ucla.edu

www.cs.ucla.edu/~judea

Abstract

We propose a new definition of (*causal*) *explanation*, using *structural equations* to model counterfactuals. The definition is based on the notion of *actual cause*, as defined and motivated in a companion paper. Essentially, an explanation is a fact that is not known for certain but, if found to be true, would constitute an actual cause of the fact to be explained, regardless of the agent's initial uncertainty. We show that the definition handles well a number of problematic examples from the literature.

1 Introduction

The automatic generation of adequate explanations is a task essential in planning, diagnosis and natural language processing. A system doing inference must be able to explain its findings and recommendations to evoke a user's confidence. However, getting a good definition of explanation is a notoriously difficult problem, which has been studied for years. (See [Chajewska and Halpern, 1997; Gärdenfors, 1988; Hempel, 1965; Pearl, 1988; Salmon, 1989] and the references therein for an introduction to and discussion of the issues.)

In [Halpern and Pearl, 2001], we give a definition of actual causality using structural equations. Here we show how the ideas behind that definition can be used to give an elegant definition of (*causal*) explanation that deals well with many of the problematic examples discussed in the literature. The basic idea is that an explanation is a fact that is not known for certain but, if found to be true, would constitute an actual cause of the *explanandum* (the fact to be explained), regardless of the agent's initial uncertainty.

2 Causal Models: A Review

To make this paper self-contained, this section repeats material from [Halpern and Pearl, 2001]; we review the basic definitions of causal models, as defined in terms of structural equations, and the syntax and semantics of a language for

*Supported in part by NSF under grants IRI-96-25901 and IIS-0090145.

†Supported in part by grants from NSF, ONR, AFOSR, and MICRO.

reasoning about causality and explanations. See [Galles and Pearl, 1997; Halpern, 2000; Pearl, 2000] for more details, motivation, and intuition.

Causal Models: The basic picture is that the world is described by random variables, some of which may have a causal influence on others. This influence is modeled by a set of *structural equations*. Each equation represents a distinct mechanism (or law) in the world, which may be modified (by external actions) without altering the others. In practice, it seems useful to split the random variables into two sets, the *exogenous* variables, whose values are determined by factors outside the model, and the *endogenous* variables, whose values are determined by the endogenous variables. It is these endogenous variables whose values are described by the structural equations.

More formally, a *signature* \mathcal{S} is a tuple $(\mathcal{U}, \mathcal{V}, \mathcal{R})$, where \mathcal{U} is a finite set of exogenous variables, \mathcal{V} is a finite set of endogenous variables, and \mathcal{R} associates with every variable $Y \in \mathcal{U} \cup \mathcal{V}$ a nonempty set $\mathcal{R}(Y)$ of possible values for Y . A *causal* (or *structural*) *model* over signature \mathcal{S} is a tuple $M = (\mathcal{S}, \mathcal{F})$, where \mathcal{F} associates with each variable $X \in \mathcal{V}$ a function denoted F_X such that $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U)) \times (\times_{Y \in \mathcal{V} - \{X\}} \mathcal{R}(Y)) \rightarrow \mathcal{R}(X)$. F_X tells us the value of X given the values of all the other variables in $\mathcal{U} \cup \mathcal{V}$.

Example 2.1: Suppose that we want to reason about a forest fire that could be caused by either lightning or a match lit by an arsonist. Then the causal model would have the following endogenous variables (and perhaps others):

- F for fire ($F = 1$ if there is one, $F = 0$ otherwise)
- L for lightning ($L = 1$ if lightning occurred, $L = 0$ otherwise)
- ML for match lit ($ML = 1$ if the match was lit and $ML = 0$ otherwise).

The set \mathcal{U} of exogenous variables includes conditions that suffice to make all relationships deterministic (such as whether the wood is dry, there is enough oxygen in the air, etc.). Suppose that \vec{u} is a setting of the exogenous variables that makes a forest fire possible (i.e., the wood is sufficiently dry, there is oxygen in the air, and so on). Then, for example, $F_F(\vec{u}, L, ML)$ is such that $F = 1$ if either $L = 1$ or $ML = 1$.

■

Given a causal model $M = (\mathcal{S}, \mathcal{F})$, a (possibly empty) vector \vec{X} of variables in \mathcal{V} , and vectors \vec{x} and \vec{u} of values for the variables in \vec{X} and \mathcal{U} , respectively, we can define a new causal model denoted $M_{\vec{X} \leftarrow \vec{x}}$ over the signature $\mathcal{S}_{\vec{X}} = (\mathcal{U}, \mathcal{V} - \vec{X}, \mathcal{R}|_{\mathcal{V} - \vec{X}})$. Intuitively, this is the causal model that results when the variables in \vec{X} are set to \vec{x} by external action, the cause of which is not modeled explicitly. Formally, $M_{\vec{X} \leftarrow \vec{x}} = (\mathcal{S}_{\vec{X}}, \mathcal{F}^{\vec{X} \leftarrow \vec{x}})$, where $F_Y^{\vec{X} \leftarrow \vec{x}}$ is obtained from F_Y by setting the values of the variables in \vec{X} to \vec{x} .

We can describe (some salient features of) a causal model M using a *causal network*. This is a graph with nodes corresponding to the random variables in \mathcal{V} and an edge from a node labeled X to one labeled Y if F_Y depends on the value of X . Intuitively, variables can have a causal effect only on their descendants in the causal network; if Y is not a descendant of X , then a change in the value of X has no effect on the value of Y .

We restrict attention to what are called *recursive* (or *acyclic*) equations; these are ones that can be described with a causal network that is a dag. It should be clear that if M is a recursive causal model, then there is always a unique solution to the equations in $M_{\vec{X} \leftarrow \vec{x}}$, given a setting \vec{u} for the variables in \mathcal{U} . Such a setting is called a *context*. Contexts will play the role of possible worlds when we model uncertainty.

Syntax and Semantics: Given a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a formula of the form $X = x$, for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$, is called a *primitive event*. A *basic causal formula* is one of the form $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k]\varphi$, where φ is a Boolean combination of primitive events; Y_1, \dots, Y_k, X are variables in \mathcal{V} ; Y_1, \dots, Y_k are distinct; $x \in \mathcal{R}(X)$; and $y_i \in \mathcal{R}(Y_i)$. Such a formula is abbreviated as $[\vec{Y} \leftarrow \vec{y}]\varphi$. The special case where $k = 0$ is abbreviated as φ . Intuitively, $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k]\varphi$ says that φ holds in the counterfactual world that would arise if Y_i is set to y_i , $i = 1, \dots, k$. A *causal formula* is a Boolean combination of basic causal formulas.

A causal formula φ is true or false in a causal model, given a context. We write $(M, \vec{u}) \models \varphi$ if φ is true in causal model M given context \vec{u} . $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$ if the variable X has value x in the unique (since we are dealing with recursive models) solution to the equations in $M_{\vec{Y} \leftarrow \vec{y}}$ in context \vec{u} (that is, the unique vector of values for the exogenous variables that simultaneously satisfies all equations $F_Z^{\vec{Y} \leftarrow \vec{y}}$, $Z \in \mathcal{V} - \vec{Y}$, with the variables in \mathcal{U} set to \vec{u}). We extend the definition to arbitrary causal formulas in the obvious way.

Note that the structural equations are deterministic. We later add probability to the picture by putting a probability on the set of contexts (i.e., on the possible worlds).

3 The Definition of Explanation

As we said in the introduction, many definitions of causal explanation have been given in the literature. The “classical” approaches in the philosophy literature, such as Hempel’s 1965 *deductive-nomological* model and Salmon’s 1989 *statistical relevance* model (as well as many other approaches)

have a serious problem: they fail to exhibit the directionality inherent in common explanations. While it seems reasonable to say “the height of the flag pole explains the length of the shadow”, it would sound awkward if one were to explain the former with the latter. Despite all the examples in the philosophy literature on the need for taking causality and counterfactuals into account, and the extensive work on causality defined in terms of counterfactuals in the philosophy literature, as Woodward 2001 observes, philosophers have been reluctant to build a theory of explanation on top of a theory of causality. The concern seems to be one of circularity.

In [Halpern and Pearl, 2001], we give a definition of causality that assumes that the causal model and all the relevant facts are given; the problem is to determine which of the given facts are causes. (We discuss this definition in more detail below.) We give a definition of explanation based on this definition of causality. The role of explanation is to provide the information needed to establish causation. As discussed in the introduction, we view an explanation as a fact that is not known for certain but, if found to be true, would constitute a genuine cause of the explanandum, regardless of the agent’s initial uncertainty. Thus, what counts as an explanation depends on what you already know and, naturally, the definition of explanation is relative to the agent’s epistemic state (as in Gärdenfors 1988). It is also natural, from this viewpoint, that an explanation includes fragments of the causal model M , or reference to the physical laws which underly the connection between the cause and the effect. To borrow an example from [Gärdenfors, 1988], if we want an explanation of why Mr. Johansson has been taken ill with lung cancer, the information that he worked in asbestos manufacturing for many years is not going to be a satisfactory explanation to someone who does not know anything about the effects of asbestos on people’s health. In this case, the causal model (or relevant parts of it) must be part of the explanation. On the other hand, for someone who knows the causal model but does not know that Mr. Johansson worked in asbestos manufacturing, the explanation would involve Mr. Johansson’s employment but would not mention the causal model.

Our definition of explanation is motivated by the following intuitions. An individual in a given epistemic state K asks why φ holds. What constitutes a good answer to his question? A good answer must provide information that goes beyond K and be such that the individual can see that it would, if true, be (or be very likely to be) a cause of φ . We may also want to require that φ be true (or at least probable). Although our basic definition does not require this, but it is easy to do so.

To make this precise, we must explain (1) what it means for ψ to be a cause of φ and (2) how to capture the agent’s epistemic state. In [Halpern and Pearl, 2001], we dealt with the first question. In the next subsection we review the definitions. The following subsections discuss the second question.

3.1 The Definition of Causality

We want to make sense of statements of the form “event A is an actual cause of event B in context \vec{u} of model M ”. Note that we assume the context and model are given. Intuitively, they encode the background knowledge. All the relevant facts are known. The only question is picking out which of them

are the causes of φ .

The types of events that we allow as actual causes are ones of the form $X_1 = x_1 \wedge \dots \wedge X_k = x_k$ —that is, conjunctions of primitive events; we typically abbreviate this as $\vec{X} = \vec{x}$. The events that can be caused are arbitrary Boolean combinations of primitive events. We argue in [Halpern and Pearl, 2001] that it is reasonable to restrict causes to conjunctions (and, in particular, to disallow disjunctions). This restriction seems less reasonable in the case of explanation; we return to this point below. In any case, the definition of causality we give is restricted to conjunctive causes.

Definition 3.1: (Actual cause) $\vec{X} = \vec{x}$ is an *actual cause* of φ in (M, \vec{u}) if the following three conditions hold:

- AC1. $(M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi$. (That is, both $\vec{X} = \vec{x}$ and φ are true in the actual world.)
- AC2. There exists a partition (\vec{Z}, \vec{W}) of \mathcal{V} with $\vec{X} \subseteq \vec{Z}$ and some setting (\vec{x}', \vec{w}') of the variables in (\vec{X}, \vec{W}) such that if $(M, \vec{u}) \models (Z = z^*)$ for $Z \in \vec{Z}$, then
- $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}'] \neg \varphi$. In words, changing (\vec{X}, \vec{W}) from (\vec{x}, \vec{w}) to (\vec{x}', \vec{w}') changes φ from true to false;
 - $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W} \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*] \varphi$ for all subsets \vec{Z}' of \vec{Z} . In words, setting \vec{W} to \vec{w}' should have no effect on φ as long as \vec{X} is kept at its current value \vec{x} , even if all the variables in an arbitrary subset of \vec{Z} are set to their original values in the context \vec{u} .
- AC3. \vec{X} is minimal; no subset of \vec{X} satisfies conditions AC1 and AC2. Minimality ensures that only those elements of the conjunction $\vec{X} = \vec{x}$ that are essential for changing φ in AC2(a) are considered part of a cause; inessential elements are pruned. ■

For future reference, we say that $\vec{X} = \vec{x}$ is a *weak cause* of φ in (M, \vec{u}) if AC1 and AC2 hold, but not necessarily AC3.

The core of this definition lies in AC2. Informally, the variables in \vec{Z} should be thought of as describing the “active causal process” from \vec{X} to φ . These are the variables that mediate between \vec{X} and φ . AC2(a) says that there exists a setting \vec{x}' of \vec{X} that changes φ to $\neg\varphi$, as long as the variables not involved in the causal process (\vec{W}) take on value \vec{w}' . AC2(a) is reminiscent of the traditional counterfactual criterion of Lewis 1986b, according to which φ should be false if it were not for \vec{X} being \vec{x} . However, AC2(a) is more permissive than the traditional criterion; it allows the dependence of φ on \vec{X} to be tested under special circumstances.

AC2(b) is an attempt to counteract the “permissiveness” of AC2(a) with regard to structural contingencies. Essentially, it ensures that \vec{X} alone suffices to bring about the change from φ to $\neg\varphi$; setting \vec{W} to \vec{w}' merely eliminates spurious side effects that tend to mask the action of \vec{X} . It captures the fact that setting \vec{W} to \vec{w}' should not affect the causal process, by requiring that changing \vec{W} from \vec{w} to \vec{w}' has no effect on the value of φ .

This definition is discussed and defended in much more detail in [Halpern and Pearl, 2001], where it is compared to other definitions of causality. In particular, it is shown to avoid a number of problems that have been identified with Lewis’s account (e.g., see [Pearl, 2000, Chapter 10]), such as commitment to transitivity of causes. For the purposes of this paper, we ask that the reader accept the definition. We note that, to some extent, our definition of explanation is modular in its use of causality, in that another definition of causality could be substituted for the one we use in the definition of explanation (provided it was given in the same framework).

The following example will help to clarify the definition of both causality and explanation.

Example 3.2: Suppose that two arsonists drop lit matches in different parts of a dry forest; both cause trees to start burning. Consider two scenarios. In the first, called “disjunctive,” either match by itself suffices to burn down the whole forest. That is, even if only one match were lit, the forest would burn down. In the second scenario, called “conjunctive,” both matches are necessary to burn down the forest; if only one match were lit, the fire would die down before the forest was consumed. We can describe the essential structure of these two scenarios using a causal model with four variables:

- an exogenous variable U which determines, among other things, the motivation and state of mind of the arsonists. For simplicity, assume that $\mathcal{R}(U) = \{u_{00}, u_{10}, u_{01}, u_{11}\}$; if $U = u_{ij}$, then the first arsonist intends to start a fire iff $i = 1$ and the second arsonist intends to start a fire iff $j = 1$. In both scenarios $U = u_{11}$.
- endogenous variables ML_1 and ML_2 , each either 0 or 1, where $ML_i = 0$ if arsonist i doesn’t drop the match and $ML_i = 1$ if he does, for $i = 1, 2$.
- an endogenous variable FB for forest burns down, with values 0 (it doesn’t) and 1 (it does).

Both scenarios have the same causal network (see Figure 1); they differ only in the equation for FB . Given $u \in \mathcal{R}(U)$, for the disjunctive scenario we have $F_{FB}(u, 1, 1) = F_{FB}(u, 0, 1) = F_{FB}(u, 1, 0) = 1$ and $F_{FB}(u, 0, 0) = 0$; for the conjunctive scenario we have $F_{FB}(u, 1, 1) = 1$ and $F_{FB}(u, 0, 0) = F_{FB}(u, 1, 0) = F_{FB}(u, 0, 1) = 0$.

In general, the causal model for reasoning about forest fires would involve many other variables; in particular, variables for other potential causes of forest fires such as lightning and unattended campfires. Here we focus on that part of the causal model that involves forest fires started by arsonists. Since for causality we assume that all the relevant facts are given, we can assume here that it is known that there were no unattended campfires and there was no lightning, which makes it safe to ignore that portion of the causal model.

Denote by M_1 and M_2 the (portion of the) causal models associated with the disjunctive and conjunctive scenarios, respectively. The causal network for the relevant portion of M_1 and M_2 is described in Figure 1.

Despite the differences in the underlying models, it is not hard to show that each of $ML_1 = 1$ and $ML_2 = 1$ is a cause of $FB = 1$ in both scenarios. We present the argument for $ML_1 = 1$ here. To show that $ML_1 = 1$ is a cause in M_1 let

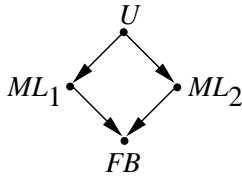


Figure 1: The causal network for M_1 and M_2 .

$\vec{Z} = \{ML_1, FB\}$, so $\vec{W} = \{ML_2\}$. It is easy to see that the contingency $ML_2 = 0$ satisfies the two conditions in AC2. AC2(a) is satisfied because, in the absence of the second arsonist ($ML_2 = 0$), the first arsonist is necessary and sufficient for the fire to occur ($FB = 1$). AC2(b) is satisfied because, if the first match is lit ($ML_1 = 1$) the contingency $ML_2 = 0$ does not prevent the fire from burning the forest. Thus, $ML_1 = 1$ is a cause of $FB = 1$ in M_1 . (Note that we needed to set ML_2 to 0, contrary to facts, in order to reveal the latent dependence of FB on ML_1 . Such a setting constitutes a structural change in the original model, since it involves the removal of some structural equations.) The argument that $ML_1 = 1$ is also a cause of $FB = 1$ in M_2 is similar. (Again, taking $\vec{Z} = \{ML_1, FB\}$ and $\vec{W} = \{ML_2\}$ works.)

This example also illustrates the need for the minimality condition AC3. For example, if lighting a match qualifies as the cause of fire then lighting a match and sneezing would also pass the tests of AC1 and AC2, and awkwardly qualify as the cause of fire. Minimality serves here to strip “sneezing” and other irrelevant, over-specific details from the cause.

It might be argued that allowing disjunctive causes would be useful in this case to distinguish M_1 from M_2 as far as causality goes. A purely counterfactual definition of causality would make $ML_1 = 1 \vee ML_2 = 1$ a cause of $FB = 1$ in M_1 (since, if $ML_1 = 1 \vee ML_2 = 1$ were not true, then $FB = 1$ would not be true), but would make neither $ML_1 = 1$ nor $ML_2 = 1$ individually a cause (for example, if $ML_1 = 1$ were not true in M_1 , $FB = 1$ would still be true). Clearly, our definition does not enforce this intuition. Purely counterfactual definitions of causality have other well-known problems. We do not have a strong intuition as to the best way to deal with disjunction in the context of causality, and believe that disallowing it is reasonably consistent with intuitions. Interestingly, as we shall see in Section 3.2, our definition of explanation *does* distinguish M_1 from M_2 ; each of $ML_1 = 1$ and $ML_2 = 1$ is an explanation of $FB = 1$ in M_1 under our definition of explanation, but neither is an explanation of $FB = 1$ in M_2 . In M_2 , the explanation of $FB = 1$ is $ML_1 = 1 \wedge ML_2 = 1$: both matches being lit are necessary to explain the forest burning down. ■

3.2 The Basic Definition of Explanation

All that remains to do before giving the definition of explanation is to discuss how to capture the agent’s epistemic state in our framework. For ease of exposition, we first consider the case where the causal model is known and the context is uncertain. (The minor modifications required to deal with the general case are described in Section 3.4.) In that case, one way of describing an agent’s epistemic state is by simply de-

scribing the set of contexts the agent considers possible. This choice is very much in the spirit of the standard “possible worlds” definitions of knowledge and belief.

Definition 3.3: (Explanation) Given a structural model M , $\vec{X} = \vec{x}$ is an explanation of φ relative to a set \mathcal{K} of contexts if the following conditions hold:

- EX1. $(M, \vec{u}) \models \varphi$ for each context $\vec{u} \in \mathcal{K}$. (That is, φ must hold in all contexts the agent considers possible—the agent considers what she is trying to explain as an established fact)
- EX2. $\vec{X} = \vec{x}$ is a *weak cause* of φ in (M, \vec{u}) (that is, AC1 and AC2 hold, but not necessarily AC3) for each $\vec{u} \in \mathcal{K}$ such that $(M, \vec{u}) \models \vec{X} = \vec{x}$.
- EX3. \vec{X} is minimal; no subset of \vec{X} satisfies EX2.
- EX4. $(M, \vec{u}) \models \neg(\vec{X} = \vec{x})$ for some $\vec{u} \in \mathcal{K}$ and $(M, \vec{u}') \models \vec{X} = \vec{x}$ for some $\vec{u}' \in \mathcal{K}$. (This just says that the agent considers a context possible where the explanation is false, so the explanation is not known to start with, and considers a context possible where the explanation is true, so that it is not vacuous.) ■

Our requirement EX4 that the explanation is not known may seem incompatible with linguistic usage. Someone discovers some fact A and says “Aha! That explains why B happened.” Clearly, A is not an explanation of why B happened relative to the epistemic state *after* A has been discovered, since at that point A is known. However, A can legitimately be considered an explanation of B relative to the epistemic state before A was discovered.

Consider the arsonists in Example 3.2. If the causal model has only arsonists as the cause of the fire, there are two possible explanations in the disjunctive scenario: arsonist 1 did it or arsonist 2 did it (assuming \mathcal{K} consists of three contexts, where either 1, 2, or both set the fire). In the conjunctive scenario, no explanation is necessary, since the agent knows that both arsonists must have lit a match if arson is the only possible cause of the fire (assuming that the agent considers these to be the only possible arsonists).

Perhaps more interesting is to consider a causal model with other possible causes, such as lightning and unattended campfires. Since the agent knows that there was a fire, in each of the contexts in \mathcal{K} , at least one of the potential causes must have actually occurred. If we assume that there is a context where only arsonist 1 lit the fire (and, say, there was lightning) and another where only arsonist 2 lit the fire then, in the conjunctive scenario, $ML_1 = 1 \wedge ML_2 = 1$ is an explanation of $FB = 1$, but neither $ML_1 = 1$ nor $ML_2 = 1$ by itself is an explanation (since neither by itself is a cause in all contexts in \mathcal{K} that satisfy the formula). On the other hand, in the disjunctive scenario, both $ML_1 = 1$ and $ML_2 = 1$ are explanations.

It is worth noting here that the minimality clause EX3 applies to all contexts. This means that our rough gloss of $\vec{X} = \vec{x}$ being an explanation of φ relative to a set \mathcal{K} of contexts if $\vec{X} = \vec{x}$ is a cause of φ in each context in \mathcal{K} where $\vec{X} = \vec{x}$ holds is not quite correct. For example, although

$ML_1 = 1 \wedge ML_2 = 1$ is an explanation of fire in the conjunctive scenario (if \mathcal{K} includes contexts where there are other possible causes of fire), it is a cause of fire in none of the contexts in which it holds. The minimality condition AC3 would say that each of $ML_1 = 1$ and $ML_2 = 1$ is a cause, but their conjunction is not.

Note that, as for causes, we have disallowed disjunctive explanations. Here the motivation is less clear cut. It does make perfect sense to say that the reason that φ happened is either A or B (but I don't know which). There are some technical difficulties with disjunctive explanations, which suggest philosophical problems. For example, consider the conjunctive scenario of the arsonist example again. Suppose that the structural model is such that the only causes of fire are the arsonists, lightning, and unattended campfires and that \mathcal{K} consists of contexts where each of these possibilities is the actual cause of the fire. Once we allow disjunctive explanations, what is the explanation of fire? One candidate is "either there were two arsonists or there was lightning or there was an unattended campfire (which got out of hand)". But this does not satisfy EX4, since the disjunction is true in every context in \mathcal{K} . On the other hand, if we do not allow the disjunction of all possible causes, which disjunction should be allowed as an explanation? As a technical matter, how should the minimality condition EX3 be rewritten? We could not see any reasonable way to allow some disjunctions in this case without allowing the disjunction of all causes (which will not in general satisfy EX4).

We believe that, in cases where disjunctive explanations seem appropriate, it is best to capture this directly in the causal model by having a variable that represents the disjunction. (Essentially the same point is made in [Chajewska and Halpern, 1997].) For example, consider the disjunctive scenario of the arsonist example, where there are other potential causes of the fire. If we want to allow "there was an arsonist" to be an explanation without specifically mentioning who the arsonist is, then it can be easily accomplished by replacing the variables ML_1 and ML_2 in the model by a variable ML which is 1 iff at least one arsonist drops a match. Then $ML = 1$ becomes an explanation, without requiring disjunctive explanations.

Why not just add ML to the model rather than using it to replace ML_1 and ML_2 ? We have implicitly assumed in our framework that all possible combinations of assignments to the variables are possible (i.e., there is a structural contingency for any setting of the variables). If we add ML and view it as being logically equivalent to $ML_1 \vee ML_2$ (that is, $ML = 1$ by definition iff at least one of ML_1 and ML_2 is 1) then, for example, it is logically impossible for there to be a structural contingency where $ML_1 = 0$, $ML_2 = 0$, and $ML = 1$. Thus, in the presence of logical dependencies, it seems that we need to restrict the set of contingencies that can be considered to those that respect the dependencies. We have not yet considered the implications of such a change for our framework, so we do not pursue the matter here.

3.3 Partial Explanations and Explanatory Power

Not all explanations are considered equally good. Some explanations are more likely than others. An obvious way to

define the "goodness" of an explanation is by bringing probability into the picture. Suppose that the agent has a probability on the set \mathcal{K} of possible contexts. In this case, we can consider the probability of the set of contexts where the explanation $\vec{X} = \vec{x}$ is true. For example, if the agent has reason to believe that the first arsonist is extremely unlikely to have caused the fire (perhaps he had defective matches), then the set of contexts where $ML_2 = 1$ holds would have higher probability than those where $ML_1 = 1$ holds. Thus, $ML_2 = 1$ would be considered a better explanation of the fire in the disjunctive model than $ML_1 = 1$.

But the probability of an explanation is only part of the story; the other part concerns the degree to which an explanation fulfills its role (relative to φ) in the various contexts considered. This becomes clearer when we consider *partial* explanations. The following example, taken from [Gärdenfors, 1988], is one where partial explanations play a role.

Example 3.4: Suppose I see that Victoria is tanned and I seek an explanation. Suppose that the causal model includes variables for "Victoria took a vacation in the Canary Islands", "sunny in the Canary Islands", and "went to a tanning salon". The set \mathcal{K} includes contexts for all settings of these variables compatible with Victoria being tanned. Note that, in particular, there is a context where Victoria both went to the Canaries (and didn't get tanned there, since it wasn't sunny) and to a tanning salon. Gärdenfors points out that we normally accept "Victoria took a vacation in the Canary Islands" as a satisfactory explanation of Victoria being tanned and, indeed, according to his definition, it is an explanation. Victoria taking a vacation is not an explanation (relative to the context \mathcal{K}) in our framework, since there is a context $\vec{u}^* \in \mathcal{K}$ where Victoria went to the Canary Islands but it was not sunny, and in \vec{u}^* the actual cause of her tan is the tanning salon, not the vacation.

For us, the explanation would have to be "Victoria went to the Canary Islands *and* it was sunny." In this case we can view "Victoria went to the Canary Islands" as a partial explanation (in a formal sense to be defined below). ■

In Example 3.4 the partial explanation can be extended to a full explanation by adding a conjunct. But not every partial explanation can be extended to a full explanation. Roughly speaking, the full explanation may involve exogenous factors, which are not permitted in explanations. Assume, for example, that going to a tanning salon was not considered an endogenous variable in our model but, rather, the model simply had an exogenous variable U_s that could make Victoria suntanned even in the absence of sun in Canary islands. Likewise, assume that the weather in Canary island was also part of the background context. In this case, we would still consider Victoria's vacation to provide a partial explanation of her sun tan, since the context where it fails to be a cause (no sun in the Canary island) is fairly unlikely, but we cannot add conjuncts to this event to totally exclude that context from the agent's realm of possibilities.

The situation actually is quite common, as the following example shows.

Example 3.5: Suppose that the sound on a television works but there is no picture. Furthermore, the only cause of there

being no picture that the agent is aware of is the picture tube being faulty. However, the agent is also aware that there are times when there is no picture even though the picture tube works perfectly well—intuitively, “for inexplicable reasons”. This is captured by the causal network described in Figure 2, where T describes whether or not the picture tube is working (1 if it is and 0 if it is not) and P describes whether or not there is a picture (1 if there is and 0 if there is not). The ex-

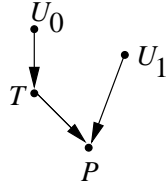


Figure 2: The television with no picture.

ogenous variable U_0 determines the status of the picture tube: $T = U_0$. The exogenous variable U_1 is meant to represent the mysterious “other possible causes”. If $U_1 = 0$, then whether or not there is a picture depends solely on the status of the picture tube—that is, $P = T$. On the other hand, if $U_1 = 1$, then there is no picture ($P = 0$) no matter what the status of the picture tube. Thus, in contexts where $U_1 = 1$, $T = 0$ is *not* a cause of $P = 0$. Now suppose that \mathcal{K} includes a context \vec{u}_{00} where $U_0 = U_1 = 0$. Then it is easy to see that there is no explanation of $P = 0$. The only plausible explanation, that the picture tube is not working, is not a cause of $P = 0$ in the context \vec{u}_{00} . On the other hand, $T = 0$ is a cause of $P = 0$ in all other contexts in \mathcal{K} satisfying $T = 0$. If the probability of \vec{u}_{00} is small (capturing the intuition that it is unlikely that more than one thing goes wrong with a television at once), then we are entitled to view $T = 0$ as a quite good partial explanation of $P = 0$. ■

These examples motivate the following definition.

Definition 3.6: Let $\mathcal{K}_{\vec{X}=\vec{x},\varphi}$ be the largest subset \mathcal{K}' of \mathcal{K} such that $\vec{X} = \vec{x}$ is an explanation of φ relative to $\mathcal{K}_{\vec{X}=\vec{x},\varphi}$. (It is easy to see that there is a largest such set.) Then $\vec{X} = \vec{x}$ is a *partial explanation of φ with goodness* $\Pr(\mathcal{K}_{\vec{X}=\vec{x},\varphi} | \vec{X} = \vec{x})$. Thus, the goodness of a partial explanation measures the extent to which it provides an explanation of φ .¹ ■

In Example 3.4, if the agent believes that it is sunny in the Canary Islands with probability .9 (that is, the probability that it was sunny given that Victoria is suntanned and that she went to the Canaries is .9), then Victoria going to the Canaries is a partial explanation of her being tanned with goodness .9. The relevant set \mathcal{K}' consists of those contexts where it is sunny in the Canaries. Similarly, in Example 3.5, if the agent

¹Here and elsewhere, a formula such as $\vec{X} = \vec{x}$ is being identified with the set of contexts where the formula is true. Recall that, since all contexts in \mathcal{K} are presumed to satisfy φ , there is no need to condition on φ ; this probability is already updated with the truth of the explanandum φ . Finally, note that our usage of partial explanation is related to, but different from, that in [Chajewska and Halpern, 1997].

believes that the probability of both the picture tube being faulty and the other mysterious causes being operative is .1, then $T = 0$ is a partial explanation of $P = 0$ with goodness .9 (with \mathcal{K}' consisting of all the contexts where $U_1 = 1$).

A full explanation is clearly a partial explanation with goodness 1, but we are often satisfied with partial explanations $\vec{X} = \vec{x}$ that are not as good, especially if they have high probability (i.e., if $\Pr(\vec{X} = \vec{x})$ is high). In general, there is a tension between the goodness of an explanation and its probability.

These ideas also lead to a definition of explanatory power. Consider Example 3.2 yet again, and suppose that there is an endogenous random variable O corresponding to the presence of oxygen. Now if $O = 1$ holds in all the contexts that the agent considers possible, then $O = 1$ is excluded as an explanation by EX4. But suppose that $O = 0$ holds in one context that the agent considers possible (for example, there may be another combustible gas), albeit a very unlikely one. In that case, $O = 1$ becomes a very good partial explanation of the fire. Nevertheless, it is an explanation with, intuitively, very little explanatory power. How can we make this precise?

Suppose that there is a probability distribution \Pr^- on a set \mathcal{K}^- of contexts larger than \mathcal{K} that intuitively represents the agent’s prior probability before the explanandum φ is discovered. That is, \Pr is the result of conditioning \Pr^- on φ and \mathcal{K} consists of the subset of \mathcal{K}^- that satisfies φ . Gärdenfors identifies the explanatory power of the (partial) explanation $\vec{X} = \vec{x}$ of φ with $\Pr^-(\varphi | \vec{X} = \vec{x})$ (see [Chajewska and Halpern, 1997; Gärdenfors, 1988]). If this probability is higher than $\Pr^-(\varphi)$, then the explanation makes φ more likely. While this explanatory power, we would argue that a better measure of the explanatory power of $\vec{X} = \vec{x}$ is $\Pr^-(\mathcal{K}_{\vec{X}=\vec{x},\varphi} | \vec{X} = \vec{x})$. According to either definition, under reasonable assumptions about \Pr^- , $O = 1$ has much lower explanatory power than, say $ML = 1$. Moreover, the two definitions agree in the case that $\vec{X} = \vec{x}$ is a full explanation (since then $\mathcal{K}_{\vec{X}=\vec{x},\varphi}$ is just \mathcal{K} , the set of contexts in \mathcal{K}^- where φ is true). The difference between the two definitions arises if there are contexts where φ and $\vec{X} = \vec{x}$ both happen to be true, but $\vec{X} = \vec{x}$ is not a cause of φ . Such spurious correlations are excluded by our suggested definition. (See [Pearl, 2000] for some examples showing that considering spurious correlations leads to bad outcomes.)

Again, (partial) explanations with higher explanatory power typically are more refined and, hence, less likely. than explanations with less explanatory power. There is no obvious way to resolve this tension. (See [Chajewska and Halpern, 1997] for more discussion of this issue.)

As this discussion suggests, our definition shares some features with that of Gärdenfors’ 1988. Like him, we consider explanation relative to an agent’s epistemic state. Gärdenfors also considers a “contracted” epistemic state characterized by the distribution \Pr^- . Intuitively, \Pr^- describes the agent’s beliefs before discovering φ . (More accurately, it describes an epistemic state as close as possible to \Pr where the agent does not ascribe probability 1 to φ .) If the agent’s current belief in φ came about as the result of an observation ψ , then

we can take \Pr to be the result of conditioning \Pr^- on ψ , as we have done above. However, Gärdenfors does necessarily assume such a connection between \Pr and \Pr^- . In any case, for Gärdenfors, $\vec{X} = \vec{x}$ is an explanation of φ relative to \Pr if (1) $\Pr(\varphi) = 1$, (2) $0 < \Pr(\vec{X} = \vec{x}) < 1$, and (3) $\Pr^-(\varphi|\vec{X} = \vec{x}) > \Pr^-(\varphi)$. (1) is the probabilistic analogue of EX1. Clearly, (2) is the probabilistic analogue of EX4. Finally, (3) says that learning the explanation increases the likelihood of φ . Gärdenfors focuses on the explanatory power of an explanation, but does not take into account its prior probability. As pointed out in [Chajewska and Halpern, 1997], Gärdenfors' definition suffers from another defect: Since there is no minimality requirement like EX3, if $\vec{X} = \vec{x}$ is an explanation of φ , so too is $\vec{X} = \vec{x} \wedge \vec{Y} = \vec{y}$.

In contrast to Gärdenfors' definition, the dominant approach to explanation in the AI literature, the *maximum a posteriori* (MAP) approach (see, for example, [Henrion and Druzdzel, 1990; Pearl, 1988; Shimony, 1991]), focuses on the probability of the explanation, given the explanandum (i.e., $\Pr^-(\vec{X} = \vec{x}|\varphi) = \Pr(\vec{X} = \vec{x})$), but does not take explanatory power into account. The MAP approach is based on the intuition that the best explanation for an observation is the state of the world (in our setting, the context) that is most probable given the evidence. The most probable explanation for φ is then the context \vec{u}^* such that $\Pr(\vec{u}^*) = \max_{\vec{u}} \Pr(\vec{u})$. Thus, an explanation is a (complete) context. This means that part of the explanation will include totally irrelevant facts (the agent sneezed). Moreover, it is quite sensitive to the description of the context (see [Chajewska and Halpern, 1997] for details) and does not directly take causality into account.

To some extent, these problems can be dealt with by limiting the set of candidate explanations to ancestors (of the explanandum) in the causal network; this also avoids many of the problems associated with non-causal approaches (although it requires there to be a causal network in the background). However, the MAP approach does not go far enough. One problem is that propositions with extremely high prior probabilities (e.g., that oxygen is present in the room) will also receive high posterior probabilities, regardless of how relevant they are to the events explained. To remedy this problem, more intricate combinations of the quantities $\Pr(\vec{X} = \vec{x})$, $\Pr^-(\varphi|\vec{X} = \vec{x})$, and $\Pr^-(\varphi)$ have been suggested to quantify the causal relevance of $\vec{X} = \vec{x}$ on φ but, as argued in [Pearl, 2000, p. 221], without taking causality into account, no such combination of parameters can work.

3.4 The General Definition

In general, an agent may be uncertain about the causal model, so an explanation will have to include information about it. (Gärdenfors 1988 and Hempel 1965 make similar observations). It is relatively straightforward to extend our definition of explanation to accommodate this. Now an epistemic state \mathcal{K} consists not only of contexts, but of pairs (M, \vec{u}) consisting of a causal model M and a context \vec{u} . Call such a pair a *situation*. Intuitively, now an explanation should consist of some causal information (such as “prayers do not cause fires”) and the facts that are true. Thus, a (*general*) *explanation* has the form $(\psi, \vec{X} = \vec{x})$, where ψ is an arbitrary formula in our

causal language and, as before, $\vec{X} = \vec{x}$ is a conjunction of primitive events. We think of the ψ component as consisting of some causal information (such as “prayers do not cause fires”), which corresponds to a conjunction of statements of the form $F = i \Rightarrow [P \leftarrow x](F = i)$, where P is a random variable describing whether or not prayer takes place). The first component in a general explanation is viewed as restricting the set of causal models. To make this precise, given a causal model M , we say ψ is *valid in M* , and write $M \models \psi$, if $(M, \vec{u}) \models \psi$ for all contexts \vec{u} consistent with M . With this background, it is easy to state the general definition.

Definition 3.7: $(\psi, \vec{X} = \vec{x})$ is an explanation of φ relative to a set \mathcal{K} of situations if the following conditions hold:

- EX1. $(M, \vec{u}) \models \varphi$ for each situation $(M, \vec{u}) \in \mathcal{K}$.
- EX2. For all $(M, \vec{u}) \in \mathcal{K}$ such that $(M, \vec{u}) \models \vec{X} = \vec{x}$ and $M \models \psi$, $\vec{X} = \vec{x}$ is a weak cause of φ in (M, \vec{u}) .
- EX3. $(\psi, \vec{X} = \vec{x})$ is minimal; there is no pair $(\psi', \vec{X}' = \vec{x}') \neq (\psi, \vec{X} = \vec{x})$ satisfying EX2 such that $\{M'' \in \mathcal{M}(\mathcal{K}) : M'' \models \psi'\} \supseteq \{M'' \in \mathcal{M}(\mathcal{K}) : M'' \models \psi\}$, where $\mathcal{M}(\mathcal{K}) = \{M : (M, \vec{u}) \in \mathcal{K} \text{ for some } \vec{u}\}$, $\vec{X}' \subseteq \vec{X}$, and \vec{x}' is the restriction of \vec{x} to the variables in \vec{X}' . Roughly speaking, this says that no subset of X provides a weak cause of φ in more contexts than those where ψ is valid.
- EX4. $(M, \vec{u}) \models \neg(\vec{X} = \vec{x})$ for some $(M, \vec{u}) \in \mathcal{K}$ and $(M', \vec{u}') \models \vec{X} = \vec{x}$ for some $(M', \vec{u}') \in \mathcal{K}$. ■

Note that, in EX2, we now restrict to situations $(M, \vec{u}) \in \mathcal{K}$ that satisfy both parts of the explanation $(\psi, \vec{X} = \vec{x})$, in that $M \models \psi$ and $(M, \vec{u}) \models \vec{X} = \vec{x}$. Furthermore, although both components of an explanation are formulas in our causal language, they play very different roles. The first component serves to restrict the set of causal models considered (to those with the appropriate structure); the second describes a cause of φ in the resulting set of situations.

Clearly Definition 3.3 is the special case of Definition 3.7 where there is no uncertainty about the causal structure (i.e., there is some M such that if $(M', \vec{u}) \in \mathcal{K}$, then $M = M'$). In this case, it is clear that we can take ψ in the explanation to be *true*.

Definition 3.7 can also be extended to deal naturally with statistical information of the kind considered by Gärdenfors and Hempel. Let a *probabilistic causal model* be a tuple $M_{Pr} = (\mathcal{S}, \mathcal{F}, \Pr)$, where $M = (\mathcal{S}, \mathcal{F})$ is a causal model and \Pr is a probability measure on the contexts defined by signature \mathcal{S} of M . Information like “with probability .9, $X = 3$ ” is a restriction on probabilistic models, and thus can be captured using a formula in an appropriate extension of our language that allows such probabilistic reasoning. With this extended language, the definition of explanation using probabilistic causal models remains unchanged.

As an orthogonal issue, there is also no difficulty considering a probability on the set \mathcal{K} of situations and defining partial explanation just as before.

Example 3.8: Using this general definition of explanation, consider Scriven’s 1959 famous paresis example. Paresis develops only in patients who have been syphilitic for a long time, but only a small number of patients who are syphilitic in fact develop paresis. Furthermore, according to Scriven, no other factor is known to be relevant in the development of paresis.² This description is captured by a simple causal model M_P . There are two endogenous variables, S (for syphilis) and P (for paresis), and two exogenous variables, U_1 , the background factors that determine S , and U_2 , which intuitively represents “disposition to paresis”, i.e., the factors that determine, in conjunction with syphilis, whether or not paresis actually develops. An agent who knows this causal model and that a patient has paresis does not need an explanation of why: the agent knows without being told that the patient must have syphilis and that $U_2 = 1$. On the other hand, for an agent who does not know the causal model (i.e., considers a number of causal models of paresis possible), $(\{M_P\}, S = 1)$ is an explanation of paresis. ■

4 Discussion

We have given a formal definition of explanation in terms of causality. As we mentioned earlier, there are not too many formal definitions of explanation in terms of causality in the literature. One of the few exceptions is Lewis 1986a, who defends the thesis that “to explain an event is to provide some information about its causal history”. While this view is compatible with our definition, there is no formal definition given to allow for a careful comparison between the approaches. In any case, if were to define causal history in terms of Lewis’s 1986b definition of causality, we would inherit all the problems of that definition. As we said earlier, our definition avoids these problems.

So what are the problems with our definition? For one thing, it inherits whatever problems our definition of causality has. As observed in [Halpern and Pearl, 2001], our definition at times declares certain events to be causes (and hence candidate explanations) that, intuitively, should not be causes because they should fail AC2(a). The only reason that they do not fail AC2(a) is because of extremely unlikely structural contingencies. To some extent, we can avoid this problem by simply ignoring structural contingencies that are extremely unlikely (this is essentially the solution suggested in [Halpern and Pearl, 2001] in the context of causality). Of course, we can do this in the context of explanation too. Another possibility is to take the probability of the structural contingency into account more directly when computing the probability of the explanation. We are currently exploring this option.

We have mentioned the other significant problem of the definition already: dealing with disjunctive explanations. Disjunctions cause problems in the definition of causality, which is why we do not deal with them in the context of explanation. As we pointed out earlier, it may be possible to modify the definition of causality so as to be able to deal with conjunctions without changing the structure of our definition of explanation. We are currently exploring this.

²Apparently there are now other known factors, but this does not change the import of the example.

Finally, our definition gives no tools for dealing with the inherent tension between explanatory power, goodness of partial beliefs, and the probability of the explanation. Clearly this is an area that requires further work.

Acknowledgments:

Thanks to Riccardo Pucella and Vicky Weissman for useful comments.

References

- [Chajewska and Halpern, 1997] U. Chajewska and J. Y. Halpern. Defining explanation in probabilistic systems. In *Proc. UAI '97*, pages 62–71, 1997.
- [Galles and Pearl, 1997] D. Galles and J. Pearl. Axioms of causal relevance. *Artificial Intelligence*, 97(1–2):9–43, 1997.
- [Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux*. MIT Press, 1988.
- [Halpern, 2000] J. Y. Halpern. Axiomatizing causal reasoning. *Journal of A.I. Research*, pages 317–337, 2000.
- [Halpern and Pearl, 2001] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach— Part I: Causes. Available at <http://www.cs.cornell.edu/home/halpern>, 2001.
- [Hempel, 1965] C. G. Hempel. *Aspects of Scientific Explanation*. Free Press, 1965.
- [Henrion and Druzdzel, 1990] M. Henrion and M. J. Druzdzel. Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In *Uncertainty in Artificial Intelligence 6*, Elsevier Science, pages 17–32, 1990.
- [Lewis, 1986a] D. Lewis. Causal explanation. In *Philosophical Papers*, volume II, pages 214–240. Oxford University Press, 1986.
- [Lewis, 1986b] D. Lewis. Causation. In *Philosophical Papers*, volume II, pages 159–213. Oxford University Press, 1986. The original version of this paper, without numerous postscripts, appeared in the *Journal of Philosophy* **70**, 1973, pp. 113–126.
- [Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- [Pearl, 2000] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.
- [Salmon, 1989] W. C. Salmon. *Four Decades of Scientific Explanation*. University of Minnesota Press, 1989.
- [Scriven, 1959] M. J. Scriven. Explanation and prediction in evolutionary theory. *Science*, 130:477–482, 1959.
- [Shimony, 1991] S. E. Shimony. Explanation, irrelevance and statistical independence. In *Proc. AAAI '91*, pages 482–487, 1991.
- [Woodward, 2001] J. Woodward. Explanation. In *The Blackwell Guide to the Philosophy of Science*. Basil Blackwell, 2001. To appear.

Complexity Results for Structure-Based Causality

Thomas Eiter and Thomas Lukasiewicz

Institut und Ludwig Wittgenstein Labor für Informationssysteme, TU Wien

Favoritenstraße 9–11, A-1040 Wien, Austria

{eiter, lukasiewicz}@kr.tuwien.ac.at

Abstract

We analyze the computational complexity of causal relationships in Pearl’s structural models, where we focus on causality between variables, event causality, and probabilistic causality. In particular, we analyze the complexity of the sophisticated notions of weak and actual causality by Halpern and Pearl. In the course of this, we also prove an open conjecture by Halpern and Pearl, and establish other semantic results. To our knowledge, no complexity aspects of causal relationships have been considered so far, and our results shed light on this issue.

1 Introduction

Representing and reasoning with causal knowledge has received much attention in the recent decade. The existing approaches to causality in the AI literature can be roughly divided into those that have been developed as modal nonmonotonic logics (especially in the context of logic programming) and those that evolved from the area of Bayesian networks.

A representative of the former is Geffner’s modal nonmonotonic logic for handling causal knowledge [1990; 1992], which has been inspired by default reasoning from conditional knowledge bases. Other more specialized formalisms play an important role in dealing with causal knowledge about actions and change; see especially the work by Turner [1999] and the references therein for an overview.

A representative of the latter is Pearl’s approach to modeling causality by structural equations [Balke and Pearl, 1994; Galles and Pearl, 1997; Pearl, 1999; 2000], which is central to a number of recent research efforts. In particular, the evaluation of deterministic and probabilistic counterfactuals has been explored, which is at the core of problems in fault diagnosis, planning, decision making, and determination of liability [Balke and Pearl, 1994].

In a recent paper, Halpern [2000] gave an axiomatization of reasoning about causal formulas in the structural-model approach, and explored its computational aspects.

It has been shown that the structural-model approach allows a precise modeling of many important causal relationships, which can especially be used in natural language processing [Galles and Pearl, 1997]. In particular, it allows an

elegant definition of the important notions of actual causation and explanation [Halpern and Pearl, 2000; 2001].

We give a simple example due to Halpern and Pearl [2000], which illustrates the structural-model approach.

Example 1.1 Suppose that two arsonists lit matches in different parts of a dry forest, and both cause trees to start burning. Assume now that either match by itself suffices to burn down the whole forest. In the structural-model framework, such a scenario may be modeled as follows. We assume two binary background variables U_1 and U_2 , which determine the motivation and the state of mind of the two arsonists, where U_i has the value 1 iff the arsonist i intends to start a fire. Moreover, we have three binary variables A_1 , A_2 , and F , which describe the observable situation, where A_i has the value 1 iff the arsonist i drops the match, and F has the value 1 iff the whole forest burns down. The causal dependencies between these variables are expressed through functions, which say that the value of A_i is given by the value of U_i , and that F has the value 1 iff either A_1 or A_2 has the value 1. These dependencies can be graphically represented as in Fig. 1.

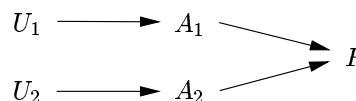


Figure 1: Causal Graph

While the semantic aspects of causal relationships in the structural-model approach have been explored in depth (see especially the work by Pearl [2000]), studies about their computational properties are missing so far. In this paper, we try to fill this gap by giving a precise account of the complexity of deciding causal relationships in structural models.

Note that Halpern’s work [2000] is orthogonal to ours, as it focuses on the computational aspects of deciding whether a given causal formula has a causal model, while our work in this paper deals with the complexity of deciding whether a given causal relationship holds in a given causal model.

The main contributions can be summarized as follows:

- We analyze the complexity of deciding causal relationships between variables in structural causal models. We consider the notions of causal irrelevance, cause, cause in a context, direct cause, and indirect cause. It turns out that testing

these notions has a complexity among NP, co-NP, and D^P . Hardness holds even in restricted cases.

- We analyze the complexity of deciding causal relationships between events. We consider the notions of necessary and possible causality, and the sophisticated notions of weak and actual causality by Halpern and Pearl [2000]. In particular, checking the latter is shown to be Σ_2^P -complete in general, and NP-complete in the case of binary variables.

- We prove some semantic results related to the notions of actual and weak causality. More precisely, we prove an open conjecture by Halpern and Pearl [2000], and we give a new characterization of weak causality for the binary case.

- As a representative for probabilistic causal relationships, we finally analyze the complexity of deciding the notion of probabilistic causal irrelevance, which is shown to be complete for the class \mathbb{C} , and thus harder than co-NP. Note that few \mathbb{C} -complete problems, and none in AI, are known.

Our results draw a clear picture of the complexity of structural causality, and give useful insight for exploiting it, e.g., in counterfactual reasoning (see Section 6).

For space reasons, we give only proof sketches of some results. Proofs of all results are given in the extended paper [Eiter and Lukasiewicz, 2001].

2 Preliminaries

We assume a finite set of *random variables*. Each variable X_i may take on *values* from a finite *domain* $D(X_i)$. A *value* for a set of variables $X = \{X_1, \dots, X_n\}$ is a mapping $x: X \rightarrow D(X_1) \cup \dots \cup D(X_n)$ such that $x(X_i) \in D(X_i)$ (where the empty mapping \emptyset is the unique value for $X = \emptyset$). The *domain* of X , denoted $D(X)$, is the set of all values for X . For $Y \subseteq X$ and $x \in D(X)$, denote by $x|Y$ the restriction of x to Y . For sets of variables X, Y and values $x \in D(X)$ and $y \in D(Y)$, we use xy to denote the union of x and y . As usual, we often identify singletons $\{X_i\}$ with X_i , and their values x with $x(X_i)$.

2.1 Causal and Probabilistic Causal Models

A *causal model* M is a triple (U, V, F) , where U is a finite set of *exogenous* variables, V is a finite set of *endogenous* variables such that $U \cap V = \emptyset$, and $F = \{F_X \mid X \in V\}$ is a set of functions $F_X: D(PA_X) \rightarrow D(X)$ that assign a value to X for each value of the *parents* $PA_X \subseteq U \cup V \setminus \{X\}$ of X .

We focus here on the principal class of *recursive* causal models $M = (U, V, F)$ (as argued in [Halpern and Pearl, 2000], we do not lose much generality by concentrating on recursive causal models) in which a total ordering \prec on V exists such that $Y \in PA_X$ implies $Y \prec X$, for all $X, Y \in V$. In such models, every assignment to the exogenous variables $U = u$ determines a unique value y for every set of endogenous variables $Y \subseteq V$, denoted $Y_M(u)$ (or simply $Y(u)$). In the sequel, M is reserved for denoting a recursive causal model.

For causal models $M = (U, V, F)$, $X \subseteq V$, and $x \in D(X)$, the causal model $M_{X=x} = (U, V, F_{X=x})$, where $F_{X=x} = \{F_Y \mid Y \in V \setminus X\} \cup \{F_{X'} = x(X') \mid X' \in X\}$, is a *submodel* of M . Intuitively, X is set to x by an “external action”. We abbreviate $M_{X=x}$ and $F_{X=x}$ by M_x and F_x , respectively. For $Y \subseteq V$, we abbreviate $Y_{M_x}(u)$ by $Y_x(u)$.

A *probabilistic causal model* (M, P) consists of a causal model $M = (U, V, F)$ and a probability function P on $D(U)$.

Example 2.1 In our introductory example, the causal model $M = (U, V, F)$ is given by $U = \{U_1, U_2\}$, $V = \{A_1, A_2, F\}$, and $F = \{F_{A_1}, F_{A_2}, F_F\}$, where $F_{A_1} = U_1$, $F_{A_2} = U_2$, and $F_F = 1$ iff $A_1 = 1$ or $A_2 = 1$ (Fig. 1 shows the parent relationship between the variables). In a probabilistic causal model (M, P) , we may use the uniform distribution P over $D(U)$.

2.2 Model Representation for Computation

We assume the following representation of causal models $M = (U, V, F)$ and probabilistic causal models (M, P) (see the full paper for a discussion of these assumptions):

- The domain $D(X)$ of each variable $X \in U \cup V$ is explicit, i.e., $D(X) = \{v_1, \dots, v_k\}$ is enumerated.

- Each function $F_X: D(PA_X) \rightarrow D(X)$, $X \in V$, is computable in polynomial time.

- P is given by a pair (f, b) , where $f: D(U) \rightarrow \{0, 1, 2, \dots\}$ is a polynomial-time computable function and $b > 0$ is an integer, such that $P(u) = f(u) / b$ for every $u \in D(U)$.

The following proposition is immediate.

Proposition 2.2 For every $X, Y \subseteq V$ and $x \in D(X)$, the values $Y(u)$ and $Y_x(u)$, given $u \in D(U)$, are computable in polynomial time.

We say M (resp., (M, P)) is *binary*, if $|D(X)| = 2$ for all $X \in V$. Furthermore, M (resp., (M, P)) is *bounded*, if $|PA_X| \leq k$ holds for each $X \in V$, i.e., X has at most k parents, where k is an arbitrary but fixed constant.

Note that in a bounded model, as for polynomiality, each F_X is w.l.o.g. given by a table that lists the output values $F_X(v)$ for all $v \in D(PA_X)$ (which is similar to a conditional probability table in Bayesian networks).

2.3 Complexity Classes

Complexity classes that we encounter are shown in Fig. 2, where arrows denote containment. P, NP, co-NP, Σ_2^P , and Π_2^P are from the Polynomial Hierarchy (PH); $D^P = \{L \cap L' \mid L \in \text{NP}, L' \in \text{co-NP}\}$ is the “conjunction” of NP and co-NP. The class \mathbb{C} is from the Counting Hierarchy (CH) of complexity classes [Torán, 1991]. Informally, \mathbb{C} contains all problems which can be expressed as deciding whether a given instance I has at least $f(|I|)$ many polynomial size “proofs” $J_1, \dots, J_{f(|I|)}$ that I is a Yes-instance, for some function f , where computing $f(|I|)$ and checking each proof J_i is polynomial. The class \mathbb{C} coincides with the famous class PP (probabilistic P) [Papadimitriou, 1994], which contains the problems decidable by a polynomial-time Turing machine which accepts an input iff the majority of its runs halt in an accepting state.

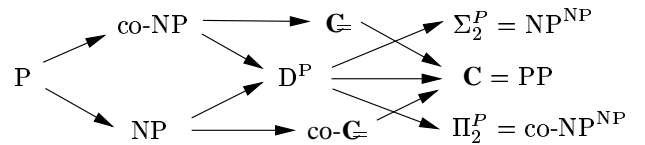


Figure 2: Containment between Complexity Classes

The class \mathbb{C} is a variant of \mathbb{C} , where “exactly $f(|I|)$ ” replaces “at least $f(|I|)$ ”. While this difference seems marginal, \mathbb{C} and \mathbb{C} have quite different properties [Torán, 1991]. Intuitively, \mathbb{C} is an extension of co-NP, and has many properties of this class. \mathbb{C} and \mathbb{C} are contained in PSPACE, and it is believed that they are not contained in PH.

3 Causality between Variables

We first focus on causal relationships between variables due to Galles and Pearl [1997]; see also [Pearl, 1999; 2000].

3.1 Definitions

In the sequel, we assume a causal model $M = (U, V, F)$ and sets of variables $X, Y, Z \subseteq V$ with $X, Y \neq \emptyset$. We say

- X is *causally irrelevant* to Y given Z iff for every $W \subseteq V \setminus X \cup Y \cup Z$, $u \in D(U)$, $x, x' \in D(X)$, $z \in D(Z)$, and $w \in D(W)$, we have $Y_{xzw}(u) = Y_{x'zw}(u)$.
- X is a *cause* of Y iff values $x, x' \in D(X)$ and $u \in D(U)$ exist such that $Y_x(u) \neq Y_{x'}(u)$.
- X is a *cause* of Y in the context $Z = z$ iff values $x, x' \in D(X)$ and $u \in D(U)$ exist with $Y_{xz}(u) \neq Y_{x'z}(u)$.
- X is a *direct cause* of Y iff values $x, x' \in D(X)$, $u \in D(U)$, and $z \in D(V \setminus X \cup Y)$ exist with $Y_{xz}(u) \neq Y_{x'z}(u)$.
- X is an *indirect cause* of Y iff X is a cause of Y and X is not a direct cause of Y .

We give some examples of such causal relationships. For a more detailed discussion of these concepts see [Galles and Pearl, 1997] and [Pearl, 1999; 2000].

Example 3.1 In our running example, A_1 is not causally irrelevant to F . For instance, if we set $A_2 \in V \setminus \{A_1, F\}$ to 0 and A_1 to 0 and 1, then F has the values 0 and 1, respectively. Informally, the actions of arsonist 1 are not causally irrelevant to the state of the forest. In fact, A_1 is a cause of F , but not a cause of F in the context $A_2 = 1$. Informally, the actions of arsonist 1 are in general a cause of the state of the forest, but not when arsonist 2 starts a fire. Finally, it is easy to verify that A_1 is in fact a direct cause of F .

3.2 Results

Our complexity results for checking the above notions of causality are summarized in Table 1. It is important to point out that for all these causal relationships, hardness holds even if M is binary and bounded, and X is a singleton.

Table 1: Complexity of Causality between Variables

X is causally irrelevant to Y given Z	co-NP-complete
X is a cause of Y	NP-complete
X is a cause of Y in the context $Z = z$	NP-complete
X is a direct cause of Y	NP-complete
X is an indirect cause of Y	D ^P -complete

Our first result shows that deciding causal irrelevance is co-NP-complete. We sketch the main ideas of its proof.

Theorem 3.2 Given $M = (U, V, F)$ and $X, Y, Z \subseteq V$ with $X, Y \neq \emptyset$, deciding whether X is causally irrelevant to Y given Z is co-NP-complete. Hardness holds even if (1) M is binary and bounded, (2) Z is empty, (3) X is a singleton, and (4a) Y is a singleton or (4b) $V = X \cup Y \cup Z$.

Proof (sketch). The problem is in co-NP, as $W \subseteq V \setminus X \cup Y \cup Z$ and values $u \in D(U)$, $x, x' \in D(X)$, $z \in D(Z)$, and $w \in D(W)$ such that $Y_{xzw}(u) \neq Y_{x'zw}(u)$ can be guessed and verified in polynomial time (see Proposition 2.2).

We show co-NP-hardness by a polynomial transformation from the co-NP-complete problem of deciding whether a given propositional formula in 3DNF $\phi = \phi_1 \vee \dots \vee \phi_k$ on the atoms A_1, \dots, A_n , where w.l.o.g. $k \geq 2$ and $n \geq 1$, is a tautology.

We now construct $M = (U, V, F)$ and $X, Y, Z \subseteq V$ such that X is causally irrelevant to Y given Z iff ϕ is a tautology. We define $X = \{A\}$, $Y = \{B\}$, and $Z = \emptyset$. Moreover, $U = \{A_1, \dots, A_n\}$ and $V = X \cup Y \cup \{D_1, \dots, D_{k-1}\}$. Let all variables have the domain $\{0, 1\}$, where 0 and 1 are identified with the truth values **false** and **true**, respectively. The values 0 and 1 of X are denoted by x_0 and x_1 , respectively. The functions F_S for $S \in V$ are as follows:

- $F_A = 1$ and $F_{D_1} = A \vee \phi_1$,
- $F_{D_i} = D_{i-1} \vee \phi_i$ for all $i \in \{2, \dots, k-1\}$,
- $F_B = D_{k-1} \vee \phi_k$.

It can now be shown that X is causally irrelevant to Y given Z iff ϕ is a tautology. Informally, under any $u \in D(U)$, if X is set to x_1 , then Y becomes 1. Whereas, if X is set to x_0 , then Y is the truth value of ϕ under u . That is, setting X to x_0 and x_1 yields the same value of Y under any $u \in D(U)$ iff ϕ is a tautology. Notice that assignments to some nonempty $W \subseteq D_1, \dots, D_{k-1}$ always yield the same value of Y .

Note that (1)–(3) and (4a) are satisfied. The proof of hardness in the case where (1)–(3) and (4b) holds is similar. \square

The following result shows that deciding causes and causes in a context is NP-complete. Here, NP-hardness can be shown by a reduction from the NP-complete problem of deciding whether a propositional formula is not a tautology.

Theorem 3.3 Given $M = (U, V, F)$ and $X, Y \subseteq V$ (resp., $X, Y, Z \subseteq V$ and $z \in D(Z)$) with $X, Y \neq \emptyset$, deciding whether X is a cause of Y (resp., X is a cause of Y in the context $Z = z$) is NP-complete. Hardness holds even if (1) M is binary and bounded, and (2) X and Y are singletons.

4 Event Causality

We now focus on causality between events. In particular, we consider causal relationships between events due to Galles and Pearl [1997]; see also [Pearl, 1999; 2000], and the notions of weak and actual causality by Halpern and Pearl [2000], which are inspired by Pearl’s causal beams [2000].

4.1 Definitions

A *primitive event* is an expression of the form $Y = y$, where Y is a variable and y is a value for Y . The set of *events* is the

closure of the set of primitive events under the Boolean operations \neg and \wedge (that is, every primitive event is an event, and if ϕ and ψ are events, then also $\neg\phi$ and $\phi \wedge \psi$).

The *truth* of an event ϕ in a causal model $M = (U, V, F)$ under $u \in D(U)$, denoted $(M, u) \models \phi$, is inductively given by:

- $(M, u) \models Y = y$ iff $Y_M(u) = y$,
- $(M, u) \models \neg\phi$ iff $(M, u) \models \phi$ does not hold,
- $(M, u) \models \phi \wedge \psi$ iff $(M, u) \models \phi$ and $(M, u) \models \psi$.

We write $\phi(u)$ to abbreviate $(M, u) \models \phi$. For $X \subseteq V$ and $x \in D(X)$, we write $\phi_x(u)$ to abbreviate $(M_x, u) \models \phi$. For $X = \{X_1, \dots, X_k\} \subseteq V$ with $k \geq 1$ and $x_i \in D(X_i)$, we use $X = x_1 \cdots x_k$ to abbreviate $X_1 = x_1 \wedge \dots \wedge X_k = x_k$.

The following proposition is immediate.

Proposition 4.1 *Let $X \subseteq V$ and $x \in D(X)$. Given $u \in D(U)$ and an event ϕ , deciding whether $\phi(u)$ and $\phi_x(u)$ (given x) hold can be done in polynomial time.*

We are now ready to define causal relationships between events. We first define the notions of necessary and possible causality (which are slightly more general than in [Galles and Pearl, 1997]). Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$. Let ϕ be an event. We say

- $X = x$ *always causes* ϕ iff (i) $\phi_x(u)$ for all $u \in D(U)$, and (ii) some $x' \in D(X)$, $u' \in D(U)$ exist with $x' \neq x$ and $\neg\phi_{x'}(u')$.
- $X = x$ *may have caused* ϕ iff (i) $X = x$ and ϕ are observed (which implies that $X(u) = x$ and $\phi(u)$ for some $u \in D(U)$) and (ii) some $x' \in D(X)$ and $u \in D(U)$ exist such that $x' \neq x$, $X(u) = x$, $\phi(u)$, and $\neg\phi_{x'}(u)$.

We next define weak and actual causality. We say $X = x$ is a *weak cause* of ϕ under u iff the following conditions hold:

AC1. $X(u) = x$ and $\phi(u)$.

AC2. Some set of variables $W \subseteq V \setminus X$ and some values $\bar{x} \in D(X)$ and $w \in D(W)$ exist such that:

- (a) $\neg\phi_{\bar{x}w}(u)$,
- (b) $\phi_{xw\hat{z}}(u)$ for all $\hat{Z} \subseteq V \setminus (X \cup W)$ and $\hat{z} = \hat{Z}(u)$.

We say $X = x$ is an *actual cause* of ϕ under u iff additionally the following condition is satisfied:

AC3. X is minimal. That is, no proper subset of X satisfies both AC1 and AC2.

We give some illustrative examples.

Example 4.2 In our running example, each event among $A_1 = 1$, $A_2 = 1$, and $A_1 = 1 \wedge A_2 = 1$ always causes $F = 1$. For instance, let us show that $A_1 = 1$ always causes $F = 1$: (i) if A_1 is set to 1, then F has the value 1 under every $u \in D(U)$, and (ii) if U_2 is set to 0, and A_1 to 0, then F has the value 0. Informally, at least one arsonist starting a fire always has the effect that the whole forest burns down.

Consider now the background context $u = (1, 1)$ in which both arsonists intend to start a fire. Then, each among $A_1 = 1$, $A_2 = 1$, and $A_1 = 1 \wedge A_2 = 1$ is a weak cause of $F = 1$. For instance, let us show that $A_1 = 1$ is a weak cause of $F = 1$: (AC1) both A_1 and F have the value 1 under u , (AC2 (a)) if both A_1 and A_2 are set to 0, then F has the value 0, and (AC2 (b)) if A_1 is set to 1 and A_2 to 0, then F has the value 1. In fact, $A_1 = 1$ and $A_2 = 1$ are actual causes of $F = 1$, while $A_1 = 1 \wedge A_2 = 1$ is not an actual cause of $F = 1$.

4.2 Results

Our complexity results for the above causal relationships between events are summarized in Table 2. We distinguish between the general and the binary case, in which we assume a syntactic restriction to binary causal models. We remark that for all these causal relationships between events, hardness holds even if M is bounded and X is a singleton.

Table 2: Complexity of Event Causality

Problem	general case	binary case
$X = x$ always causes ϕ	D^P -complete	D^P -complete
$X = x$ may cause ϕ	NP-complete	NP-complete
$X = x$ is a weak cause of ϕ	Σ_2^P -complete	NP-complete
$X = x$ is an actual cause of ϕ	Σ_2^P -complete	NP-complete

The following theorem shows that deciding weak causality is Σ_2^P -complete in the general case. We sketch the main ideas behind the technically quite involved proof of this result.

Theorem 4.3 *Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ , deciding whether $X = x$ is a weak cause of ϕ under u is Σ_2^P -complete. Hardness holds even if X is a singleton, and either M is bounded or ϕ is primitive.*

Proof (sketch). As for membership in Σ_2^P , recall that $X = x$ is a weak cause of ϕ under u iff AC1 and AC2 hold. By Proposition 4.1, verifying $X(u) = x$ and $\phi(u)$ is polynomial. Moreover, some W , \bar{x} , and w as in AC2 can be guessed and verified with an NP-oracle (needed for (b)) in polynomial time. In summary, checking AC1 and AC2 is in Σ_2^P .

Σ_2^P -hardness is shown by a reduction from evaluating a quantified Boolean formula $\Phi = \exists A \forall B \gamma$, where γ is a propositional formula on the variables $A = \{A_1, \dots, A_m\}$ and $B = \{B_1, \dots, B_n\}$. We now define M , etc. as stated such that $X = x$ is a weak cause of ϕ under u iff Φ is true (deciding this is Σ_2^P -complete [Papadimitriou, 1994]).

We define $X = \{G\}$ and $Y = \{H\}$. The exogenous and endogenous variables are $U = \{E\}$ and $V = X \cup Y \cup \{C\} \cup A \cup B$, respectively. Define $D(S) = \{0, 1, 2\}$ for all $S \in B$, and $D(S) = \{0, 1\}$ for all $S \in U \cup V \setminus B$. We define

$$\phi' = (\gamma' \wedge \bigwedge_{S \in B} S \neq 2) \vee (C = 0) \vee (G = 1 \wedge C = 1 \wedge \bigvee_{S \in B} S \neq 2),$$

where γ' is obtained from γ by replacing each $S \in A \cup B$ by “ $S = 1$ ”. The functions F_S with $S \in V$ are defined by:

- $F_S = 0$ for all $S \in X \cup \{C\} \cup A$,
- $F_S = G + C$ for all $S \in B$, and $F_H = 1$ iff ϕ' is true.

Let $u \in D(U)$ and $x = 0$. Let ϕ be $Y = 1$. It can now be shown that $X = x$ is a weak cause of ϕ under u iff Φ is true. More precisely, AC1 is trivially satisfied, and AC2 holds iff Φ is true. Roughly speaking, the existential and universal quantification over A and B in Φ is reflected by the variables in W and $V \setminus (X \cup W)$, respectively. We then especially have to ensure that (i) $B \cap W = \emptyset$ and (ii) no truth assignment to the variables in B is ignored. In detail, to make ϕ false in AC2 (a), C must be set to 1 and all $S \in B$ must have

the value 2. Whereas, to make ϕ true in AC2 (b), all $S \in B$ must have a value from $\{0, 1\}$. This already ensures (i). Since G is set to 0 in AC2 (b), and $F_S = G + C$ for all $S \in B$, each variable in B has the value 1 in AC2 (b). As every $S \in B$ has the value 0 in M , this then ensures (ii).

Note that X is a singleton and that ϕ is primitive. To show Σ_2^P -hardness for a singleton X and bounded M , define $M' = (U, V \setminus \{H\}, F \setminus \{F_H\})$. Then, $X = x$ is a weak cause of ϕ' under u iff Φ evaluates to true. \square

The just sketched proof of Σ_2^P -hardness makes use of *non-binary* causal models. Thus, we may ask whether deciding weak causality in the binary case has a lower complexity.

In fact, the following semantic result shows that in the binary case, AC2 can be expressed in a different way.

Theorem 4.4 *Let $M = (U, V, F)$ be binary. Let $X \subseteq V$ and $x \in D(X)$. Let ϕ be an event. Then, $X = x$ is a weak cause of ϕ under $u \in D(U)$ iff AC1 holds, and (AC2') some $W \subseteq V \setminus X$, $\bar{x} \in D(X)$, and $w \in D(W)$ exist such that (a) $\neg \phi_{\bar{x}w}(u)$, (b) $\phi_{xw}(u)$, and (c) $Z_{xw}(u) = Z(u)$ for $Z = V \setminus (X \cup W)$.*

Proof (sketch). Notice that AC2' (a) is identical to AC2 (a). Moreover, AC2' (b) can be replaced by AC2 (b), as AC2' (c) implies that setting $\hat{Z} \subseteq V \setminus (X \cup W)$ to $\hat{z} = \hat{Z}(u)$ is immaterial in $\phi_{xw\hat{z}}(u)$ of AC2 (b). Thus, it is now sufficient to show that for binary M , we can add AC2' (c) to AC2 (a) and (b).

Roughly speaking, we can additionally satisfy AC2' (c) by iteratively moving variables from $V \setminus (X \cup W)$ to the W -part in AC2 (a) and (b). More precisely, any singleton $S \subseteq V \setminus (X \cup W)$ with $S_{xw}(u) \neq S(u)$ can be moved to the W -part assigning them $S_{\bar{x}w}(u)$. This is always feasible for $S_{\bar{x}w}(u) = S_{xw}(u)$. If M is binary, this is also feasible for $S_{\bar{x}w}(u) \neq S_{xw}(u)$, which then implies $S_{\bar{x}w}(u) = S(u)$. This construction can now be iterated until AC2' (c) holds. \square

Based on this result, it can now be shown that deciding weak causality in the binary case is NP-complete. This is more formally expressed by the following theorem.

Theorem 4.5 *Given a binary $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ , deciding whether $X = x$ is a weak cause of ϕ under u is NP-complete. Hardness holds even if M is bounded, X is a singleton, and ϕ is primitive.*

We next focus on the problem of deciding actual causality. We first prove a conjecture by Halpern and Pearl [2000], which says that only primitive events can be actual causes. Note that the proof also goes through for their setting of possibly infinite domains and sets of endogenous variables.

Theorem 4.6 *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$. Let ϕ be an event. If $X = x$ is an actual cause of ϕ under u , then X is a singleton.*

Proof (sketch). We give a proof by contradiction. Let $X = x$ be an actual cause of ϕ under u . That is, AC1–AC3 hold. In particular, AC2 (a) and (b) hold for some $W \subseteq V \setminus X$ and some $\bar{x} \in D(X)$ and $w \in D(W)$. Suppose that X is not a singleton. We consider two cases:

Case 1: There exists some nonempty $X'' \subseteq X$, $X'' \neq X$, such that $\phi_{x'\bar{x}''w\hat{z}}(u)$ for all $\hat{Z} \subseteq V \setminus (X \cup W)$, where $X' = X \setminus X''$, $x' = x|X'$, $\bar{x}'' = \bar{x}|X''$, and $\hat{z} = \hat{Z}(u)$. Informally,

we can then move all variables in X'' to the W -part of AC2 assigning them \bar{x}'' . That is, set in AC2 $\bar{x}' = \bar{x}|X'$ and $w' = \bar{x}''w$ where $W' = X'' \cup W$ is the respective W -part. Then, $X' = x'$ is another weak cause of ϕ under u , which is smaller than $X = x$. This is a contradiction.

Case 2: For every nonempty $X'' \subseteq X$, $X'' \neq X$, there exists some $\hat{Z} \subseteq V \setminus (X \cup W)$ such that $\neg \phi_{x'\bar{x}''w\hat{z}}(u)$, where $X' = X \setminus X''$, $x' = x|X'$, $\bar{x}'' = \bar{x}|X''$, and $\hat{z} = \hat{Z}(u)$. Informally, we can then take any such X'' and move the variables in X' and \hat{Z} to the W -part of AC2 assigning them values $x' = x|X'$ and $\hat{z} = \hat{Z}(u)$. That is, set in AC2 $x'' = x|X''$ and $w' = x'w\hat{z}$ where $W' = X' \cup W \cup \hat{Z}$ is the respective W -part. Then, $X'' = x''$ is a weak cause of ϕ under u (note that each instance of AC2(b) for $X'' = x''$ amounts to an instance of AC2(b) for $X = x$). This is a contradiction. \square

Hence, $X = x$ is an actual cause of ϕ under u iff (i) $X = x$ is a weak cause of ϕ under u , and (ii) $X = x$ is primitive. As an immediate corollary of Theorems 4.3 and 4.6, it thus follows that deciding whether $X = x$ is an actual cause of ϕ under u is Σ_2^P -complete in the general case.

Corollary 4.7 *Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ , deciding whether $X = x$ is an actual cause of ϕ under u is Σ_2^P -complete. Hardness holds even if X is a singleton, and M is bounded or ϕ is primitive.*

Furthermore, as an immediate corollary of Theorems 4.5 and 4.6, we obtain that deciding whether $X = x$ is an actual cause of ϕ under u is NP-complete in the binary case.

Corollary 4.8 *Given a binary $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ , deciding whether $X = x$ is an actual cause of ϕ under u is NP-complete. Hardness holds even if M is bounded, X is a singleton, and ϕ is primitive.*

5 Probabilistic Causality

As a representative of probabilistic causal relationships, we finally consider the notion of probabilistic causal irrelevance by Galles and Pearl [1997]; see also [Pearl, 1999; 2000].

A *counterfactual formula* is an expression $Y_x = y$, where $X, Y \subseteq V$, $x \in D(X)$, and $y \in D(Y)$. Given a probabilistic causal model (M, P) , where $M = (U, V, F)$, the *probability* of $Y_x = y$, denoted $P(Y_x = y)$, is the sum of all $P(u)$ such that $Y_x(u) = y$. For $X, Y, Z \subseteq V$ with $X, Y \neq \emptyset$, we say X is *probabilistically causally irrelevant* to Y given Z , denoted $(X \not\rightarrow Y | Z)_P$, iff for all $x, x' \in D(X)$, $y \in D(Y)$, and $z \in D(Z)$, it holds that $P(Y_{xz} = y) = P(Y_{x'z} = y)$. Intuitively, once the value of Z is fixed at z , the value of X does not affect the probability of Y .

The following theorem shows that deciding probabilistic causal irrelevance is complete for \mathbf{C} .

Theorem 5.1 *Given a probabilistic causal model (M, P) , where $M = (U, V, F)$, and $X, Y, Z \subseteq V$, deciding whether $(X \not\rightarrow Y | Z)_P$ is complete for \mathbf{C} .*

Theorem 5.1 is nontrivial and needs some explanations. Firstly, the result means that, in a sense, testing probabilistic causal irrelevance is harder than co-NP, and thus not polynomially reducible to SAT-testing. Moreover, it cannot be reduced to any problem solver for problems in the polynomial

hierarchy. On the other hand, the problem is “easier” than PP-complete problems, which could perhaps help in finding polynomial time (randomized) approximation algorithms.

The easier part of this result is the lower bound. Hardness for \mathbb{C}^- can be proved by a reduction from the following \mathbb{C}^- -complete problem HALFSAT: Given a propositional formula in 3DNF γ on the variables x_1, \dots, x_n , decide whether exactly half of the assignments to x_1, \dots, x_n satisfy γ . The construction is based on ideas in the proof of Theorem 3.2, but more involved. In fact, it establishes hardness for the case where M is binary and bounded, Z is empty, X, Y are singletons, and P is the uniform distribution. Thus, the problem shows its full complexity already in a minimalistic setting.

The more difficult part is membership in \mathbb{C}^- . Recall that $(X \not\rightarrow Y | Z)_P$ iff $P(Y_{xz} = y) = P(Y_{x'z} = y)$ for all values x', x, y , and z . Checking that $P(Y_{xz} = y) = P(Y_{x'z} = y)$, for given x, x', y , and z , can be reduced to the following problem EQUALRUN, which is in \mathbb{C}^- : Given two NP Turing machines T_1 and T_2 and an input string w , decide whether T_1 and T_2 have the same number of accepting runs on w .

Indeed, let P be given by (f, b) as described in Section 2.2. Let the NP Turing machines T_1 and T_2 have input x, x', y, z and nondeterministically generate all $u \in D(U)$. Then, for each u , let T_1 and T_2 generate $f(u)$ paths. On each of these paths, T_1 (resp., T_2) computes deterministically $Y_{xz}(u)$ (resp., $Y_{x'z}(u)$), and accepts if this value coincides with y , otherwise it rejects. Then, $P(Y_{xz} = y) = P(Y_{x'z} = y)$ iff T_1 and T_2 have the same numbers of accepting paths.

Obviously, T_1 and T_2 can be constructed in polynomial time from M and x, x', y, z . However, we actually need to test that $P(Y_{xz} = y) = P(Y_{x'z} = y)$ for all values x', x, y , and z . What we obtain (by slight adaptations) is that the problem is in the class $\forall\mathbb{C}^-$, which is a generalization of \mathbb{C}^- similar to Π_2^P for co-NP: deciding whether I is a Yes-instance can be informally expressed as deciding whether for every polynomial size string J , it holds that I, J is a Yes-instance of the same problem in \mathbb{C}^- . Since $\forall\mathbb{C}^-$ is equal to \mathbb{C}^- (cf. [Green, 1993]), this reduction actually proves membership in \mathbb{C}^- .

6 Conclusion and Outlook

We analyzed the complexity of causal relationships in Pearl’s structural models. In particular, we considered causality between variables, event causality, and probabilistic causality. It turned out that all discussed notions of causality are intractable, where the sophisticated notions of weak and actual causality, and the notion of probabilistic causal irrelevance have the highest complexity (Σ_2^P and \mathbb{C}^- , respectively).

Our results give useful insight, and may be exploited e.g. in evaluating probabilistic counterfactuals as defined in [Balke and Pearl, 1994]. Note that the evaluation of probabilistic counterfactuals can be polynomially reduced to standard inference tasks in Bayesian networks, and thus has similar computational properties. It is easy to see that the complexity of computing conditional probabilities in Bayesian networks, which is complete for #P [Roth, 1996], carries over to computing probabilities of counterfactual statements. Similar to independencies [Pearl, 2000], deterministic and probabilistic causal relationships might now be used to simplify the eval-

uation of probabilistic counterfactuals. By our results, this seems reasonable, as the complexity of testing simple causal relationships (at most D^P , \mathbb{C}^-) is much lower than the complexity of evaluating probabilistic counterfactuals (#P-hard).

Moreover, our results may help to analyze the computational aspects of explanations and partial explanations as introduced by Halpern and Pearl [2000], which are crucially based on the notions of weak and actual causality.

An interesting topic of future research is to explore whether there are restricted cases in which testing causal relationships in the structural-model approach is tractable. For example, probabilistic causal irrelevance in stable causal models [Galles and Pearl, 1997] can be tested in polynomial time as it coincides with path interception in their causal graphs.

Acknowledgments

This work has been partially supported by the Austrian Science Fund Project N Z29-INF and by a DFG grant. We are very grateful to Jacobo Torán for useful comments related to counting classes and for pointing out relevant literature. Many thanks also to the reviewers for their useful comments.

References

- [Balke and Pearl, 1994] A. Balke and J. Pearl. Probabilistic evaluation of counterfactual queries. In *Proceedings AAAI-94*, pages 230–237, 1994.
- [Eiter and Lukasiewicz, 2001] T. Eiter and T. Lukasiewicz. Complexity results for structure-based causality. Technical Report INFSYS RR-1843-01-01, Institut für Informationssysteme, TU Wien, 2001.
- [Galles and Pearl, 1997] D. Galles and J. Pearl. Axioms of causal relevance. *Artif. Intell.*, 97:9–43, 1997.
- [Geffner, 1990] H. Geffner. Causal theories for nonmonotonic reasoning. In *Proc. AAAI-90*, pages 524–530, 1990.
- [Geffner, 1992] H. Geffner. *Default Reasoning: Causal and Conditional Theories*. MIT Press, 1992.
- [Green, 1993] F. Green. On the power of deterministic reductions to $\mathbb{C}=\text{P}$. *Math. Syst. Theory*, 26(2):215–233, 1993.
- [Halpern and Pearl, 2000] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Technical Report R-266, UCLA Cognitive Systems Lab, 2000.
- [Halpern and Pearl, 2001] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach — Part II: Explanation. In *Proceedings IJCAI-01*, 2001.
- [Halpern, 2000] J. Y. Halpern. Axiomatizing causal reasoning. *J. Artif. Intell. Res.*, 12:317–337, 2000.
- [Papadimitriou, 1994] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.
- [Pearl, 1999] J. Pearl. Reasoning with cause and effect. In *Proceedings IJCAI-99*, pages 1437–1449, 1999.
- [Pearl, 2000] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.
- [Roth, 1996] D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1–2):273–302, 1996.
- [Torán, 1991] J. Torán. Complexity classes defined by counting quantifiers. *J. ACM*, 38(3):753–774, 1991.
- [Turner, 1999] H. Turner. A logic of universal causation. *Artif. Intell.*, 113:87–123, 1999.

KNOWLEDGE REPRESENTATION AND REASONING

SPATIAL REASONING

Ambiguity-Directed Sampling for Qualitative Analysis of Sparse Data from Spatially-Distributed Physical Systems

Chris Bailey-Kellogg

Dartmouth Computer Science Dept.
6211 Sudikoff Laboratory
Hanover, NH 03755
cbk@cs.dartmouth.edu

Naren Ramakrishnan

Virginia Tech Dept. of Computer Science
629 McBryde Hall
Virginia Tech, VA 24061
naren@cs.vt.edu

Abstract

A number of important scientific and engineering applications, such as fluid dynamics simulation and aircraft design, require analysis of spatially-distributed data from expensive experiments and complex simulations. In such data-scarce applications, it is advantageous to use models of given sparse data to identify promising regions for additional data collection. This paper presents a principled mechanism for applying domain-specific knowledge to design focused sampling strategies. In particular, our approach uses ambiguities identified in a multi-level qualitative analysis of sparse data to guide iterative data collection. Two case studies demonstrate that this approach leads to highly effective sampling decisions that are also explainable in terms of problem structures and domain knowledge.

1 Introduction

A number of important scientific and engineering applications, such as fluid dynamics simulation and aircraft design, require qualitative analysis of spatially-distributed data from expensive experiments and/or complex simulations demanding days, weeks, or even years on petaflops-class computing systems. For example, Fig. 1 shows a cross-section of the design space for a multidisciplinary aircraft design problem involving 29 design variables with 68 constraints in a highly non-convex design space [Knill *et al.*, 1999]. Frequently, the designer will change some aspect of a nominal design point, and run a simulation to see how the change affects the objective function and various constraints dealing with aircraft geometry and performance/aerodynamics. This approach is inadequate for exploring such large high-dimensional design spaces, even at low fidelity. Ideally, the design engineer would like a high-level mining system to identify the *pockets* that contain good designs and which merit further consideration; traditional tools from optimization and approximation theory can then be applied to fine-tune such preliminary analyses.

Two important characteristics distinguish these applications. First, they must deal not with an abundance of data, but rather with a scarcity of data, owing to the cost and time

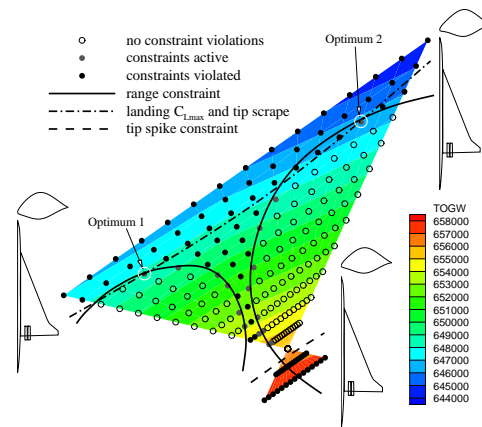


Figure 1: A pocket in an aircraft design space viewed as a slice through three design points (courtesy Layne T. Watson).

involved in conducting simulations. Second, and more importantly, the computational scientist has complete control over the data acquisition process (e.g. regions of the design space where data can be collected), especially via computer simulations. It is natural therefore to focus data collection so as to maximize information content, minimize the number and expense of samples, and so forth.

This paper presents a principled mechanism for applying domain-specific knowledge to focus sampling strategies for data-scarce applications. In particular, *ambiguities* identified by a multi-level qualitative analysis of data collected in one iteration guide succeeding iterations of data collection so as to improve the qualitative analysis. This approach leads to highly effective sampling decisions that are *explainable* in terms of problem structures and domain knowledge. We demonstrate the effectiveness of our approach by two case studies: (1) identification of pockets in n -dimensional space, and (2) decomposition of a field based on control influences.

2 Qualitative Analysis of Spatially-Distributed Physical Systems

The mechanism we develop for ambiguity-directed sampling is based on the Spatial Aggregation Language (SAL) [Bailey-

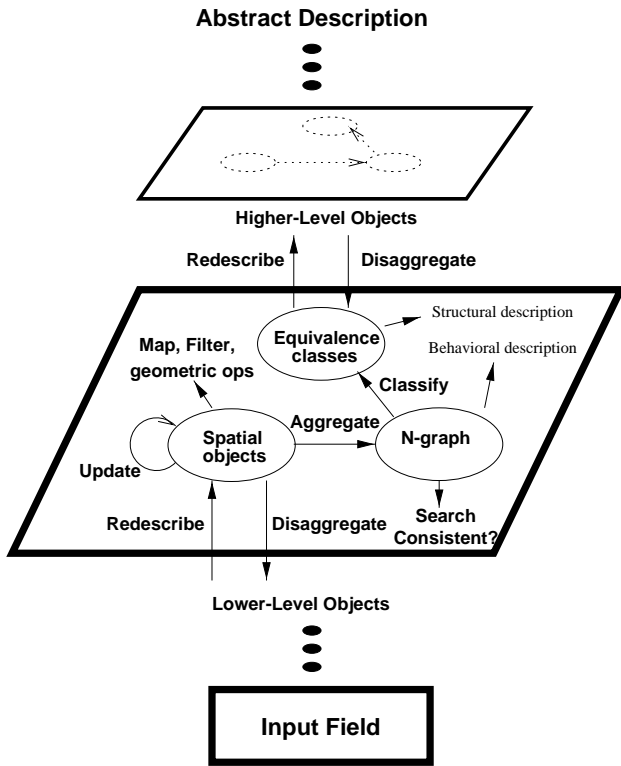


Figure 2: SAL multi-layer spatial aggregates, uncovered by a uniform vocabulary of operators utilizing domain knowledge.

Kellogg *et al.*, 1996; Yip and Zhao, 1996], which supports construction of data interpretation and control design applications for spatially-distributed physical systems. SAL programs uncover and manipulate multi-layer geometric and topological structures in spatially distributed data by using a small number of uniform operators and data types, parameterized by domain-specific knowledge. These operators and data types mediate increasingly abstract descriptions of the input data, as shown in Fig. 2. They utilize knowledge of physical properties such as continuity and locality, based on specified metrics, adjacency relations, and equivalence predicates, to uncover regions of uniformity in spatially distributed data.

As an example (see Fig. 3), consider a SAL program for bundling the vectors in a given vector field (e.g. wind velocity or temperature gradient) into a set of streamlines (paths through the field following the vector directions):

1. *Aggregate* vectors into a neighborhood graph (say 8-adjacency), localizing computation.
2. *Filter* edges in the graph, ensuring edge direction is similar enough to vector direction.
3. Cluster into *equivalence classes* neighboring vectors whose directions match best.
4. *Redescribe* equivalence classes of vectors into more abstract streamline curves.

In a second level of abstraction, streamlines are aggregated and classified into groups with similar flow behavior

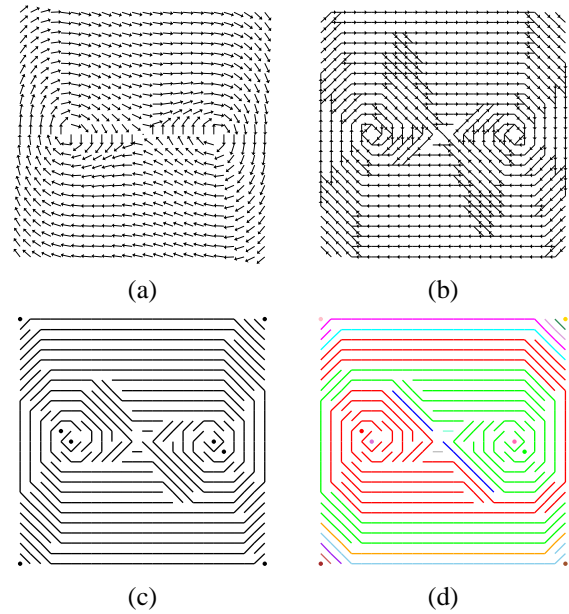


Figure 3: Example steps in SAL vector field analysis. (a) Input vector field. (b) Filtered neighborhood graph. (c) Equivalence classes (make a choice at each fork edge) redescribed as streamline curves. (d) Higher-level aggregation and classification of curves whose flows converge.

(Fig. 3(d)), using the exact same operators but with different metrics. As this example illustrates, SAL provides a vocabulary for expressing the knowledge required — distance metrics, similarity metrics, etc. — for uncovering multi-level structures in spatial data sets. It has been applied to applications ranging from decentralized control design [Bailey-Kellogg and Zhao, 1999; 2001] to analysis of diffusion-reaction morphogenesis [Ordóñez and Zhao, 2000].

3 Ambiguity-Directed Sampling

We extend SAL for data-scarce, rather than data-rich, applications, by focusing data collection in areas that will yield information most useful in discriminating among possible models. Given a set of possible SAL models $M = \{m_1, m_2, \dots, m_n\}$ for the data, we want to choose a new sample s to help discriminate among posterior probabilities $P(m_i | s)$. For instance, in the vector-bundling example (Fig. 3), models would represent different choices of how to group vectors into streamlines. By Bayes rule, we need to evaluate $P(s | m_i)$ and $P(m_i)$. The domain knowledge used to enumerate the possible SAL structures also places priors $P(m_i)$ on the identified models. In the vector-bundling example, a possible streamline can be scored based on how well its curvature matches the directions of the vectors it aggregates. Additional domain knowledge then characterizes the dependence of potential sample values on different models, thus helping to optimize the next sample location. As we will show later in this section, one useful form of such dependence relates to addressing *ambiguity*. For example, the best aggregation for ambiguous streamlines can be determined by sampling the

interpolate : field \times new_objects \times surrogate \rightarrow new_values Determine values for new objects based on values for nearby objects in field, according to surrogate function.
classify : objects \times equiv_predicate \rightarrow classes \times ambiguities Apply predicate to neighboring objects, partitioning them into equivalence classes and left-over ambiguous objects. Predicate is a function taking a pair of neighbors and returning one of {true, false, ambiguous}.
sample : objects \times ambiguities \times objective_fn \rightarrow new_objects Determine new objects to be sampled based on optimization of an objective function indicating information gain with respect to the ambiguities.

Table 1: Ambiguity-directed sampling operators.

flow near streamline “branch points.”

Tab. 1 summarizes the incorporation of domain knowledge in new SAL operators by our ambiguity-directed sampling framework. The data interpretation and sampling process proceeds as follows, starting from some initial sparse data. (1) Derive qualitative SAL structures from either the sparse data, or a dense dataset interpolated with a surrogate function. (2) Identify ambiguities arising in the structure formation process. (3) Target a sample point that will optimize the information gain with respect to these ambiguities. (4) Update the data set and repeat, as long as information gain is substantial enough. The following subsections describe the key parts of this approach in more detail.

3.1 Interpolation with a Surrogate Function

In some cases it is advantageous to generate a dense dataset and find structures in it, rather than to work directly from sparse data. For example, when possible models have a known, common structure (e.g. they can be treated as locally smooth quadratic functions), then interpolating dense data can simplify structure and ambiguity identification. The *interpolate* operator in Tab. 1 generates such dense data, according to a given surrogate representation.

The choice of surrogate representation is constrained by the local nature of SAL computations. For example, global, least-squares type approximations are inappropriate since measurements at all locations are equally considered to uncover trends and patterns in a particular region. We advocate the use of kriging-type interpolators [Sacks *et al.*, 1989], which are local modeling methods with roots in Bayesian statistics. Kriging can handle situations with multiple local extrema (for example, in weather data, remote sensing data, etc.) and can easily exploit anisotropies and trends. Given k observations, the interpolated model gives exact responses at these k sites and estimates values at other sites by minimizing the mean squared error (MSE), assuming a random data process of known functional form.

Formally (for two dimensions), the true function p is assumed to be the realization of a random process such as:

$$p(x, y) = \beta + Z(x, y) \quad (1)$$

where β is typically a uniform random variate, estimated based on the known k values of p , and Z is a correlation

function with zero mean and known variance. Kriging then estimates a model p' of the same form, based on the k observations:

$$p'(x_i, y_i) = E(p(x_i, y_i) | p(x_1, y_1), \dots, p(x_k, y_k)) \quad (2)$$

and minimizing mean squared error between p' and p :

$$MSE = E(p'(x, y) - p(x, y))^2 \quad (3)$$

A typical choice for Z in p' is $\sigma^2 R$, where scalar σ^2 is the *estimated* variance, and the symmetric correlation matrix R can encode domain-specific constraints and factors reflecting the current fidelity of data. We use an exponential function for entries in R , with two parameters C_1 and C_2 :

$$R_{ij} = e^{-C_1|x_i-x_j|^2 - C_2|y_i-y_j|^2} \quad (4)$$

Intuitively, function estimation at a given point is influenced more by observations nearby than by those farther away.

The estimator minimizing mean squared error is then obtained by multi-dimensional optimization:

$$\max_C \frac{-k}{2} (\ln \sigma^2 + \ln |R|) \quad (5)$$

This expression can be derived from the conditions that there is no error between the model and the true values at the chosen k sites, and that all variability in the model arises from the design of Z (the derivation is beyond the scope of this paper). The multi-dimensional optimization is often performed by gradient descent or pattern search methods. More details are available in [Sacks *et al.*, 1989], which demonstrates this methodology in the context of the design and analysis of computer experiments.

3.2 Bottom-Up Detection of Ambiguity

The SAL equivalence class clustering mechanism (operating on the sparse input data or the dense surrogate model) exploits continuity, grouping neighboring objects that satisfy a domain-specific equivalence predicate (e.g. similar vector direction). At discontinuities, dissimilar neighboring objects are placed in separate classes. However, within a weakly-similar class or across a weakly-different discontinuity, neighboring objects might *almost* satisfy the predicate. For example, some vectors in Fig. 3(b) have two possible forward neighbors; in some cases, a vector might equally well belong to either of two flows. We call such unclear classification choice points *ambiguous*.

The bottom-up SAL operators introduced in Sec. 2 can be used to detect ambiguities if the equivalence class clustering operator *classify* is extended as in Tab. 1. In particular, a domain-specific equivalence predicate indicates when neighbors are equivalent, not equivalent, or ambiguous, allowing *classify* to delay ambiguous classification decisions.

3.3 Top-Down Utilization of Ambiguity

Ambiguity can reflect the desirability of acquiring data at or near a specified point, to clarify the correct classification and to serve as a mathematical criterion of information content. The *sample* operator specified in Tab. 1 addresses this opportunity by generating samples to optimize a given

domain-specific objective function, given a set of ambiguous objects. For example, in response to a vector with an ambiguous neighbor, it might suggest nearby locations to sample. In other applications, it might pick the midpoint between a pair of ambiguous points, or even (see the influence-based model decomposition application below) apply SAL recursively to qualitatively analyze a set of ambiguous points. In conjunction with a surrogate function, some functional of the MSE (Eq. 3) can be used to focus sampling, by a suitable statistical design.¹ Section 4.2 describes the use of such an objective function.

When using a surrogate function, the correlation matrix R (Eq. 4) can be modified to emphasize the desirability of focusing the fitting effort on ambiguous regions. In particular, *indicator covariance terms* modulate R when the standard uniformly parameterized model (C_1 and C_2 in our case) does not adequately capture the observed variability. Our approach is reminiscent of incorporating “Type C soft data” into variogram estimation [Journel, 1986]: “soft” data have non-negligible uncertainty and “Type C” data are obtained without additional experimentation (in our case, via SAL analysis). By using the pcdf of ambiguous objects as an indicator covariance term, we can improve covariance estimates, and also help suggest data locations that will clarify the correct classification. The exact equations are beyond the scope of this article, but we refer the reader to [Journel, 1986] for an account of this “soft kriging” approach.

3.4 Iteration

Data are collected for the indicated sample points, by experiment or simulation. When a surrogate function is used, the fitted model is refined with real data at the indicated points, via *interpolate*. We note that efficient implementations of some data structures (e.g. Delaunay triangulation neighborhood graphs) can be incrementally updated with the additional samples [Ordóñez and Zhao, 2000]. The aggregation process can then be repeated with the extended data set, terminating when the information-theoretic metric used by *sample* drops below some specified level.

4 Applications

This section discusses how the computational framework of two existing applications can be redescribed in terms of ambiguity-directed sampling, and then illustrates the effectiveness of our approach with two new case studies.

4.1 Existing Applications

KAM [Yip, 1991] interprets the behaviors of Hamiltonian dynamical systems by phase-space analysis. Geometric points represent states of the system for a given set of parameters. KAM works directly with these samples — it does not *interpolate* a dense representation. KAM groups points into orbits describing the system’s temporal evolution; it groups orbits into phase portraits describing evolution of all states for a given set of parameters; and it groups phase portraits into bifurcation maps describing variations in portraits due to

¹Sample selection optimization is different from kriging interpolation optimization (Eq. 5), used to generate a dense data field.

variations in parameters. At each stage, KAM adds samples when it detects an inadequate description. In our vocabulary, the *classify* predicate clustering orbits into a phase portrait notices when two neighboring orbits cannot physically be adjacent; *sample* then starts orbit integration from the mid-point of an ambiguous pair of neighboring points. Similarly, in a bifurcation map additional phase portraits are generated for parameter values between those of ambiguous neighboring phase portraits.

STA [Ordóñez and Zhao, 2000] has been applied to build high-level descriptions of morphogenesis in diffusion-reaction systems by tracking aggregates of sample “floaters” that react to changes in the underlying field. In particular, floaters attempt to ensure an adequate sampling of the field (no interpolation is required), especially in high-gradient areas. They do this in a manner similar to the particle system of Witkin and Heckbert [1994], by repelling each other, splitting, and merging. In our vocabulary, the *classify* predicate bundling floaters in a region tests whether or not neighboring floaters are near enough relative to an energy metric measuring adequate representation of the region; *sample* simply splits one ambiguous floater into two adjacent floaters.

4.2 Pocket Identification

Our first application domain is motivated by research in spatial statistics [Journel, 1986; Sacks *et al.*, 1989] and multidisciplinary system design [Knill *et al.*, 1999]. Visualize the n -dimensional hypercube defined by $x_i \in [-1, 1], i = 1 \dots n$, with the n -sphere of radius 1 centered at the origin ($\sum x_i^2 \leq 1$) embedded inside it. Notice that the ratio of the volume of the cube (2^n) to that of the sphere ($\pi^{n/2}/(n/2)!$) grows unboundedly with n . In other words, the volume of a high-dimensional cube is concentrated in its corners (a counter-intuitive notion at first). Carl de Boer exploited this property to design a difficult-to-optimize function which assumes a *pocket* in each corner of the cube (Fig. 4), that is just outside the sphere [Rice, 1992]. Formally, it can be defined as:

$$\alpha(\mathbf{X}) = \cos \left(\sum_{i=1}^n 2^i \left(1 + \frac{x_i}{|x_i|} \right) \right) - 2 \quad (6)$$

$$\delta(\mathbf{X}) = \|\mathbf{X} - 0.5\mathbf{I}\| \quad (7)$$

$$p(\mathbf{X}) = \alpha(\mathbf{X})(1 - \delta^2(\mathbf{X})(3 - 2\delta(\mathbf{X}))) + 1 \quad (8)$$

where \mathbf{X} is the n -dimensional point (x_1, x_2, \dots, x_n) at which the pocket function p is evaluated, \mathbf{I} is the identity n -vector, and $\|\cdot\|$ is the L_2 norm.

It is easily seen that p has 2^n local minima; if n is large (say, 30, which means it will take more than half a million points to just represent the corners of the n -cube!), naive global optimization algorithms will require an unreasonable number of function evaluations. However, in real-world domains, significant structure exists and can often be exploited. A good example is the STAGE algorithm [Boyan and Moore, 2000], which intelligently selects starting points for local search algorithms. Our goals here are very different from global optimization: we wish to obtain a qualitative indication of the existence, number, and locations of pockets, using low-fidelity models and/or as few data points as possible. The

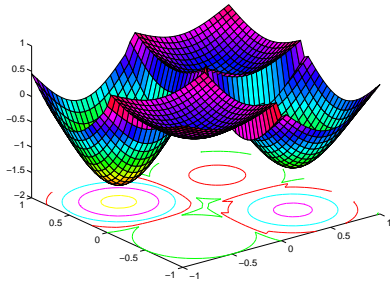


Figure 4: A 2D pocket function.

results can then be used to seed higher-fidelity calculations. This is also fundamentally different from DACE [Sacks *et al.*, 1989], polynomial response surface approximations [Knill *et al.*, 1999], and other approaches in geo-statistics where the goal is accuracy of functional prediction at untested data points. Here, accuracy of estimation is traded for the ability to mine pockets.

In a dense field of data, it is straightforward to identify pockets by applying the vector field bundling implementation discussed in the introduction (see Fig. 3) to the gradient field. In the data-scarce setting, we follow the ambiguity-directed sampling framework, incorporating the domain-specific knowledge summarized in Tab. 2. Given a surrogate model, vector bundling identifies vectors which can participate in multiple good streamlines. The surrogate model incorporates these ambiguities with an indicator covariance term counting the number of possible good neighbors. This “ambiguity distribution” provides a novel mechanism to include qualitative information — streamlines that agree will generally contribute less to data mining, just as samples that are far apart are weighted less in the original R matrix. Thus, this framework can be viewed as a natural generalization of the assumptions of sample clustering that underlie kriging.

The *sample* objective function described in Tab. 2 minimizes the expected posterior entropy on the *unsampled* design space, which by a reduction argument, can be shown to be maximizing the prior entropy over the *entire* design space [Sacks *et al.*, 1989]. In turn, this means that the amount of information obtained from an experiment is maximized. For our purposes, the objective function thus provides a basis to choose sample points that will improve our modeling of p .

We applied the ambiguity-driven mechanism to determining pockets in both 2D and 3D. We used a variation of the pocket function with a pseudorandom perturbation that shifts the pockets away from the corners in a somewhat unpredictable way. This twist precludes many forms of analyses, such as symbolic parsing, by imposing a highly nonlinear global map of pocket locations. In the traditional pocket function, the dips can be viewed as being influenced by little spheres at the corners, with known radii and centers. The new pocket design uses an additional parameter to impose non-symmetric perturbations which randomize both the radii and centers. As a result, local modeling must be carried out at each corner to determine the exact location of the pocket. More detail about this function can be found in [Rice, 1992,

Surrogate model

Use kriging interpolator with indicator covariance term (modeling number of similar-enough neighbors from predicate below) to estimate p at unknown points.

Vector equivalence predicate

Return true if vector directions are similar enough, false if they aren't, and ambiguous if a vector has multiple neighbors with similar-enough directions.

Sample objective function

Minimize the entropy $E(-\log d)$, where d is the conditional density of p over the design space *not covered* by the current data values.

Table 2: Domain knowledge for ambiguity-directed sampling in pocket identification.

pp. 113-114].

The initial experimental configuration used a face-centered design (4 points in the 2D case). The surrogate model then generated a 41^n -point grid. The ambiguity-directed mechanism selected new design points, using the vector field bundling approach discussed above. Standard parameter settings were applied: required similarity of 0.8 for dot product of adjacency direction and vector field direction, and factor of $0.01 \times \text{distance}$ penalizing the grouping of far-apart vectors.

Fig. 5 shows a design involving only 7 total data points that is able to mine the four pockets. As previously discussed, our sampling decisions result in highly sub-optimal designs according to traditional metrics of variance in predicted values and D-optimality, but are sufficient to determine pockets. In particular, the ambiguity-driven framework completely skips one of the quadrants in selecting new points. This indicates that neighborhood calculations involving the other three quadrants are enough to uncover the pocket in the fourth quadrant. Since the kriging interpolator uses local modeling and since pockets in 2D effectively occupy the quadrants, obtaining measurements at ambiguous locations serves to capture the relatively narrow regime of each dip, which in turn helps to distinguish the pocket in the neighboring quadrant. This effect is hard to achieve without qualitative feedback. For higher dimensions (including 3D), the pockets move further away from the center of the design space, necessitating the sampling of points in all corners.

Fig. 6 shows the distributions of number of design points required for ambiguity-directed and kriging-based pocket identification over 100 perturbed variations of the 2D pocket function. Ambiguity-directed sampling required 3 to 11 additional samples, with the latter figure in the pathological case where the random perturbations cause a nearly quintic dip, rendering the initial adjacency calculations misleading. In comparison, conventional incremental kriging techniques (of the form described in Section 3.1 without qualitative analysis) required 13 to 19 additional data points. While pockets in the bigger dips are discovered quickly, the quintic and shallow dips require more function evaluations. Tests with pockets in 3D yielded even more significant results: up to 151 additional points for regular kriging, but at most 42 for ambiguity-directed sampling. With the use of block kriging, reductions in both values could be enjoyed, but these figures illustrate

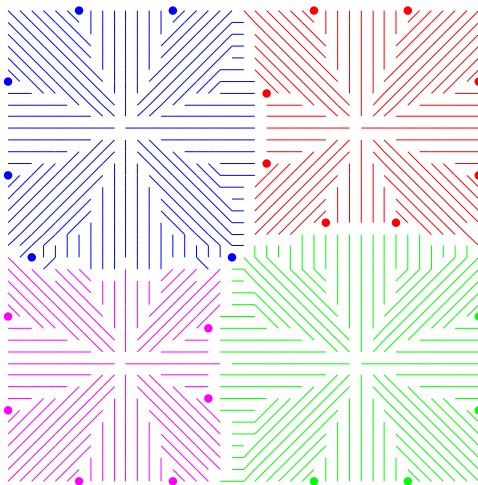
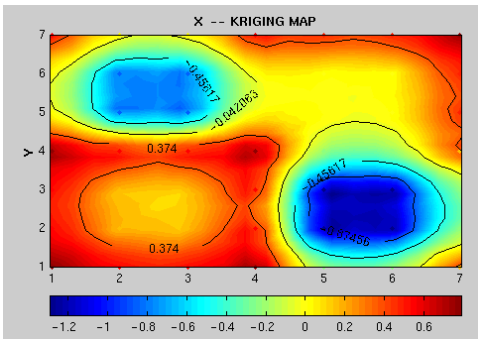
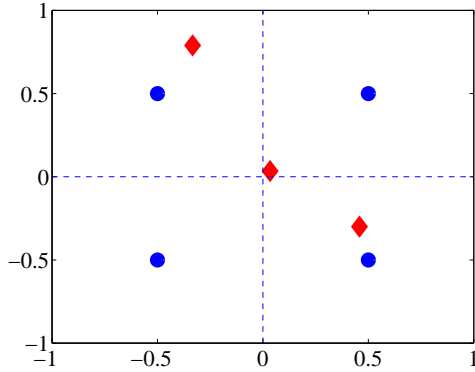


Figure 5: Mining pockets from only 7 sample points (2D). (top) The chosen sample locations: 4 initial face-centered samples (marked as blue circles) plus 3 ambiguity-directed samples (marked as red diamonds). Note that no additional sample is required in the lower-left quadrant. (middle) Computed variogram for resulting surrogate model: color represents estimated p and isocontours join points of equal estimated MSE. (bottom) SAL structures in surrogate model data, confirming the existence of four pockets.

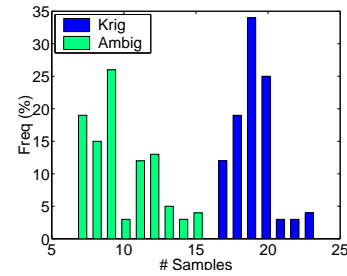


Figure 6: Pocket-finding results (2D) show that ambiguity-directed sampling always requires fewer total samples (7-15) than conventional kriging (17-23).

the effectiveness of our technique.

The extension to more than 3 dimensions is straightforward and is not detailed here for ease of presentation. It essentially entails using the appropriate covariance matrix and SAL data structures (e.g. 8-adjacency in 2D, 26-adjacency in 3D, ...). While we believe our ambiguity-directed framework will fare well compared to traditional kriging, a more careful study will be needed to characterize the scalability of our approach.

4.3 Influence-Based Model Decomposition

Influence-based model decomposition [Bailey-Kellogg and Zhao, 1999; 2001] is an approach to designing spatially-distributed data interpretation and decentralized control applications, such as thermal regulation for semiconductor wafer processing and noise control in photocopy machines. A decentralized *influence graph*, built by sampling the effects of controls on a field (either physically or by solving a partial differential equation), represents influences of controls on distributed physical fields. Given the expense of obtaining influence graph values, it is desirable to minimize the number of samples required. This section demonstrates that ambiguity-directed sampling can greatly reduce the number of samples required. Note that we do not *interpolate* a dense representation, following the explicit kriging methodology, since it sometimes does not result in explainable designs, by overlooking “nice” properties such as balance, symmetry, collapsibility, and comparability [Easterling, 1989].

Influence-based model decomposition uses influence graphs for control placement and parameter design algorithms that exploit physical knowledge of locality, linear superposability, and continuity for distributed systems with large numbers of coupled variables (often modeled by partial differential equations). By leveraging the physical knowledge encapsulated in influence graphs, these control design algorithms are more efficient than standard techniques, and produce designs explainable in terms of problem structures. Influence-based model decomposition decomposes a problem domain so as to allow relatively independent design of controls for the resulting regions. Fig. 7 overviews the approach:

1. Represent in an influence graph the effects of a few sample *probe* controls on the field — in this example, the heat flows induced in a piece of material by point heat sources.

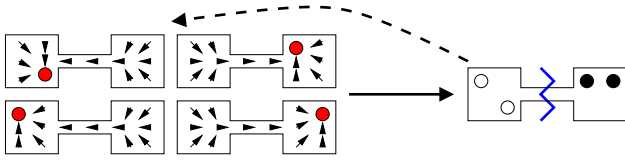


Figure 7: Influence-based model decomposition: sample an influence graph, and cluster probes and partition field based on similar control effects. Ambiguity-directed sampling techniques close the loop by suggesting new probe locations.

- Cluster the probes based on similarities in their effects, as represented in the influence graph. For example, the geometric constraint imposed by the narrow channel in the dumbbell-shaped piece of material results in similar field responses to the two probes in the left half of the dumbbell and similar responses to the two probes in the right half of the dumbbell. Note that influence graphs encapsulate not only geometry but also material properties, which can greatly impact heat flows and thus the proper decomposition.
- Cluster the field nodes based on the probe clustering, applying a predicate testing if neighboring field nodes are well-represented by the same probe nodes. In the example, the field nodes in the left half of the dumbbell are best represented by the probe nodes also in the left half (which belong to the same probe equivalence class), and are thus decomposed from the nodes in the right half. Controls are placed in the regions and optimized by a separate process not discussed here.

The quality of decompositions from a small number of randomly-placed probes is competitive with that of a spectral partitioning of the complete influence graph (computed following an approach developed for image segmentation [Shi and Malik, 1997]), but with orders of magnitude less computation and in a decentralized model [Bailey-Kellogg and Zhao, 2001]. We now extend this approach to show that replacing random sampling with ambiguity-directed sampling achieves even better results. Ambiguity-directed sampling effectively closes the loop between the field decomposition and influence graph sampling (dashed arrow in Fig. 7). Tab. 3 describes the domain-specific knowledge used in ambiguity-directed sampling for model-based decomposition.

We applied ambiguity-directed sampling to the three problems presented by [Bailey-Kellogg and Zhao, 2001]: a plus-shaped piece of material, a P-shaped piece of material, and an anisotropic bar, illustrating different geometries, topologies (the P-shaped material has a hole), and material properties. Results were collected for 1000 runs each by random probing, and using each possible node in the discretization for the initial probe in ambiguity-directed probing. Results are relative to a baseline spectral partitioning of the complete influence graph (computed essentially using probes at every one of the hundreds of nodes in a discretization).

Given a decomposition, a quality metric compares the amount of influence that stays within a region to the amount that leaves it: To be more specific, define the decomposition

Field node equivalence predicate

Return true if nodes have similar-enough effect to one probe, false if they don't, and ambiguous if the magnitude of the effect is not large enough or if two competing probes yield similar effects.

Sample objective function

Perform secondary aggregation and classification to find regions of ambiguities. For each ambiguous field node, measure how similar its flows are to other ambiguous field nodes in its region; choose the node with the best similarity to the most ambiguous nodes.

Table 3: Domain knowledge for ambiguity-directed sampling in influence-based model decomposition.

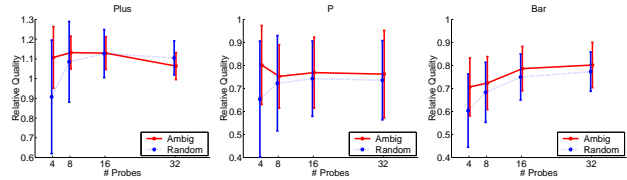


Figure 8: Comparison of influence-based model decomposition quality using random and ambiguity-directed probes, for three different problems: (left) plus; (middle) p; (right) bar. Results are relative to spectral partitioning.

quality q ($0 \leq q \leq 1$) for a partition P of a set of nodes S as follows (i is the influence):

$$q = \prod_{R \in P} \sum_{c \in R} \frac{\sum_{r \in R} i(c, r)}{\sum_{s \in S} i(c, s)}$$

Fig. 8 summarizes the results. The ambiguity-directed method generally does much better than random for a given number of probes, both in mean and standard deviation of quality, and it generally can do as well with 4-8 probes as random sampling can with 16-32. One interesting case is the taper in the plus-shaped piece of material. This is due to over-sampling: the samples are clustered in the middle of the plus, yielding a jagged decomposition that results in a worse quality score. In fact, with default parameters, the ambiguity-based metric declines to add samples beyond about 10, indicating that the field was adequately sampled. In order to achieve the desired number of samples, parameters were set to force sampling for only small information gain.

5 Discussion

The idea of selective sampling to satisfy particular design criteria arises in many contexts, such as Gaussian quadrature, spline smoothing in geometric design, remote sensing data acquisition, crystallography [Gopalakrishnan *et al.*, 2000] and engineering design optimization. In data mining, sampling has been viewed as a methodology to avoid costly disk accesses (this thread of research, however, doesn't address the issue of where to sample) [Kivinen and Mannila, 1994]. All these approaches (including ours) rely on capturing properties of a desirable design in terms of a novel objective function.

The distinguishing feature of our work is that it uses *spatial* information gleaned from a higher level of abstraction to focus data collection at the field/simulation code layer. While flavors of the *consistent labeling* problem in mobile vision have this feature, they are more attuned to transferring information across two *successive* abstraction levels. The applications presented here are novel in that they span and connect arbitrary levels of abstraction, thus suggesting new ways to integrate qualitative and quantitative simulation [Berleant and Kuipers, 1998].

The effectiveness of our approach relies on the trustworthiness of the ambiguity detection mechanism and the ability to act decisively on new information. In both our applications, this was easily achieved by relying on fairly specific qualitative features whose causes are well understood. However, in other applications (e.g. phase portrait exploration for sensitivity analysis of highly non-normal matrices), it is difficult to distinguish between qualitative changes in problem characteristic and numerical error such as roundoff. In such cases, a more detailed modeling of qualitative behavior should be exploited for ambiguity-directed sampling to be successful. In terms of the pocket study, this might require a domain-specific enumeration of the various ways in which pockets (and ambiguities in detecting them) can arise, and a probabilistic model of the elements of a SAL hierarchy using, say, superpositions of Bayesian expectation-maximization terms.

SAL provides a natural framework for exploiting *continuity* to uncover structures in spatial data; ambiguity-directed sampling focuses SAL's efforts on clarifying those *discontinuities* that yield multiple, qualitatively-different interpretations. This effort is leading us to explore a completely probabilistic SAL framework. Such a framework should also be able to incorporate information from multiple, perhaps conflicting, SAL hierarchies. This is an emerging frontier in several applications (such as bioinformatics), where diverse experimental methodologies can cause contradictory results at the highest levels of abstraction. Our work provides some encouraging results addressing such grand-challenge problems.

Acknowledgments

Thanks to Layne T. Watson (Virginia Tech) and Feng Zhao (Xerox PARC) for helpful discussions. This work is supported in part by the following grants to Bruce Randall Donald: National Science Foundation grants NSF IIS-9906790, NSF EIA-9901407, NSF EIA-9802068, NSF CDA-9726389, NSF EIA-9818299, NSF CISE/CDA-9805548, NSF IRI-9896020, and NSF IRI-9530785, U. S. Department of Justice contract 2000-DT-CX-K001, and an equipment grant from Microsoft Research; and NSF grant EIA-9984317 to Naren Ramakrishnan.

References

- [Bailey-Kellogg and Zhao, 1999] C. Bailey-Kellogg and F. Zhao. Influence-based model decomposition. In *Proc. AAAI*, 1999.
- [Bailey-Kellogg and Zhao, 2001] C. Bailey-Kellogg and F. Zhao. Influence-based model decomposition. *Artificial Intelligence*, 2001. Accepted, to appear.
- [Bailey-Kellogg *et al.*, 1996] C. Bailey-Kellogg, F. Zhao, and K. Yip. Spatial aggregation: language and applications. In *Proc. AAAI*, 1996.
- [Berleant and Kuipers, 1998] D. Berleant and B. Kuipers. Qualitative and quantitative simulation: bridging the gap. *Artificial Intelligence*, 95(2):215–255, 1998.
- [Boyan and Moore, 2000] J.A. Boyan and A.W. Moore. Learning evaluation functions to improve optimization by local search. *J. Machine Learning Research*, 1:77–112, 2000.
- [Easterling, 1989] R.G. Easterling. Comment on ‘Design and Analysis of Computer Experiments’. *Statistical Science*, 4(4):425–427, 1989.
- [Gopalakrishnan *et al.*, 2000] V. Gopalakrishnan, B.G. Buchanan, and J.M. Rosenberg. Intelligent aids for parallel experiment planning and macromolecular crystallization. In *Proc. ISMB*, volume 8, pages 171–182, 2000.
- [Journel, 1986] A. Journel. Constrained Interpolation and Qualitative Information - The Soft Kriging Approach. *Mathematical Geology*, 18(2):269–286, November 1986.
- [Kivinen and Mannila, 1994] J. Kivinen and H. Mannila. The use of sampling in knowledge discovery. In *Proc. 13th ACM Symposium on Principles of Database Systems*, pages 77–85, 1994.
- [Knill *et al.*, 1999] D.L. Knill, A.A. Giunta, C.A. Baker, B. Grossman, W.H. Mason, R.T. Haftka, and L.T. Watson. Response Surface Models Combining Linear and Euler Aerodynamics for Supersonic Transport Design. *J. of Aircraft*, 36(1):75–86, 1999.
- [Ordóñez and Zhao, 2000] I. Ordóñez and F. Zhao. STA: Spatio-temporal aggregation with applications to analysis of diffusion-reaction phenomena. In *Proc. AAAI*, 2000.
- [Rice, 1992] J.R. Rice. Learning, Teaching, Optimization and Approximation. In E.N. Houstis, J.R. Rice, and R. Vichnevetsky, editors, *Expert Systems for Scientific Computing*, pages 89–123. North-Holland, Amsterdam, 1992.
- [Sacks *et al.*, 1989] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409–435, 1989.
- [Shi and Malik, 1997] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. CVPR*, 1997.
- [Witkin and Heckbert, 1994] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. SIGGRAPH*, 1994.
- [Yip and Zhao, 1996] K.M. Yip and F. Zhao. Spatial aggregation: theory and applications. *J. Artificial Intelligence Research*, 5, 1996.
- [Yip, 1991] K.M. Yip. *KAM: A system for intelligently guiding numerical experimentation by computer*. MIT Press, 1991.

A Spatial Odyssey of the Interval Algebra: 1. Directed Intervals

Jochen Renz *

Institut für Informationssysteme
Technische Universität Wien
A-1040 Vienna, Austria

Abstract

Allen's well-known Interval Algebra has been developed for temporal representation and reasoning, but there are also interesting spatial applications where intervals can be used. A prototypical example are traffic scenarios where cars and their regions of influence can be represented as intervals on a road as the underlying line. There are several differences of temporal and spatial intervals which have to be considered when developing a spatial interval algebra. In this paper we analyze the first important difference: as opposed to temporal intervals, spatial intervals can have an intrinsic direction with respect to the underlying line. We develop an algebra for qualitative spatial representation and reasoning about directed intervals, identify tractable subsets, and show that path-consistency is sufficient for deciding consistency for a particular subset which contains all base relations.

1 Introduction

Qualitative spatial representation and reasoning has become more and more important in recent years. The best-known approach in this field is the Region Connection Calculus RCC8 [Randell *et al.*, 1992] which describes topological relationships between n -dimensional spatial regions of arbitrary shape. For some applications, however, it is sufficient to use spatial regions with more restricted properties. The block algebra [Balbiani *et al.*, 1999], for instance, considers only spatial regions which are n -dimensional blocks whose sides are parallel to the defining axes. The most restricted spatial regions are (one-dimensional) intervals. A prototypical spatial application of intervals are traffic scenarios. Vehicles usually move only along given ways (also sea-/airways). Therefore, when looking at vehicles on one particular way, vehicles and their regions of influence (such as safety margin, braking distance, or reaction distance) could be represented as intervals on a line which represents the possibly winded way. Similar

*This research was carried out while the author was visiting the Department of Computer and Information Science at the University of Linköping, Sweden. Supported by the Wallenberg foundation as part of the WITAS project. Thanks to Patrick Doherty, Guy Even, Alfonso Gerevini, and Erik Sandewall for their comments.

to the well-known Interval Algebra [Allen, 1983] developed for temporal intervals, it seems useful to develop a spatial interval algebra for spatial intervals.

There are several differences between spatial and temporal intervals which have to be considered when extending the Interval Algebra towards dealing with spatial applications. (1) spatial intervals can have different directions, either the same or the opposite direction as the underlying line. (2) ways usually have more than one lane where vehicles can move, i.e., it should be possible to represent that intervals are on different lanes and that one interval is, e.g., left of, right of, or beside another interval. (3) it is interesting to represent intervals on way networks instead of considering just isolated ways. (4) intervals such as those corresponding to regions of influence often depend on the speed of vehicles, i.e., it should be possible to represent dynamic information. This is also necessary for predicting future positions of vehicles which is an important task in traffic control. As for temporal intervals it is also important to represent qualitative or metric information on the length of intervals and on the distance between intervals.

We start this spatial odyssey of the Interval Algebra by analyzing the first important difference between spatial and temporal intervals, namely, direction of intervals. We define the directed intervals algebra which consists of 26 jointly exhaustive and pairwise disjoint *base relations*, identify tractable subsets, and show that path-consistency decides consistency for a particular subset which contains all base relations.

2 Directed Intervals

The Interval Algebra (IA) describes the possible relationships between convex intervals on a directed line. The default application of the Interval Algebra is temporal, so the directed line is usually considered to be the timeline. The 13 IA base relations (before \prec , after \succ , meets m , met-by m_i , overlaps o , overlapped-by o_i , equals \equiv , during d , includes d_i , starts s , started-by s_i , finishes f , and finished-by f_i) describe a combination of topological relations (disconnected, externally connected, partial overlap, equal, non-tangential proper part, tangential proper part, and the converse of the latter two) and order relations (\prec, \succ). The topological distinctions are exactly those which are made by RCC8. Therefore, RCC8 is often considered as the spatial counterpart of the Interval Algebra. Or, from another point of view, what distinguishes the Interval Algebra from RCC8 and what makes it its key

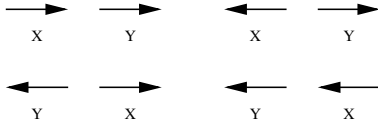


Figure 1: Four structurally different instantiations of the relation “ x behind y ” with directed intervals

feature is the given direction of the (one-dimensional) line. This given direction naturally imposes a direction also on the intervals: an interval can have the same or the opposite direction as the underlying line. However, because of its original temporal interpretation (no event can end before it starts), direction of intervals has never been considered in AI. Actually, directed intervals have been studied in the large field of Interval Arithmetics, but work in this field is completely different from the qualitative and constraint-based approaches studied in AI. When using the Interval Algebra for spatial applications, direction of intervals has to be taken into account. This leads to the obvious question: can the large body of work and the large number of results obtained on the Interval Algebra such as algorithms and complexity results also be applied to a spatial interpretation of the Interval Algebra, or is it necessary to completely start from scratch again?

Before answering this question, consider the example of Figure 1 which illustrates the differences of having directed intervals from having only intervals of the same direction. Since all four combinations of the directions of the two intervals are possible, there are four structurally different instantiations of every relation instead of just one. Therefore, it is possible that inconsistent instances of the Interval Algebra become consistent when allowing directed intervals.

3 The Directed Intervals Algebra

A straightforward way for dealing with directed intervals would be to add additional constraints on the direction of intervals to constraints over the Interval Algebra and treat the two types of constraints separately while propagating information from one type to the other (similar to what has been done in [Gerevini and Renz, 1998].) We say that an interval has *positive direction* if it has the same direction as the underlying line and *negative direction* otherwise. So possible direction constraints could be unary constraints like “ x has positive/negative direction” or binary constraints like “ x and y have the same/opposite direction”. This approach, however, is not possible since the Interval Algebra loses its property of being a relation algebra when permitting directed intervals. This can be easily seen when considering the “behind” relation of Figure 1. The converse of “ x behind y ” is “ y is behind or in front of x ”, whose converse is “ x is behind or in front of y ”, i.e., applying the converse operation (\smile) twice leads to a different relation than the original relation. This is a contradiction to one of the requirements of relation algebras ($R^{\smile\smile} = R$) [Ladkin and Maddux, 1994]. This contradiction does not occur when we refine the “behind” relation into two disjoint sub-relations “behind₌” and “behind_≠” where the subscript indicates that both intervals have the same (=) or opposite (≠) direction. The converse of both relations is “in-front-of₌” and “behind_≠”, respectively. Applying the

Directed Intervals Base Relation	Sym- bol	Pictorial Example
x behind ₌ y	$b_{=}$	-x->
y in-front-of ₌ x	$f_{=}$	-y->
x behind _≠ y	$b_{≠}$	<-x- -y->
x in-front-of _≠ y	$f_{≠}$	-x-> <-y-
x meets-from-behind ₌ y	$mb_{=}$	-x->
y meets-in-the-front ₌ x	$mf_{=}$	-y->
x meets-from-behind _≠ y	$mb_{≠}$	<-x- -y->
x meets-in-the-front _≠ y	$mf_{≠}$	-x-> <-y-
x overlaps-from-behind ₌ y	$ob_{=}$	-x->
y overlaps-in-the-front ₌ x	$of_{=}$	-y->
x overlaps-from-behind _≠ y	$ob_{≠}$	<-x- -y->
x overlaps-in-the-front _≠ y	$of_{≠}$	-x-> <-y-
x contained-in ₌ y	$c_{=}$	-x->
y extends ₌ x	$e_{=}$	-y->
x contained-in _≠ y	$c_{≠}$	<-x- -y->
y extends _≠ x	$e_{≠}$	-y->
x contained-in-the-back-of ₌ y	$cb_{=}$	-x->
y extends-the-front-of ₌ x	$ef_{=}$	-y->
x contained-in-the-back-of _≠ y	$cb_{≠}$	<-x- -y->
y extends-the-back-of _≠ x	$eb_{≠}$	-y->
x contained-in-the-front-of ₌ y	$cf_{=}$	-x->
y extends-the-back-of ₌ x	$eb_{=}$	-y->
x contained-in-the-front-of _≠ y	$cf_{≠}$	<-x- -y->
y extends-the-front-of _≠ x	$ef_{≠}$	-y->
x equals ₌ y	$eq_{=}$	-x-> -y->
x equals _≠ y	$eq_{≠}$	-x-> <-y-

Table 1: The 26 base relations of the directed intervals algebra

converse operation again leads to the original relations.

Since a relation algebra must be closed under composition, intersection, and converse, we have to make the same distinction also for all other IA relations. This leads us to the definition of the directed intervals algebra (DIA). It consists of the 26 base relations given in Table 1, which result from refining each IA relation into two sub-relations specifying either same or opposite direction of the involved intervals, and of all possible unions of the base relations. This gives a total number of 2^{26} DIA relations. Converse relations are given in the same table entry. If a converse relation is not explicitly given, the corresponding relation is its own converse. We denote the set of 26 DIA base relations as \mathcal{B} . Then $\text{DIA} = 2^{\mathcal{B}}$. Complex relations which are the union of more than one base relation R_1, \dots, R_k are written as $\{R_1, \dots, R_k\}$. The union of all base relations, the universal relation, is denoted $\{*\}$.

A DIA base relation $R = I_d$ consist of two parts, the interval part I which is a spatial interpretation of the Interval Algebra and the direction part d which gives the mutual direction of both intervals, either = or \neq . If a complex relation R consist of base relations with the same direction part d , we can combine the interval parts and write $R = \{I^1, \dots, I^k\}_d$ instead of $R = \{I_d^1, \dots, I_d^k\}$. We write R_e (resp. R_n) in

R	\prec	\succ	m	mi	o	oi	s	si	d	di	f	fi	\equiv
R^r	\succ	\prec	mi	m	oi	o	f	fi	d	di	s	si	\equiv
$\text{dia}(R)$	b	f	mb	mf	ob	of	cb	ef	c	e	cf	eb	eq

Table 2: IA base relations R , their reverses R^r , and their spatial interpretations $\text{dia}(R)$

order to refer to the union of the interval parts of every sub-relation of a complex relation R where the direction part is $\{=\}$ (resp. $\{\neq\}$.) In this way, every DIA relation R can be written as $R = \{R_e\}_= \cup \{R_n\}_{\neq}$. DIA_I denotes the set of 2^{13} possible interval parts of DIA relations.

It is important to note that the spatial interpretation of the Interval Algebra was chosen in a way that the interval part of a relation $xI_d y$ only depends on the direction of y and not on the direction of x . Therefore, if the direction of x is reversed, written as \bar{x} , then only the direction part changes, i.e., $xI_d y = \bar{x}I_{-d} y$. This would not be the case in a straightforward spatial interpretation of the original temporal relations. For instance, IA relations like “ x started-by y ” or “ x finished-by y ” depend on the direction of x . Instead, we interpret these relations spatially as “ x extends-the-front/back-of y ” and “ x contained-in-the-front/back-of y ”. This interpretation is independent of the direction of x . When all intervals have the same direction, both interpretations are equivalent. In order to transform the spatial and the temporal interval relations (independent of the direction of the intervals) into each other, we introduce two mutually inverse functions $\text{dia} : \text{IA} \rightarrow \text{DIA}_I$ and $\text{ia} : \text{DIA}_I \rightarrow \text{IA}$, i.e., $\text{dia}(\text{ia}(R)) = R$ and $\text{ia}(\text{dia}(R)) = R$. The mapping is given in Table 2.

All relations of the directed intervals algebra are invariant with respect to the direction of the underlying line, i.e., when reversing the direction of the line, all relations remain the same. This is obviously not the case for the Interval Algebra, e.g., if x is before y and one reverses the direction of the timeline, then x is after y . In order to transform DIA relations into the corresponding IA relations and *vice versa*, we introduce a unary *reverse* operator (\cdot^r) on relations R such that R^r specifies the relation which results from R when reversing the direction of the underlying line. For all relations $R \in \text{DIA}$ we have that $R^r = R$. For IA relations, the reverse relation is given in Table 2. The reverse of a complex relation is the union of the reverses of the involved base relations. The reverse of the composition (\circ) of two relations is equivalent to the composition of the reverses of the two involved relations, i.e., $(R \circ S)^r = R^r \circ S^r$. Applying the reverse operator twice results in the original relation, i.e., $R^{rr} = R$. Using the reverse operator we can also specify what happens with a relation $xI_d y$ if only the direction of y is changed. Then the topological relation of the intervals stays the same, but the order changes, i.e., “front” becomes “behind”/“back” and *vice versa*. The mutual direction also changes. This can be expressed in the following way: $xI_d y = x \text{dia}(\text{ia}(I)^r)_{-d} \bar{y}$.

We now have all requirements for computing the composition (\circ) of DIA relations using composition of IA relations (denoted here by \circ_{ia}) as specified by Allen [1983].

Theorem 3.1 *Let R_p, S_q be DIA base relations.*

1. If $q = \{=\}$, then $R_p \circ S_q = \text{dia}(\text{ia}(R) \circ_{ia} \text{ia}(S))_p$
2. If $q = \{\neq\}$, then $R_p \circ S_q = \text{dia}(\text{ia}(R)^r \circ_{ia} \text{ia}(S))_{\neg p}$

Proof. Assume that $xR_p y$ and $yS_q z$ holds. If $p = q = \{=\}$ and x, y, z have positive direction, it is clear that the interval part of the composition of the DIA relations is the same as the composition of the IA relations (with respect to the different interpretations.) The result of the composition is the same if x, y, z have negative direction, since DIA relations are invariant with respect to the direction of the underlying line. R only depends on the direction of y and S only depends on the direction of z . Therefore, reversing the direction of x (i.e., $p = \{\neq\}, q = \{=\}$) does not change the result of the interval part of the composition, only the resulting direction part. This proves the first rule.

Assume that $p = q = \{\neq\}$ and x, z have positive direction while y has negative direction. If we reverse the direction of y , which changes the relations to $x \text{dia}(\text{ia}(R)^r)_{=} \bar{y}$ and to $\bar{y}S_{=} z$, then we can apply the first composition rule. This results in $R_{\neq} \circ S_{=} = \text{dia}(\text{ia}(R)^r)_{=} \circ S_{=} =_{(1.)} \text{dia}(\text{ia}(\text{dia}(\text{ia}(R)^r)) \circ_{ia} \text{ia}(S))_{=} = \text{dia}(\text{ia}(R)^r \circ_{ia} \text{ia}(S))_{=}$, the second composition rule. As in the first case, this rule does not change when we reverse the direction of x (i.e., $p = \{=\}, q = \{\neq\}$) or the direction of all three intervals. This proves the second rule. ■

The composition of complex relations is as usual the union of the composition of the contained base relations. It follows from the closedness of the Interval Algebra that DIA is closed under composition, intersection, converse, and reverse.

4 Reasoning over Directed Intervals

The main reasoning problem in spatial and temporal reasoning is the consistency problem $\text{CSPSAT}(S)$ where S is a set of relations over a relation algebra [Renz and Nebel, 1999].

Instance: A set \mathcal{V} of variables over a domain \mathcal{D} and a finite set Θ of binary constraints xRy ($R \in \mathcal{S}$ and $x, y \in \mathcal{V}$.)

Question: Is there a consistent instantiation of all n variables in Θ with values from \mathcal{D} which satisfies all constraints?

The consistency problem of the directed intervals algebra, $\text{CSPSAT}(\text{DIA})$, is clearly NP-hard since the consistency problem of the Interval Algebra is already NP-hard. On the other hand it is not clear whether the consistency problem is tractable if only the DIA base relations are used.

Additional to the DIA relations, we also give the possibility of explicitly specifying the direction of intervals. We maintain them in a set Δ which contains unary direction constraints of the form (x, d) where x is a variable over a directed interval and $d \subseteq \{+, -\}$ gives the direction of x , either positive $\{+\}$, negative $\{-\}$, or indefinite $\{+, -\}$. Unary direction constraints and DIA constraints interact in two ways.

Proposition 4.1 *Given two intervals x, y , the DIA constraint xRy with $R = \{R_{m_1}^1, \dots, R_{m_k}^k\}$, and the unary direction constraints (x, d_1) and (y, d_2) . These constraints interact in the following way:*

1. If all m_i ($i = 1 \dots k$) are equivalent, then (a) $d_1 = d_1 \cap d_2$ and $d_2 = d_1 \cap d_2$ if $m_1 = \{=\}$ and (b) $d_1 = d_1 \cap \neg d_2$ and $d_2 = \neg d_1 \cap d_2$ if $m_1 = \{\neq\}$.
2. If d_1 and d_2 are both definite, then (a) $R = \{R_e\}_=$ if $d_1 = d_2$ and (b) $R = \{R_n\}_{\neq}$ if $d_1 \neq d_2$.

If all information is propagated from Θ to Δ and from Δ to Θ we write the resulting sets as Θ_Δ and Δ_Θ . If the empty constraint occurs during this propagation, then Θ is inconsistent.

There are several ways of deciding consistency of a given set of constraints over a set of relations \mathcal{S} . The most common way is to use backtracking over a tractable subset of \mathcal{S} which contains all base relations and enforce *path-consistency* as forward-checking (this is done by applying for each triple of constraints xRy, ySz, xTz the operation $T := T \cap (R \circ S)$; if the empty relation is not contained, the resulting set is path-consistent) [Ladkin and Reinefeld, 1997]. Before we can use this method for deciding CSPSAT(DIA), we must prove that the consistency problem is tractable for the DIA base relations and preferably that path-consistency decides consistency for these relations. In order to prove this, we need a different method for deciding consistency and we have to show that this method is polynomial for the set of DIA base relations.

For the Interval Algebra most tractability proofs were carried out using the endpoint encoding of the IA relations (e.g. [Nebel and Bürckert, 1995]) which describes the qualitative relations between the four endpoints of the two involved intervals. For instance, the “before” relation can be encoded as $X_e < Y_s$ plus the default relations $X_s < X_e$ and $Y_s < Y_e$ which hold for all non-directed intervals (X_s, Y_s denote the start points and X_e, Y_e the end points of the intervals X, Y .) It is also possible to specify an endpoint encoding of the DIA relations. Since spatial intervals can have different directions, the default relations do not hold anymore. Furthermore, we have to take into consideration that DIA relations are invariant with respect to the reverse operation. Therefore, it is the most compact way to use the “betweenness” predicate for specifying an endpoint encoding of DIA relations. *between*(X_e, X_s, Y_s) means that X_s is between X_e and Y_s , no matter which direction the intervals have. Using this predicate, the relation b_{\neq} , for instance, can be encoded as *between*(X_s, Y_s, Y_e) \wedge *between*(X_e, X_s, Y_s). Since the BETWEENNESS problem is NP-hard [Garey and Johnson, 1979], this encoding does not seem to be helpful for proving any tractability results. We will therefore refrain from specifying the endpoint formulas of the DIA base relations.

Another possibility of deciding the DIA consistency problem is to transform a set of DIA constraints Θ into an equivalent set of IA constraints Θ' and decide consistency of Θ' . In order to make such a transformation, the direction of every interval must be known. Then it is possible to reverse the direction of certain intervals such that all intervals have the same direction and transform the updated DIA constraints into IA constraints. We call this the *normal form* of a set of DIA constraints Θ and a set of definite unary direction constraints Δ for each interval involved in Θ . The normal form (written as $\text{nf}(\Theta, \Delta)$) is obtained as follows.

Proposition 4.2 *Given a set of DIA constraints Θ and a set Δ of definite unary direction constraints for each interval involved in Θ . The normal form $\text{nf}(\Theta, \Delta)$ is obtained by applying the following procedure.*

1. For each constraint $xR_d y \in \Theta_\Delta$ do
 2. If y has negative direction, add $x \text{ ia}(R)^r y$ to $\text{nf}(\Theta, \Delta)$
 3. If y has positive direction, add $x \text{ ia}(R) y$ to $\text{nf}(\Theta, \Delta)$

Lemma 4.3 *Given a set of DIA constraints Θ and a set Δ of definite unary direction constraints for each interval involved in Θ . $\text{nf}(\Theta, \Delta)$ can be computed in time $O(n^2)$.*

Proof. Θ_Δ can be computed in time $O(n^2)$, since all constraints of Δ are definite and information has to be propagated only from every pair of intervals to the corresponding constraint in Θ using rule 2 of Proposition 4.1. Θ_Δ is transformed to $\text{nf}(\Theta, \Delta)$ in time $O(n^2)$, since each of the $O(n^2)$ constraints is transformed separately in constant time. ■

Lemma 4.4 *Given a set of DIA constraints Θ and a set Δ of definite unary direction constraints for each interval involved in Θ . Θ_Δ is consistent if and only if $\text{nf}(\Theta, \Delta)$ is consistent.*

Proof. Suppose that Θ_Δ is consistent and that \mathcal{I} is an instantiation of Θ_Δ . The direction of each interval of \mathcal{I} is as specified in Δ and the relation between each pair of intervals x, y is a base relation R'_d which is a sub-relation of $xR_d y \in \Theta_\Delta$. We can now reverse the direction of all intervals of \mathcal{I} with negative directions, resulting in \mathcal{I}^+ . Since all DIA relations xRy only depend on the direction of y , the relations between the intervals of \mathcal{I}^+ are now $x \text{ dia}(\text{ia}(R')^r)_= y$ if the direction of y was negative in \mathcal{I} and $xR'_= y$ if the direction of y was positive in \mathcal{I} . Transforming these relations into IA relations results for every pair of intervals in sub-relations of $\text{nf}(\Theta, \Delta)$. Thus, \mathcal{I}^+ is a consistent instantiation of $\text{nf}(\Theta, \Delta)$. The opposite direction can be proved similarly. Suppose that $\text{nf}(\Theta, \Delta)$ is consistent and that \mathcal{J} is an instantiation of it where all intervals are considered to have positive direction. Let Θ^+ be the set of constraints between all intervals of \mathcal{J} using DIA base relations. Reversing the direction of all intervals which must have negative direction according to Δ results in \mathcal{J}^\pm and adopting the constraints of Θ^+ results in Θ^\pm . Since applying the reverse operator twice gives the original relation, each constraint of Θ^\pm is a sub-constraint of a constraint of Θ_Δ . Thus, \mathcal{J}^\pm is a consistent instantiation of Θ_Δ . ■

Using the normal form, we can now decide consistency of a set of DIA constraints Θ by computing or guessing a set Δ containing the direction of all intervals, computing $\text{nf}(\Theta, \Delta)$, and deciding consistency of $\text{nf}(\Theta, \Delta)$ using the methods developed for the Interval Algebra. Since there are 2^n different direction combinations of n directed intervals, it is in general NP-hard to find a suitable set Δ for which $\text{nf}(\Theta, \Delta)$ is consistent or to show that there is no such set. If, however, we can show that for a given set \mathcal{S} of DIA relations all possible candidate sets Δ can be identified in polynomial time and if $\text{nf}(\Theta, \Delta)$ contains only relations of a tractable subset of the Interval Algebra, then CSPSAT(\mathcal{S}) is tractable. Using this method, we identify several tractable subsets of the directed intervals algebra in the following section.

5 Tractable Subsets of DIA

The first set we analyze is the set of DIA base relations \mathcal{B} .

Lemma 5.1 *CSPSAT($\mathcal{B} \cup \{*\}$) is tractable.*

Proof. Consistency of a set Θ of constraints over $\mathcal{B} \cup \{*\}$ can be decided in polynomial time by using the following steps.

1. Transform Θ into a graph $G_\Theta = (V, E)$ where V is the set of variables involved in Θ and E contains an (undirected) edge (x, y) if $xRy \in \Theta$ where $R \in \mathcal{B}$.

2. Split V into disjoint subsets $V = V_1 \cup \dots \cup V_k$ such that for each pair of variables $x, y \in V_i$ there is a path from x to y in E and for each pair of variables $x \in V_i, y \in V_j$ ($i \neq j$) there is no path from x to y in E .
3. Generate a set of direction constraints Δ by selecting one variable x_i for each V_i and adding $(x_i, \{+\})$ to Δ .
4. Compute $\text{nf}(\Theta, \Delta_\Theta)$ and decide its consistency.

It is clear that each of the four steps can be computed in polynomial time. For each pair of variables of different sets V_i, V_j there are only constraints involving the universal relation, i.e., $\Theta(V_i)$ and $\Theta(V_j)$ which specify subsets of Θ containing all constraints involving only variables of V_i or V_j , respectively, are completely independent of each other. There is a path from each variable of V_i to every other variable of V_i and each path consists of constraints where each constraint involves only DIA relations with the same direction part. Therefore, it is sufficient to have the direction of only one variable $x_i \in V_i$ given in order to compute the direction of all variables $y \in V_i$. If the path contains an odd number of constraints involving DIA relations of the type R_{\neq} , the direction of y is opposite to the direction of x_i . Otherwise they have the same direction. Thus, Δ_Θ contains definite unary direction constraints for all variables of Θ . If there are conflicting paths, then Δ_Θ is inconsistent. Since DIA relations are invariant with respect to changing the direction of the underlying line, it does not matter for consistency purposes if we select the direction of x as positive or negative. $\text{nf}(\Theta, \Delta_\Theta)$ contains only relations of a tractable subset of IA. It follows from Lemma 4.4 that its consistency is equivalent to the consistency of Θ . ■

In the above proof it is not important that all non-universal relations are base relations, only that all non-universal relations consist of DIA base relations with the same direction part. Therefore, we can easily extend the above result.

Theorem 5.2 *Let \mathcal{S} be a tractable subset of the Interval Algebra which is closed under the reverse operator. Then $\mathcal{S}^\pm = \{\text{dia}(R)_= | R \in \mathcal{S}\} \cup \{\text{dia}(R)_\neq | R \in \mathcal{S}\} \cup \{*\}$ is a tractable subset of the directed intervals algebra.*

Proof. We can apply the same proof as given for Lemma 5.1. But only if all IA relations contained in the normal form are contained in a tractable subset of the Interval Algebra. This is clearly the case if \mathcal{S} is closed under the reverse operator which is used in the transformation into the normal form. ■

ORD-Horn (also denoted \mathcal{H}) is the only maximal tractable subset of the Interval Algebra which contains all IA base relations (and for which path-consistency decides consistency) [Nebel and Bürckert, 1995]. Using a machine-assisted comparison of the ORD-Horn relations we found that they are closed under the reverse operator. This is not true for some of the maximal tractable subclasses identified in [Drakengren and Jonsson, 1998] which do not contain all IA base relations.

All tractability results we have given so far rely on given mutual directions of intervals. For some applications this is a realistic assumption, but what happens if this is not given in all cases, if some constraints involve relations with different direction parts such as $x\{b_-, c_\neq, mb_\neq\}y$? Assume that we have given a set Θ of constraints over arbitrary DIA relations.

x_d	y_d	z_d	xRy	ySz	xTz
+	+	+	$\text{ia}(R_e)$	$\text{ia}(S_e)$	$\text{ia}(T_e)$
+	+	-	$\text{ia}(R_e)$	$\text{ia}(S_n)^r$	$\text{ia}(T_n)^r$
+	-	+	$\text{ia}(R_n)^r$	$\text{ia}(S_n)$	$\text{ia}(T_e)$
+	-	-	$\text{ia}(R_n)^r$	$\text{ia}(S_e)^r$	$\text{ia}(T_n)^r$
-	+	+	$\text{ia}(R_n)$	$\text{ia}(S_e)$	$\text{ia}(T_n)$
-	+	-	$\text{ia}(R_n)$	$\text{ia}(S_n)^r$	$\text{ia}(T_e)^r$
-	-	+	$\text{ia}(R_e)^r$	$\text{ia}(S_n)$	$\text{ia}(T_n)$
-	-	-	$\text{ia}(R_e)^r$	$\text{ia}(S_e)^r$	$\text{ia}(T_e)^r$

Table 3: Transformation of DIA constraints over a triple of variables x, y, z (depending on their directions x_d, y_d, z_d) into IA relations of the normal form.

One way of obtaining a possible candidate set Δ of definite unary directions for each interval in Θ is to look at each triple of variables (x, y, z) of Θ separately and check all 2^3 different combinations of directions (x_d, y_d, z_d) of (x, y, z) (x_d can be either + or -). If enforcing path-consistency to the normal form gives the empty relation for a particular choice of (x_d, y_d, z_d) , then this choice makes Θ inconsistent. If we combine all such inconsistent triples $t_i = \{(x, x_d), (y, y_d), (z, z_d)\}$, then Θ is inconsistent if $\Phi = \bigvee_i (l_x \wedge l_y \wedge l_z)$ is satisfiable (l_x is a placeholder for $\neg x_d$ if $(x, -) \in t_i$ and for x_d if $(x, +) \in t_i$, analogously for l_y and l_z .) A possible candidate set Δ can be obtained by computing a model of the complement of this formula, namely, $\Psi = \neg \Phi = \bigwedge_i (\neg l_x \vee \neg l_y \vee \neg l_z)$. This formula is an instance of 3SAT and, thus, NP-hard to compute. Eventually, since we did not propagate information between different triples, we have to check all possible models of Ψ .

Because of its NP hardness, this way of generating a candidate set Δ does not seem to be helpful. However, we can show that it leads to a tractability proof for a restricted but interesting set of DIA relations, namely, those DIA relations B_A which correspond to the set \mathcal{A} of 13 base relations of the Interval Algebra, i.e., $B_A = \{\{\text{dia}(R)_=, \text{dia}(R)_\neq\} | R \in \mathcal{A}\}$.

Theorem 5.3 *Let Θ be a set of DIA constraints over the variables x_1, \dots, x_n which contains a constraint $x_i R x_j$ with $R \in B_A$ if and only if $i < j$. Consistency of Θ can be decided in polynomial time.*

Proof. For each triple of variables x_i, x_j, x_k of Θ with $i \neq j \neq k, i < k$ we check for all 2^3 possible directions if the normal form of the triple is consistent or not. Depending on whether $i < j$, either $x_i R x_j$ or $x_j R x_i$ is given in Θ . Equivalently, either $x_j S x_k$ or $x_k S x_j$ is given in Θ . Since $i < k, x_i T x_k \in \Theta$. Obviously, it is not possible that $x_j R x_i$ and $x_k S x_j$ are both in Θ . Therefore, we compare in the normal form either (1) $R \circ S$, (2) $R^- \circ S$, or (3) $R \circ S^-$ with T in order to check consistency of the triple. In all three cases, the result is invariant with respect to the direction of one of the three variables and depends only on the direction of the other two variables. We can verify this using Table 3. In the first case, the only thing which changes in Table 3 when varying the direction of x_i (which is x_d in the table) are the subscripts of R and T : R_e changes to R_n , T_e changes to T_n and *vice versa*. Since all relations used in Θ are of the form $\{I_-, I_\neq\}$, I_e and I_n are always equivalent, i.e., the result of comparing $R \circ S$ with T is invariant with respect to the direction of x_i . In the second case, the result is invariant with

respect to the direction of x_j (which is y_d in the table.) When varying y_d in the table, S_e changes to S_n , $ia(R_e^-)$ changes to $ia(R_n^-)^r$, and $ia(R_n^-)$ changes to $ia(R_e^-)^r$ and *vice versa*. Since S_e is always equal to S_n , the change of S does not change the result. The change of R^- is more difficult. Using a case analysis of the 13 different cases, we were able to prove that $ia(R_e^-) = ia(R_n^-)^r$ and that $ia(R_n^-) = ia(R_e^-)^r$ if R is of type $\{I=, I\neq\}$. As an example consider the relation $R = \{b=, b\neq\}$ whose converse is $R^- = \{f=, b\neq\}$. $ia(f) = \{>\} = ia(b)^r$ and $ia(b) = \{<\} = ia(f)^r$. In the third case, the result is invariant with respect to the direction of x_i which can be proved equivalently to the first case.

Because consistency of every triple depends on the direction of only two of the three variables, the resulting formula Ψ (see above) is an instance of 2SAT and, thus, solvable in polynomial time. For any model of Ψ , the resulting set Δ of unary direction constraints leads to a consistent normal form $nf(\Theta, \Delta)$. This is because $nf(\Theta, \Delta)$ contains only IA base relations and because all triples are consistent (as it has been checked when Ψ was generated.) Therefore, enforcing path-consistency does not change any relation of $nf(\Theta, \Delta)$. ■

Instead of transforming every set Θ of DIA constraints to the normal form in order to decide consistency, it would be nice to know if and for which sets of DIA relations path-consistency is sufficient for deciding consistency when applied directly to Θ . We show this for \mathcal{H}^\pm , the set of DIA relations which results from ORD-Horn (see Theorem 5.2).

Theorem 5.4 *Path-consistency decides CSPSAT(\mathcal{H}^\pm).*

Proof. Consistency of a set Θ of DIA constraints over \mathcal{H}^\pm can be decided in polynomial time by deciding consistency of its normal form Θ' as it is obtained by applying the steps given in the proof of Lemma 5.1. Assume that Θ is path-consistent, i.e., for every triple of variables x, y, z with $xRy, ySz, xTz \in \Theta$ we have that $T \subseteq R \circ S$. If Θ' is also path-consistent, then Θ is consistent. In order to show that Θ' is path-consistent, we have to show that $T \subseteq R \circ S$ implies $T' \subseteq R' \circ_{ia} S'$ for all triples x, y, z ($xR'y, yS'z, xT'z \in \Theta'$.) Since all relations of \mathcal{H}^\pm consist of DIA base relations with the same direction part, we can extend the composition rules given in Theorem 3.1 to all \mathcal{H}^\pm relations. According to these rules, $T \subseteq R \circ S$ can be written as either $T_\neq \subseteq dia(ia(R_e)^r \circ_{ia} ia(S_n))_\neq$, $T_\neq \subseteq dia(ia(R_n) \circ_{ia} ia(S_e))_\neq$, $T_= \subseteq dia(ia(R_n)^r \circ_{ia} ia(S_n))_=$, or $T_= \subseteq dia(ia(R_e) \circ_{ia} ia(S_e))_=$ depending on the directions of x, y, z . The corresponding restrictions of T' can be derived using Table 3. If $z_d = \{+\}$, the restrictions of T' are equivalent to the results of applying ia to the interval parts of the above given restrictions of T . If $z_d = \{-\}$, we have to reverse T and the restrictions of T . Since $(U \circ_{ia} V)^r = U^r \circ_{ia} V^r$ and $U^{rr} = U$ for all IA relations U, V , the restrictions of T' are also equivalent to the results of applying ia to the interval parts of the restrictions of T (e.g. the first restriction $(ia(R_e)^r \circ_{ia} ia(S_n))^r = ia(R_e) \circ_{ia} ia(S_n)^r$ is the same as line 2 in Table 3.) Thus, $T \subseteq R \circ S$ implies $T' \subseteq R' \circ_{ia} S'$. ■

This result enables us to decide consistency of arbitrary sets Θ of DIA constraints by backtracking over the \mathcal{H}^\pm relations and by using path-consistency as a forward-checking method and as a decision procedure for sub-instances of Θ which contain only relations of \mathcal{H}^\pm .

6 Discussion & Future Work

We extended the Interval Algebra for dealing with directed intervals which occur when interpreting intervals as spatial instead of temporal entities. Reasoning over the directed intervals algebra DIA is more difficult than over the Interval Algebra IA, but it is possible to transform a set of DIA constraints into an equivalent set of IA constraints if the mutual directions of all intervals are known. This enabled us to transfer some tractability results from IA to DIA. If the mutual directions are not known, a tractable subset of DIA can be identified if two potentially exponential nested problems can be shown to be tractable for the subset: (a) compute a set of mutual directions and (b) decide consistency of the resulting set of constraints. We proved this for a small but interesting subset of DIA, but the problem is mostly open. Consequently, no maximal tractable subset of DIA has been identified so far.

DIA can also be used instead of IA as a basis for defining a block algebra [Balbiani *et al.*, 1999]. Then it is possible to reason about n -dimensional blocks with intrinsic directions. Remotely related to directed intervals are line segments with arbitrary directions which were analyzed by Moratz [2000].

The spatial odyssey of the Interval Algebra is to be continued as follows: (1) extend DIA to deal with intervals on parallel lines and on networks of lines, (2) add qualitative and metric information on the length of intervals and on the distance between intervals, and finally (3) extend the algebra to deal with dynamic instead of just static information, e.g., intervals move on lines with a certain velocity and sometimes switch to accessible lines. These are the desired properties of a calculus for representing and reasoning about traffic scenarios, a prototypical application of spatial intervals. Hopefully, contact with applications will be made before 2010...

References

- [Allen, 1983] J. F. Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26(11):832–843, 1983.
- [Balbiani *et al.*, 1999] P. Balbiani, J.-F. Condotta, and L. Farinas del Cerro. A tractable subclass of the block algebra: constraint propagation and preconvex relations. In *Proc. EPIA'99*, 1999.
- [Drakengren and Jonsson, 1998] T. Drakengren and P. Jonsson. A complete classification of tractability in Allen's algebra relative to subsets of basic relations. *AIJ*, 106(2):205–219, 1998.
- [Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA, 1979.
- [Gerevini and Renz, 1998] A. Gerevini and J. Renz. Combining topological and qualitative size constraints for spatial reasoning. In *Proc. CP'98*, 1998.
- [Ladkin and Maddux, 1994] P. B. Ladkin and R. Maddux. On binary constraint problems. *JACM*, 41(3):435–469, 1994.
- [Ladkin and Reinefeld, 1997] P. B. Ladkin and A. Reinefeld. Fast algebraic methods for interval constraint problems. *Annals of Mathematics and Artificial Intelligence*, 19:383–411, 1997.
- [Moratz *et al.*, 2000] R. Moratz, J. Renz, and D. Wolter. Qualitative spatial reasoning about line segments. In *Proc. ECAI'00*, 2000.
- [Nebel and Bürckert, 1995] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *JACM*, 42(1):43–66, 1995.
- [Randell *et al.*, 1992] D.A. Randell, Z. Cui, and A.G. Cohn. A spatial logic based on regions and connection. In *Proc. KR'92*, 1992.
- [Renz and Nebel, 1999] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. *AIJ*, 108(1-2):69–123, 1999.

From Images to Bodies: Modelling and Exploiting Spatial Occlusion and Motion Parallax

David Randell, Mark Witkowski and Murray Shanahan

Dept. of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
Exhibition Road
London SW7 2BT
United Kingdom
{d.randell, m.witkowski, m.shanahan}@ic.ac.uk

Abstract

This paper describes the Region Occlusion Calculus (*ROC-20*), that can be used to model spatial occlusion and the effects of motion parallax of arbitrary shaped objects. *ROC-20* assumes the region based ontology of *RCC-8* and extends Galton's *Lines of Sight* Calculus by allowing concave shaped objects into the modelled domain. This extension is used to describe the effects of mutually occluding bodies. The inclusion of van Benthem's axiomatisation of comparative nearness facilitates reasoning about relative distances between occluding bodies. Further, an envisionment table is developed to model sequences of occlusion events enabling reasoning about objects and their images formed in a changing visual field.

1 Introduction

Spatial occlusion (or interposition) arises when one object obscures the view of another. Spatial occlusion is one of several visual cues we exploit to build up our awareness of three-dimensional form and distance. Another is motion parallax, whereby a change in viewpoint causes relative displacements of objects at different distances in the visual field [Braddick and Atkinson, 1982]. Occlusion events help us determine where an object's boundary lies, or infer why an object cannot be seen, and what we need to do in order to render it visible.

For example, consider two objects A and B in an agent's visual field. Suppose the agent moves to its left, while keeping these objects in sight. If object A passes across B, or, when moving toward A, B becomes completely obscured, the agent can infer that A is in front of B. Similarly, if, when moving to the right, no relative change arises, the agent may infer that A and B are far away, or close by and possibly moving in the same direction as itself. Conversely, if A, when visible, always appears to be subtended by B, the agent may infer that A and B are physically connected. In each case, occlusion

events and motion parallax are being used to derive an objective model of the world from a naturally restricted viewpoint (Figure 1).

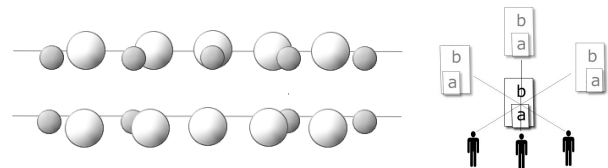


Figure 1: Spatial occlusion at work. Assuming a fixed viewpoint, in the two sequences shown on the left, the smaller ball passes in front of the larger one (top sequence) and behind it (bottom sequence). On the right, occlusion events arise with a change in viewpoint.

While visual occlusion remains a topic of some interest in the machine vision literature [e.g., Plantinga and Dyer, 1990; Geiger, *et al.*, 1995], an opportunity arises to investigate occlusion within the Qualitative Spatial Reasoning (QSR) domain. For example, Galton's [1994] *Lines of Sight* calculus outlines a theory of occlusion for modelling convex bodies using a discrete set of 14 occlusion relations. It is natural to take a topological approach to modelling occlusion, since occlusion events are very general, and apply to all objects irrespective of their size, shape and function. Petrov and Kuzmin [1996] provide an axiomatisation of spatial occlusion founded on a point-based ontology.

Randell, *et al.* [1992] develop a mereo-topological theory, *RCC-8*, used to describe spatial relationships between regions based on the primitive relation of connection. Cui, *et al.* [1992] use *RCC-8* to develop a qualitative simulation program to model physical processes by specifying direct topological transitions between these relations over time. Their work is one example of using qualitative spatial representations to model continuous change [Cohn, 1997]. *ROC-20* extends *RCC-8* to reason about relative distances between bodies from occlusion events, and transitions between occlusion events to model the effects of motion parallax from both object motion and changing viewpoints.

2 The Formal Theory

Our universe of discourse includes bodies, regions and points, all forming pairwise disjoint sets. In terms of interpretation, bodies denote physical objects, while regions split into two further disjoint sets that denote either three-dimensional volumes (typically the spaces occupied by bodies) or two-dimensional regions (typically projected *images* of bodies as seen from some viewpoint).

For the purposes of this paper, a set of sorts and a sorted logic are assumed. Within the sorted logic, possible values of variables in formulae are derived implicitly from the specified sort of the argument position in which it appears, allowing *ad hoc* polymorphic functions and predicates to be handled.

The notation and conventions used throughout this paper is as follows: **type** $a(\tau_1, \dots, \tau_n)$: τ_{n+1} means function symbol a is well sorted when its argument sorts are τ_1, \dots, τ_n with τ_{n+1} as the result sort, and **type** $a(\tau_1, \dots, \tau_n)$ means predicate a is well sorted when defined on argument sorts τ_1, \dots, τ_n . Axioms, definitions and theorems are respectively indicated in the text as follows: $(A1, \dots, A_n)$, $(D1, \dots, D_n)$, and $(T1, \dots, T_n)$. Where axiom/definitional schema are used, the numbering in the parentheses reflects the number of object-level axioms and definitions generated, e.g. $(A10-A15)$ would indicate that six axioms are defined.

2.1 RCC-8

The mereo-topological theory *RCC-8* [Randell, *et al.*, 1992] is embedded into *ROC-20*. As with *RCC-8*, the same primitive dyadic relation *C/2* is used: ' $C(x,y)$ ' is read as " x is connected with y " and is interpreted to mean that the topological closures of regions x and y share a point in common. All the relations defined in *RCC-8* are used, and all carry their usual readings: *DC/2* (disconnected), *P/2* (part), *EQ/2* (equal), *O/2* (overlaps), *DR/2* (discrete) *PO/2* (partial overlap), *EC/2* (external connection), *PP/2* (proper part), *TPP/2* (tangential proper part), *NTPP/2* (non-tangential proper part). *PI/2*, *PPI/2*, *TPPI/2* and *NTPI/2* are the inverse relations for *P/2*, *PP/2*, *TPP/2* and *NTPP/2*, respectively. Of these relations, eight are provably *Jointly Exhaustive and Pairwise Disjoint* (JEPD) and can be singled out for reasoning about state-state topological changes [Cui, *et al.*, 1992]. For brevity this set of relations is referred to as *JEPD^{RCC-8}*.

Axioms for *C/2* and definitions for the dyadic relations of *RCC-8* are as follows:

- (A1) $\forall x C(x,x)$
(A2) $\forall x \forall y [C(x,y) \rightarrow C(y,x)]$
- (D1) $DC(x,y) \equiv_{def} \neg C(x,y)$
(D2) $P(x,y) \equiv_{def} \forall z [C(z,x) \rightarrow C(z,y)]$
(D3) $EQ(x,y) \equiv_{def} P(x,y) \ \& \ P(y,x)$
(D4) $O(x,y) \equiv_{def} \exists z [P(z,x) \ \& \ P(z,y)]$
(D5) $DR(x,y) \equiv_{def} \neg O(x,y)$
(D6) $PO(x,y) \equiv_{def} O(x,y) \ \& \ \neg P(x,y) \ \& \ \neg P(y,x)$
(D7) $EC(x,y) \equiv_{def} C(x,y) \ \& \ \neg O(x,y)$
(D8) $PP(x,y) \equiv_{def} P(x,y) \ \& \ \neg P(y,x)$

- (D9) $TPP(x,y) \equiv_{def} PP(x,y) \ \& \ \exists z [EC(z,x) \ \& \ EC(z,y)]$
(D10) $NTPP(x,y) \equiv_{def} PP(x,y) \ \& \ \neg \exists z [EC(z,x) \ \& \ EC(z,y)]$
(D11) $PI(x,y) \equiv_{def} P(y,x)$
(D12) $PPI(x,y) \equiv_{def} PP(y,x)$
(D13) $TPPI(x,y) \equiv_{def} TPP(y,x)$
(D14) $NTPPI(x,y) \equiv_{def} NTPP(y,x)$

type $\Phi(Region, Region)$; where $\Phi \in \{C, DC, P, EQ, DR, PO, EC, PP, TPP, NTPP, PI, PPI, TPPI, NTPPI\}$

Not reproduced here, but assumed, is an axiom in *RCC-8* that guarantees every region has a nontangential proper part (A3), and a set of axioms (A4-A9) introducing Boolean functions for the sum, complement, product, difference of regions, and the universal spatial region, and an axiom that introduces the sort *Null* enabling partial functions to be handled – see [Randell, *et al.*, 1992].

2.2 Mapping Functions and Axioms

ROC-20 uses the set of dyadic relations from *RCC-8* to model the spatial relationship between bodies, volumes, and images. The distinction between bodies and regions is maintained by introducing two functions: '*region(x)*' read as "the region occupied by x " and '*image(x,v)*' read as "the image of x with respect to viewpoint v ". The function: *region/1*, maps a body to the volume of space its occupies, and *image/2* maps a body and a viewpoint to its image; i.e. the region defined by the set of projected half-lines originating at the viewpoint and intersecting the body, so forming part of the surface of a sphere of infinite radius centred on the viewpoint. A set of axioms incorporating these functions are defined by the following axiom schema¹:

- (A10-A15) $\forall x \forall y [\Phi(\text{region}(x), \text{region}(y)) \rightarrow \forall v [\Phi(\text{image}(x,v), \text{image}(y,v))]]$

type *region(Body):Region*²
type *image(Body, Point):Region*
type $\Phi(Region, Region)$ where $\Phi \in \{C, O, P, PP, NTPP, EQ\}$

Not all of the defined *RCC-8* relations are shown. For example, given $DC(\text{region}(a), \text{region}(b))$ all image relationships between the a and b are *possible* depending on the shape of the objects and the viewpoint assumed. This shows that these axioms function as a set of *spatial constraints* between bodies, a given viewpoint, and their corresponding images. This point is re-visited in section 6

¹ Although not developed here, the distinction made between bodies and regions enables one to define the notion of free space and model spatial occupancy – see [Shanahan, 1996].

² Sortal declarations given here are not as restricted as they could be, for example we could declare: **type** *region(Body):3DRegion*, and **type** *image(Body, Point):2DRegion*, where *2DRegion* and *3DRegion* are (disjoint) subsorts of the sort *Region*.

below, where a change in viewpoint, or a change in the relative positions of bodies with respect to a viewpoint, is discussed.

2.3 Occlusion Defined

A second primitive relation: ‘*TotallyOccludes*(x,y,v)’, read as “ x totally occludes y with respect to viewpoint v ”, is now added, and is axiomatised to be irreflexive and transitive (and is, by implication, asymmetric):

$$(A16) \quad \forall x \forall v \neg \text{TotallyOccludes}(x,x,v)$$

$$(A17) \quad \forall x \forall y \forall z \forall v [[\text{TotallyOccludes}(x,y,v) \ \& \ \text{TotallyOccludes}(y,z,v)] \rightarrow \text{TotallyOccludes}(x,z,v)]$$

type *TotallyOccludes*(*Body,Body,Point*)

The intended *geometric* meaning of total occlusion is as follows. Let *line*($p1,p2,p3$) mean that points $p1$, $p2$ and $p3$ fall on a straight line with $p2$ strictly between $p1$ and $p3$. Then, x totally occludes y from v iff for every point p in y , there exists a point q in x such that *line*(v,q,p), and there are no points p' in y , and q' in x , such that *line*(v,p',q'). Given the transitivity of total occlusion, an object x can totally occlude an object y even if x itself is totally occluded by another object.

Several axioms are now introduced to embed *RCC-8* into this theory:

$$(A18) \quad \forall x \forall y \forall z \forall v [[\text{TotallyOccludes}(x,y,v) \ \& \ P(\text{region}(z),\text{region}(y))] \rightarrow \text{TotallyOccludes}(x,z,v)]$$

i.e. if x totally occludes y , x totally occludes any part of y .

$$(A19) \quad \forall x \forall y \forall v [\text{TotallyOccludes}(x,y,v) \rightarrow \forall z [P(\text{region}(z),\text{region}(y))] \rightarrow \neg \text{TotallyOccludes}(z,x,v)]$$

i.e. if x totally occludes y no part of y totally occludes x .

$$(A20) \quad \forall x \forall y \forall v [\text{TotallyOccludes}(x,y,v) \rightarrow \forall z [[P(\text{region}(z),\text{region}(x)) \ \& \ P(\text{region}(u),\text{region}(y))] \rightarrow \neg \text{TotallyOccludes}(u,z,v)]]$$

i.e. if x totally occludes y no part of y totally occludes part of x .

This latter axiom excludes cases where the occluding body has parts that wrap ‘behind’ the occluding object. That is to say, while some nested bodies satisfy this relation, not all do, as in the case where, for example, a body is totally enveloped by another. This particular model is an example of *mutual occlusion*, which is defined below in definition (D17).

$$(A21) \quad \forall x \forall v \exists y \exists z [P(\text{region}(y),\text{region}(x)) \ \& \ P(\text{region}(z),\text{region}(x)) \ \& \ \text{TotallyOccludes}(y,z,v)]$$

i.e. every x has a part that totally occludes another part of x . This axiom guarantees that bodies have ‘depth’.

$$(A22) \quad \forall x \forall y \forall v [\text{TotallyOccludes}(x,y,v) \rightarrow P(\text{image}(y,v),\text{image}(x,v))]$$

i.e. if x totally occludes y , the image of x subtends the image of y . Note that (A22) is not a biconditional because the *P/2* relation does not take account of relative distance, a topic to be considered shortly.

By separating out volumes and images, two non-identical bodies having identical images (as in the case where one body exactly occludes another) can be modelled without inconsistency. Spatial identity in terms of co-location still applies, but is restricted to the dimensionality of the regions being modelled.

Next, the relation of occlusion is weakened to include, for example, partial occlusion: ‘*Occludes*(x,y,v)’ is read as “ x occludes y from viewpoint v ”:

$$(D15) \quad \text{Occludes}(x,y,v) \equiv \text{def.} \\ \exists z \exists u [P(\text{region}(z),\text{region}(x)) \ \& \ P(\text{region}(u),\text{region}(y)) \ \& \ \text{TotallyOccludes}(z,u,v)]$$

type *Occludes*(*Body,Body,Point*)

i.e. x occludes y if a part of x totally occludes a part of y .

Total occlusion between two objects implies occlusion, which in turn implies region overlap between their corresponding images:

$$(T1) \quad \forall x \forall y \forall v [\text{TotallyOccludes}(x,y,v) \rightarrow \text{Occludes}(x,y,v)] \\ (T2) \quad \forall x \forall y \forall v [\text{Occludes}(x,y,v) \rightarrow O(\text{image}(x,v),\text{image}(y,v))]$$

Occludes/3 is *non-symmetrical*. By contrast, the *O/2* relation in *RCC-8* is symmetrical, which renders it unsuitable for modelling occlusion relationships. Hence the need to augment *RCC-8* with an additional primitive relation.

Other more specific occlusion relations may now be defined: partial, and mutual occlusion. An example of mutual occlusion is two interlinked rings. These relations will then be finessed further by combining them with the set of *RCC-8* relations:

$$(D16) \quad \text{PartiallyOccludes}(x,y,v) \equiv \text{def.} \\ \text{Occludes}(x,y,v) \ \& \ \neg \text{TotallyOccludes}(x,y,v) \ \& \ \neg \text{Occludes}(y,x,v)$$

type *PartiallyOccludes*(*Body,Body,Point*)

i.e. x occludes (but does not totally occlude) y , but y does not occlude x .

(D17) $MutuallyOccludes(x,y,v) \equiv def.$
 $Occludes(x,y,v) \ \& \ Occludes(y,x,v)$

type $MutuallyOccludes(Body,Body,Point)$

i.e. x and y occlude each other.

For completeness (not listed here) inverse relations for $Occludes/3$, $TotallyOccludes/3$ and $PartiallyOccludes/3$ are defined (D18-D20); leaving the null case: $NonOccludes/3$, where no occlusion arises:

(D21) $NonOccludes(x,y,v) \equiv def.$
 $\neg Occludes(x,y,v) \ \& \ \neg Occludes(y,x,v)$

type $NonOccludes(Body,Body,Point)$

The six relations: $NonOccludes/3$, $MutuallyOccludes/3$; and $TotallyOccludes/3$, $PartiallyOccludes/3$, and their inverses are pairwise disjoint.

Finally, these new occlusion relations must be mapped to their RCC analogues:

(A23) $\forall x \forall y \forall v [NonOccludes(x,y,v) \rightarrow$
 $DR(image(x,v),image(y,v))]$

(A24) $\forall x \forall y \forall v [PartiallyOccludes(x,y,v) \rightarrow$
 $[PO(image(x,v),image(y,v)) \vee$
 $PP(image(x,v),image(y,v))]]]$

(A25) $\forall x \forall y \forall v [MutuallyOccludes(x,y,v) \rightarrow$
 $[PO(image(x,v),image(y,v)) \vee$
 $P(image(x,v),image(y,v)) \vee$
 $PI(image(x,v),image(y,v))]]]$

2.4 Finessing the Occlusion Relations

Although a variety of occlusion relations have now been defined, they are still very general, as no spatial relation stronger than $P/2$ from $RCC-8$ is used. Total occlusion, for example, covers three cases: (i) where the *image* of the occluded body is a tangential proper part of that of the occluding body, (ii) where it is a nontangential proper part, or (iii) the images are identical because one body exactly occludes the other. By refining the existing set of occlusion relations in this manner, a total set of 20 JEPD relations become definable. These are generated using the following definitional schemas:

(D22-D33) $\Phi\Psi(x,y,v) \equiv def.$
 $\Phi(x,y,v) \ \& \ \Psi(image(x,v),image(y,v))$

(D34-D41) $X\Psi^{-1}(x,y,v) \equiv def.$
 $X(y,x,v) \ \& \ \Psi(image(y,v),image(x,v))$

type $\Phi(Body,Body,Point)$

where if:

$\Phi = NonOccludes$, then $\Psi \in \{DC,EC\}$

$\Phi = TotallyOccludes$, then $\Psi \in \{EQ,TPPI,NTPPI\}$

$\Phi = PartiallyOccludes$, then $\Psi \in \{PO,TPP,NTPP\}$

$\Phi = MutuallyOccludes$, then $\Psi \in \{PO,EQ,TPP,NTPP\}$
and where if:

$X = TotallyOccludes$, then $\Psi \in \{EQ,TPPI,NTPPI\}$

$X = PartiallyOccludes$, then $\Psi \in \{PO,TPP,NTPP\}$

$X = MutuallyOccludes$, then $\Psi \in \{TPP,NTPP\}$

e.g. $TotallyOccludesEQ(x,y,v) \equiv def.$

$TotallyOccludes(x,y,v) \ \&$

$EQ(image(x,v),image(y,v))$

$TotallyOccludesEQ^{-1}(x,y,v) \equiv def.$

$TotallyOccludesEQ(y,x,v) \ \&$

$EQ(image(y,v),image(x,v))$


type $\Phi(Body,Body,Point)$ where: Φ is an element from the set of all 20 occlusion relations.

It is this part of the *Region Occlusion Calculus* that is now referred to as $ROC-20$, and the set of 20 JEPD relations as $JEPD^{ROC-20}$.

3 Theory Comparisons

It is now possible to map out the relationship between $RCC-8$, Galton's [1994] *Lines of Sight Calculus* ($LOS-14$), and $ROC-20$. Consider the $JEPD^{RCC-8}$ overlap relations first, i.e. $\{PO, TPP, NTPP, EQ, TPPI, NTPPI\}$. These relations are indifferent to relative distance with respect to a viewpoint, and each conflates a pair of Galton's relations. For example, given only that x partially overlaps y , it is impossible to say whether x is in front of or behind y . In both $LOS-14$ and $ROC-20$, these two cases are distinguished.

This leaves two $RCC-8$ relations $\{DC, EC\}$. These map respectively to the $LOS-14$ relations $C/2$ (clears) and JC (just clears), and to the two $ROC-20$ relations: $NonOccludesDC/3$ and $NonOccludesEC/3$. The six remaining relations of $ROC-20$ are precisely the cases where non-convex bodies (ruled out in $LOS-14$) are allowed into the modelled domain. These correspondences are illustrated in table 1.

In table 1 mutually occluding objects are shown thus, , indicating that the lighter coloured 'U'-shaped object both occludes and is occluded by the darker. In the special case of $MutuallyOccludesEQ/3$ part of the darker body lies behind the lighter body, and is exactly subtended by it, while a (visible) part of this, extends through a slot in the lighter body and occludes it.

The formal relationship between $LOS-14$ and $ROC-20$ is also illustrated by the following theorem:

(T3) $\forall x \forall y \forall v [\neg MutuallyOccludes(x,y,v) \leftrightarrow$
 $[NonOccludes(x,y,v) \vee$
 $TotallyOccludes(x,y,v) \vee$
 $PartiallyOccludes(x,y,v) \vee$
 $TotallyOccludes^{-1}(x,y,v) \vee$
 $PartiallyOccludes^{-1}(x,y,v)]]]$

where the five disjuncts are provably pairwise disjoint, and where each disjunct in turn respectively splits into $2+3+3+3+3 = 14$ of the *JEPD*^{ROC-20} ‘base’ relations.

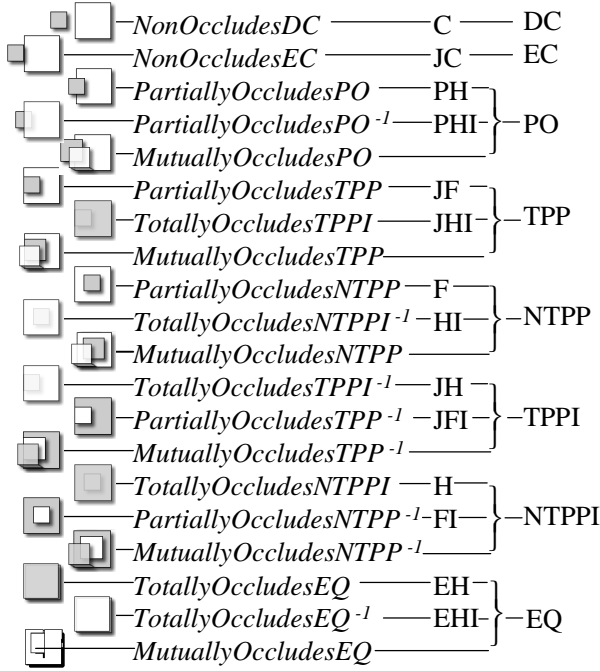


Table 1: The comparison between the *JEPD* relations of *ROC-20*, *LOS-14* and *RCC-8*. In each case the dark and light objects in the model respectively maps to the x, y variables of each $\Phi(x,y,v)$ relation of *ROC-20*, and to each corresponding $\Phi'(x,y)$ relation of *LOS-14* and *RCC-8*.

4 Comparative Distance and Occlusion

While the notion of relative distance between bodies appears in this theory, it only forms part of the *interpretation* resulting from the model used, and is implicit. Made explicit, a robot, for example, can exploit this information to reason about partial orderings of radial distances between itself and bodies based on their observed or inferred occlusion properties. A reworked subset of comparative distance axioms originally proposed by van Benthem [1982] is embedded into the theory. The primitive relation: ‘ $N(x,y,z)$ ’ used here, is read as “point x is nearer to body y than x is to body z ”, while ‘ $E(x,y,z)$ ’ is read as “body y is as near to point x as is body z ”:

$$(A26) \forall x \forall y \forall z \forall u [[N(x,y,z) \& N(x,z,u)] \rightarrow N(x,y,u)]$$

$$(A27) \forall x \forall y \neg N(x,y,y)$$

$$(A28) \forall x \forall y \forall z \forall u [N(x,y,z) \rightarrow [N(x,y,u) \vee N(x,u,z)]]$$

$$(D42) E(x,y,z) \equiv_{def} \neg N(x,y,z) \& \neg N(x,z,y)$$

type $\Phi(Point, Body, Body)$, where: $\Phi \in \{N, E\}$

Comparative distance is related to occlusion, and is embedded into *ROC-20*, with the following axioms:

$$(A29) \forall x \forall y \forall v [TotallyOccludes(x,y,v) \rightarrow N(v,x,y)]$$

i.e. if x totally occludes y with respect to some viewpoint v , then x is nearer to v than y is to v .

$$(A30) \forall x \forall y \forall v [N(v,x,y) \rightarrow \forall z [P(region(z), region(y)) \rightarrow N(v,x,z)]]$$

i.e. if v is nearer to x than y , then v is nearer to x than any part of y .

Note that the named viewpoint is not necessarily identified with an agent, and intentionally so. For example, if the agent holds and aligns two objects (one in each hand) where the one totally occludes the other, it does not follow the agent is closer to the occluding object, than the one occluded. It is also because of the guiding projective geometry assumed here (and which interprets the *image/2* function) that a viewpoint is identified with a point, and not an extended region in space.

5 Relative Orientation

If one body lies just to the left of another with respect to a line of sight, and is closer to the observer, movement to the right will typically increase the apparent separation between them. The relative left-right hand positions of the bodies will reverse as the line of sight intersects both bodies and passes to the left of that point. In order to be able to model and exploit this example of motion parallax, the ternary primitive relation: ‘ $Left(x,y,v)$ ’, read as “ x is to the left of y from viewpoint v ”, is added and axiomatized. Its dual (*Right/3*) is also defined:³

$$(A31) \forall x \forall v \neg Left(x,x,v)$$

$$(A32) \forall x \forall y \forall v [Left(x,y,v) \rightarrow \neg Left(y,x,v)]$$

$$(A33) \forall x \forall y \forall z \forall v [[Left(x,y,v) \& Left(y,z,v)] \rightarrow Left(x,z,v)]$$

$$(D43) Right(x,y,v) \equiv_{def} Left(y,x,v)$$

type $\Phi(Body, Body, Point)$ where: $\Phi \in \{Left, Right\}$

For completeness, the relation: ‘ $NonLeftRight(x,y,v)$ ’ read as “ x is neither to the left or right of y relative to viewpoint v ”, is added:

$$(D44) NonLeftRight(x,y,v) \equiv_{def} \neg Left(x,y,v) \& \neg Right(x,y,v)$$

type $NonLeftRight(Body, Body, Point)$

Here it is assumed that the observer’s horizon is fixed, and that the field of view is restricted. Without these assumptions, the transitivity of *Left/3*, for example, would fail in the intended model. This would be the case if the agent were at the centre of a circular arrangement of objects (Stonehenge,

³ Other spatial orientation duals with exactly the same properties (irreflexivity, etc.) are easily definable, e.g. forward/rearward, or above/below.

for example), entailing each object could be both to the left and the right of itself.

The primitive relation *Left/3* is embedded into the theory using the following axioms:

$$(A34) \forall x \forall y \forall v [Left(x,y,v) \rightarrow [\exists z [P(region(z),region(x)) \& Left(z,y,v)] \& \neg \exists u [P(region(u),region(x)) \& Left(y,u,v)]]]$$

$$(A35) \forall x \forall y \forall v [Left(x,y,v) \rightarrow \neg P(image(x,v),image(y,v))]$$

i.e. in the first case (A34) if from v , x is left of y , some part of x projects to the left of y , while no part of x projects to the right of y ; while in the second case (A35), from v , if x is left of y then x is not subtended by y .

It is now straightforward to see how *ROC-20* can be further developed. For example, where one object lies to the left of another and is disjoint, to the left and in boundary contact, and so on. All the distinct states depicted in figure 1 can then be modelled.

6 Relative Viewpoints

A change of viewpoint always carries the possibility of a change in the apparent spatial relationships holding between bodies in the domain (figure 1). If, for example, two bodies are physically separated, and an agent is allowed to freely move around, several apparent spatial relationships may be seen to apply. However, for two bodies forming a part-whole relation, no change in the viewpoint will coincide with both bodies separating. These and other configuration possibilities form the basis of the set of global *spatial constraints* introduced in section 2.2. There still remains the question of singling out additional *dynamic* spatial constraints, this time arising from instantaneous transitions between temporally ordered sequences of occlusion events.

As with many discrete based QSR theories, the set of *JEPD^{ROC-20}* relations can be worked into an *envisionment*, where a set of axioms lay out the dynamic possibilities and constraints of spatial relationships deemed to hold between bodies over consecutive moments in time [Cohn, 1997]. For *ROC-20* this is represented as a table (table 2) where legal/illegal (instantaneous) transitions between spatial relationships are respectively denoted by “y” (yes) or “n” (no) entries mapping to pairs of named occlusion relations. A path formed by linking together pairs of nodes denotes a possible projected sequence of states from an initial state (at time t) via successor states (at times $t+1 \dots t+n$).

The symmetry about the highlighted diagonal indicates the symmetrical relationship between each pair of named nodes. For example, the relation *NonOccludesEC/3* has five such legal transitions, as read across the named row or down the named column. This means the relation *NonOccludesEC/3* from time t to the next instant $t+1$, now re-worked as an envisionment axiom (assuming a fixed viewpoint and the continued existence of the bodies from time t to $t+1$), has the following form:

$$\forall x \forall y \forall v \forall t [HoldsAt(NonOccludesEC(x,y,v),t) \rightarrow [HoldsAt(NonOccludesEC(x,y,v),t+1) \vee HoldsAt(NonOccludesDC(x,y,v),t+1) \vee HoldsAt(PartiallyOccludesPO(x,y,v),t+1) \vee HoldsAt(PartiallyOccludesPO^{-1}(x,y,v),t+1) \vee HoldsAt(MutuallyOccludesPO(x,y,v),t+1)]]]$$

	NonOccludesDC	NonOccludesEC	NonOccludesPO	PartiallyOccludesTPP	PartiallyOccludesNTTPP	TotallyOccludesEQ	TotallyOccludesTPPI	TotallyOccludesNTPPI	MutuallyOccludesPO	MutuallyOccludesTPP	MutuallyOccludesNTTPP	PartiallyOccludesPO ⁻¹	PartiallyOccludesTPP ⁻¹	PartiallyOccludesNTTPP ⁻¹	TotallyOccludesEQ ⁻¹	TotallyOccludesTPPI ⁻¹	TotallyOccludesNTPPI ⁻¹	MutuallyOccludesTPP ⁻¹	MutuallyOccludesNTTPP ⁻¹	MutuallyOccludesEQ	
NonOccludesDC	y	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
NonOccludesEC	y	y	y	n	n	n	n	n	y	n	n	y	n	n	n	n	n	n	n	n	n
PartiallyOccludesPO	n	y	y	y	n	y	y	n	y	n	n	n	n	n	n	n	n	n	n	y	n
PartiallyOccludesTPP	n	n	y	y	y	y	n	n	y	y	y	n	n	n	n	n	n	n	n	n	y
PartiallyOccludesNTTPP	n	n	n	y	y	y	n	n	n	y	y	n	n	n	n	n	n	n	n	n	y
TotallyOccludesEQ	n	n	y	y	y	y	y	y	y	y	n	n	n	n	n	n	n	n	n	y	y
TotallyOccludesTPPI	n	n	y	n	n	y	y	y	n	n	n	n	n	n	n	n	n	n	n	y	y
TotallyOccludesNTPPI	n	n	n	n	n	y	y	y	n	n	n	n	n	n	n	n	n	n	n	y	y
MutuallyOccludesPO	n	y	y	y	n	y	y	n	y	y	n	y	n	y	n	y	y	n	y	n	y
MutuallyOccludesTPP	n	n	y	y	y	n	n	y	y	y	y	n	n	y	y	n	n	y	n	n	y
MutuallyOccludesNTTPP	n	n	n	y	y	n	n	n	y	y	n	n	n	y	y	n	n	y	n	n	y
PartiallyOccludesPO ⁻¹	n	y	n	n	n	n	n	n	y	n	n	y	n	y	n	y	n	n	n	n	y
PartiallyOccludesTPP ⁻¹	n	n	n	n	n	n	n	n	y	n	n	y	y	y	n	n	n	n	y	y	y
PartiallyOccludesNTTPP ⁻¹	n	n	n	n	n	n	n	n	n	n	n	n	y	y	n	n	n	n	y	y	y
TotallyOccludesEQ ⁻¹	n	n	n	n	n	n	n	n	y	y	y	y	y	y	y	y	y	y	y	y	y
TotallyOccludesTPPI ⁻¹	n	n	n	n	n	n	n	n	y	y	n	n	n	n	n	n	n	n	n	n	y
TotallyOccludesNTPPI ⁻¹	n	n	n	n	n	n	n	n	n	y	n	n	n	n	n	n	n	n	n	n	y
MutuallyOccludesTPP ⁻¹	n	n	y	n	n	y	y	y	n	n	n	y	y	y	n	n	n	n	n	y	y
MutuallyOccludesNTTPP ⁻¹	n	n	n	n	n	y	y	n	n	n	n	y	y	n	n	n	n	n	n	y	y
MutuallyOccludesEQ	n	n	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y

Table 2: The envisionment table for *ROC-20*

The envisionment table can be interpreted two ways: either in terms of the viewpoint changing, or where the positions of the bodies change. In the former case, an additional predicate is required: *ChangePos(v1,v2)*, (meaning viewpoint $v1$ changes to viewpoint $v2$), which relates $v1$ at time t in the antecedent of the envisionment axiom to $v2$ at time $t+1$ in the consequent. These sequences of occlusion events can then be viewed as building the *topology* of motion parallax into the model. Obviously, where orientation information is added the number of relations and nodes increase, as does the overall complexity of the new set of permissible transitions between specified named occlusion relations.

7 Discussion and Conclusions

ROC-20 presents an axiomatisation of spatial occlusion. It assumes the region based ontology of *RCC-8* [Randell, et al., 1992] and extends the work of Galton [1994] by allowing both convex and concave shaped bodies. It is this extension that describes occlusion events of mutually occluding bodies. The inclusion of van Benthem’s [1982] notion of comparative nearness facilitates reasoning about relative distance between occluding bodies. An envisionment table models sequences of occlusion events to enable reasoning about objects and the images that may be formed in a visual field.

Several directions for future work are indicated. The axiomatisation of the primitive relation: *TotallyOccludes/3*, currently rules out models where the occluding body has a part that wraps behind the occluded body. In the theory this is a case of *mutual occlusion*. However, we can see potential gains by re-working the current axiomatisation (and relaxing this restriction) so that *any* degree of enclosure of one body by another (from some assumed viewpoint) could be a case of total occlusion.

Additional work is required to generate the *composition table* [see Cohn, 1997] for *JEPD* subsets of the defined occlusion relations. Also of note is the question whether there are any decidable and tractable subsystems of *ROC-20*, as has already been shown for *RCC-8* [Bennett, 1994; Renz and Nebel, 1998]. Further computational gains may be made by adding information about the relative size of bodies or regions acting as additional constraints when checking for consistency of sets of these relations [c.f. Gerevini and Renz, 1998].

ROC-20 lays the theoretical foundations for further work in Cognitive Robotics, in which the images of objects are used to infer the spatial arrangement of objects in a robot's world - ultimately with map building and route planning in mind. We argue that the modelling of occlusion and motion parallax within a traditional QSR approach offers a uniform framework to achieve this. Galton [1994] has already shown these lines of sight relations can partition an idealized plan view of the embedding space into a set of polygonal regions. For each (view) point in that space exactly one of the *JEPD* line of sight relations holds. Where objects of varying shapes and sizes exist, many named sight lines that form tangents to objects naturally intersect at points. These correspond in this theory to a conjunction of atomic formulae drawn from the set of *JEPD* relations used. This gives rise to a set of extrinsic *reference points* determined completely by the objective spatial arrangement of the objects in the robot's world. With these points, *localization* becomes possible, while enabling qualitative and metric quantitative information to be combined. Spatial constraints and envisionment axioms now lead into map building and route planning. The robot then acquires the means to plan and execute moves [Levitt and Lawton, 1990; Schlieder, 1993] while constantly monitoring and relating its own direction of movement to the observed change and sequence of occlusion events in its visual field.

Acknowledgements

Work described here has been supported by EPSRC project GR/N13104, "Cognitive Robotics II". We wish to thank Paulo Santos, Brandon Bennett and Antony Galton for fruitful discussions during the development of ideas presented in this paper; and the comments given by the anonymous referees.

References

[Braddick and Atkinson, 1982] Braddick O. J. and Atkinson J., Higher Functions in Vision, in Barlow, H. B. and Mollon, J. D. (eds.) *The Senses*. Cambridge, Cambridge University Press, page 224

- [Bennett, 1994] Bennett, B., Spatial Reasoning with Propositional Logic, *Proc. 4th Int. Conf. on Knowledge Representation and Reasoning (KR-94)*, pages 51-62
- [Cohn, 1997] Cohn, A.G., Qualitative spatial representation and reasoning techniques, *Proc. KI-97, Advances in Artificial Intelligence*, LNAI 1303, Springer, pages 1-30
- [Cui, et al., 1992] Cui, Z, Cohn, A.G and Randell D.A., Qualitative simulation based on a logical formalism of space and time, *Proc. AAAI-92*, pages 679-684
- [Galton, 1994] Galton, A.P., Lines of Sight, *AISB Workshop on Spatial and Spatio-Temporal Reasoning*.
- [Geiger, et al., 1995] Geiger, D., Ladendorf, B. and Yuille, A., Occlusions and Binocular Stereo, *Int. J. of Computer Vision*, Vol. 14, pages 211-226
- [Gerevini and Renz, 1998] Gerevini, A and Renz, J., Combining Topological and Qualitative Size Constraints for Spatial Reasoning, *Proc. 4th Int. Conf. on Principles and Practice of Constraint Programming (CP-98)*, pages 220-234
- [Levitt and Lawton, 1990] Levitt, T.S. and Lawton, D.T., Qualitative Navigation for Mobile Robots, *Artificial Intelligence*, Vol. 44, pages 305-360
- [Petrov and Kuzmin, 1996] Petrov, A.P. and Kuzmin, L.V., Visual Space Geometry Derived from Occlusion Axioms, *J. of Mathematical Imaging and Vision*, Vol. 6, pages 291-308
- [Plantinga and Dyer, 1990] Plantinga, H. and Dyer, C.R., Visibility, Occlusion, and the Aspect Graph, *Int. J. of Computer Vision*, Vol. 5, pages 137-160
- [Randell, et al., 1992] Randell, D. A.; Cui, Z and Cohn A. G., A Spatial Logic Based on Regions and Connection, *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning (KR-92)*, Morgan Kaufmann, San Mateo, pages 165-176
- [Renz and Nebel, 1998] Renz, J. and Nebel, B., Efficient Methods for Qualitative Spatial Reasoning, *Proc. 13th Euro. Conf. on Artificial Intelligence (ECAI-98)*, John Wiley & Sons, pages 562-566
- [Schlieder, 1993] Schlieder, C., Representing Visible Locations for Qualitative Navigation, in Piera Carrate, N. and Singh, M.G. (eds.) *Qualitative Reasoning and Decision Technologies*, Barcelona: CIMNE, pages 523-532
- [Shanahan, 1996] Shanahan, M.P., Robotics and the Common Sense Informatic Situation, *Proc. 12th Euro. Conf. on Artificial Intelligence (ECAI-96)*, pages 684-688
- [van Benthem, 1982] van Benthem, J., *The Logic of Time*, Synthese Library Vol. 56, Reidel, London, Appendix A.

KNOWLEDGE REPRESENTATION AND REASONING

QUALITATIVE REASONING
FOR BIOLOGICAL SYSTEMS

Qualitative Simulation of Genetic Regulatory Networks: Method and Application

Hidde de Jong,¹ Michel Page,^{1,2} Céline Hernandez,¹ and Johannes Geiselmann³

¹INRIA Rhône-Alpes, 655 avenue de l'Europe, Montbonnot, 38334 Saint Ismier CEDEX, France

²ESA, Université Pierre Mendès France, Grenoble ³PEGM, Université Joseph Fourier, Grenoble, France

{Hidde.de-Jong, Céline.Hernandez, Michel.Page}@inrialpes.fr, Johannes.Geiselmann@ujf-grenoble.fr

Abstract

Computer modeling and simulation are indispensable for understanding the functioning of an organism on a molecular level. We present an implemented method for the qualitative simulation of large and complex genetic regulatory networks. The method allows a broad range of regulatory interactions between genes to be represented and has been applied to the analysis of a real network of biological interest, the network controlling the initiation of sporulation in the bacterium *B. subtilis*.

1 Introduction

It is now commonly accepted in biology that most interesting properties of an organism emerge from the interactions among its genes, proteins, metabolites, and other molecules. This implies that, in order to understand the functioning of an organism, the networks of interactions involved in gene regulation, metabolism, signal transduction, and other cellular and intercellular processes need to be elucidated.

A *genetic regulatory network* consists of a set of genes and their mutual regulatory interactions. The interactions arise from the fact that genes code for proteins that may control the expression of other genes, for instance by activating or inhibiting DNA transcription [Lewin, 1999]. The study of genetic regulatory networks has received a major impetus from the recent development of experimental techniques permitting the spatiotemporal expression levels of genes to be rapidly measured in a massively parallel way [Brown and Botstein, 1999]. However, in addition to experimental tools, computer tools for the modeling and simulation of gene regulation processes will be indispensable. As most genetic regulatory systems of interest involve many genes connected through interlocking positive and negative feedback loops, an intuitive understanding of their dynamics is hard to obtain.

Currently, only a few regulatory networks are well-understood on the molecular level, and quantitative information about the interactions is seldom available. This has stimulated an interest in modeling and simulation techniques developed within qualitative reasoning (QR) [Heidtke and Schulze-Kremer, 1998; Trelease *et al.*, 1999]. A major problem with these approaches, based on well-known methods like QSIM [Kuipers, 1994] and QPT [Forbus, 1984], is their

lack of upscalability. Following approaches in mathematical biology, de Jong and Page [2000] have proposed a qualitative simulation method capable of handling large and complex networks.

The aim of this paper is to generalize the latter method and to demonstrate its applicability to real networks of biological interest. The generalization of the method allows a broader range of regulatory interactions between genes to be expressed. This enables more complex systems to be analyzed, such as the network of interactions controlling the initiation of sporulation in the bacterium *Bacillus subtilis*. We have simulated the sporulation network using a model constructed from published reports of experiments. The simulations reveal that an additional interaction, proposed in the literature before but not yet experimentally identified, may be involved.

In the next section, we will discuss the class of equations being used to model genetic regulatory networks. The third section describes the qualitative simulation algorithm, focusing on the representation of the qualitative state of a regulatory system and the determination of state transitions by the simulation algorithm. The subsequent sections present the results of the analysis of the sporulation network as well as a discussion of the method in the context of related work.

2 Modeling genetic regulatory networks

2.1 Approximations of regulatory interactions

In order to model a genetic regulatory network, we first have to describe the regulatory interactions in an empirically valid and mathematically rigorous way. Consider a DNA-binding protein encoded by gene j , activating the expression of a target gene i . The rate of transcription of i as a function of the concentration x_j of the regulatory protein follows a sigmoid curve [Yagil and Yagil, 1971]. Below a threshold concentration $\theta_j > 0$ the gene is hardly expressed at all, whereas above this threshold its expression rapidly saturates.

Sigmoid curves are also found in the case of more complex regulatory mechanisms. Consider the proteins J and K that form a dimer repressing the transcription of gene i (Fig. 2(b)). Analysis of a kinetic model of this regulatory mechanism reveals that the rate of expression of i depends in a sigmoidal fashion on the total concentrations x_j and x_k of J and K, respectively. That is, both J and K need to be available above

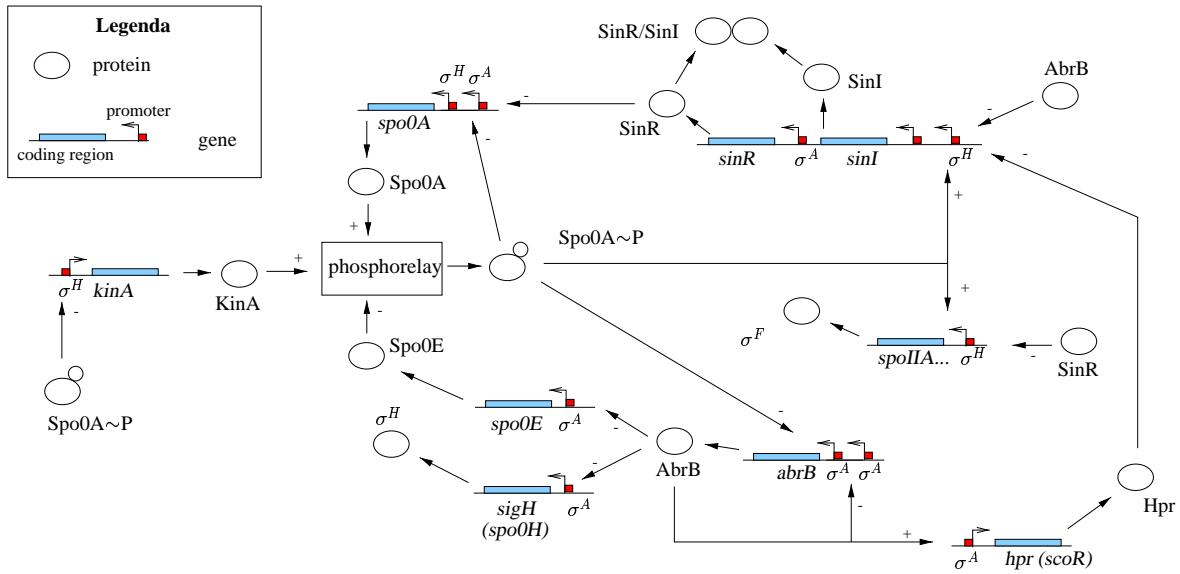


Figure 1: Genetic regulatory network underlying the initiation of sporulation in *B. subtilis*. For every gene, the coding region and the promoters are shown. Promoters are distinguished by the specific σ factor directing DNA transcription. The regulatory action of a protein tending to activate (inhibit) expression is indicated by a '+' ('-'). As a notational convention, names of genes are printed in *italics* and names of proteins start with a capital.

their threshold concentrations for i to be repressed.

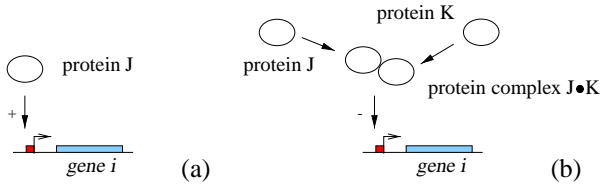


Figure 2: (a) Activation of a target gene i by a regulatory protein J . (b) Inhibition of i by a protein complex $J \bullet K$.

In the case of steep sigmoids, the combined effect of regulatory proteins on gene expression can be approximated by means of Boolean functions [Kauffman, 1993; Thomas and d'Ari, 1990]. Here we will rewrite a Boolean function in terms of step functions and arithmetic sums and multiplications, following the procedure of Plahte *et al.* [1998]. For the activator protein in Fig. 2(a), we thus obtain a *regulation function* $r(x_j) = \kappa s^+(x_j, \theta_j)$, where $s^+(\cdot)$ is a step function (Fig. 5) and $\kappa > 0$ a rate parameter. Similarly, the effect of a regulatory protein repressing gene i can be described by $r(x_j) = \kappa s^-(x_j, \theta_j)$, with $s^-(x_j, \theta_j) = 1 - s^+(x_j, \theta_j)$. The dimer repressor example in Fig. 2(b) leads to the function $r_i(x_j, x_k) = \kappa_i (1 - s^+(x_j, \theta_j) s^+(x_k, \theta_k))$.

Although the above discussion has focused on the representation of interactions regulating the synthesis of proteins, it also applies to the degradation of proteins. Sigmoid relations are observed in the latter case as well, so that the logical approximations are valid. In order to formally distinguish protein degradation rates from protein synthesis rates, we will denote the former by γ_i instead of κ_i .

2.2 State equations

The dynamics of genetic regulatory networks can be modeled by a system of differential equations suggested by Mestl *et al.* [1995], extending earlier proposals by Glass and Kauffman [1973] and Thomas *et al.* [1990].

$$\dot{x}_i = f_i(\mathbf{x}) - g_i(\mathbf{x}) x_i, \quad x_i \geq 0, \quad 1 \leq i \leq n, \quad (1)$$

where \mathbf{x} is a vector of cellular protein concentrations. The *state equations* (1) define the rate of change of the concentration x_i as the difference of the rate of synthesis $f_i(\mathbf{x})$ and the rate of degradation $-g_i(\mathbf{x})x_i$ of the protein. Exogenous variables can be defined by setting $\dot{x}_i = 0$.

The function $f_i : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$ is defined as

$$f_i(\mathbf{x}) = \sum_{l \in L} r_{il}(\mathbf{x}) > 0, \quad (2)$$

where $r_{il}(\cdot)$ is a regulation function and L a possibly empty set of indices of regulation functions. The function $g_i(\cdot)$ is defined analogously to (2), except that for reasons that will become clear below, we demand that $g_i(\mathbf{x})$ is strictly positive. Notice that for the above definitions of $f_i(\cdot)$ and $g_i(\cdot)$, the state equations (1) are *piecewise-linear*. Equations of this form, and their logical abstractions, have been well-studied in mathematical biology (e.g., [Lewis and Glass, 1991; Mestl *et al.*, 1995; Thomas and d'Ari, 1990]).

Eqs. (1) and (2) generalize upon the formalism employed in [de Jong and Page, 2000] in two respects. First, the regulation functions $r_{il}(\cdot)$ may be the mathematical equivalent of any Boolean function, whereas in the earlier paper it was restricted to logical functions composed of Boolean products. Second, the regulation of protein degradation can now

State equation for SpoOE:

$$\dot{x}_{se} = \kappa_{se} s^-(x_{ab}, \theta_{ab1}) s^+(x_a, \theta_{a1}) - \gamma_{se} x_{se},$$

Threshold inequalities:

$$0 < \theta_{se1} < \theta_{se2} < \theta_{se3} < max_{x_{se}}$$

Equilibrium inequalities: $0 < \kappa_{se}/\gamma_{se} < \theta_{se1}$

State equation for AbrB:

$$\dot{x}_{ab} = \kappa_{ab} s^-(x_{ab}, \theta_{ab2}) s^+(x_a, \theta_{a1}) \cdot (1 - s^+(x_{sa}, \theta_{sa1}) s^+(x_{ka}, \theta_{ka1}) s^-(x_{se}, \theta_{se3})) - \gamma_{ab} x_{ab}$$

Threshold inequalities:

$$0 < \theta_{ab1} < \theta_{ab2} < max_{x_{ab}}$$

Equilibrium inequalities: $\theta_{ab2} < \kappa_{ab}/\gamma_{ab} < max_{x_{ab}}$

Figure 3: State equations, threshold inequalities, and equilibrium inequalities for the genes (a) *spoOE* and (b) *abrB* in Fig. 1. The subscripts in the equations refer to the sporulation genes *kinA* (*ka*), *spoOE* (*se*), *spoOA* (*sa*), *sigA* (*a*), *sigH* (*h*), and *abrB* (*ab*).

be modeled, whereas before the degradation rate was set to $g_i(x) = \gamma_i$. These extensions allow the structure of complex regulatory networks to be formalized in a convenient way.

In Fig. 3 the state equations corresponding to two of the genes in the sporulation network of Fig. 1 are shown. The differential equation in (a) states that *spoOE* is transcribed at a rate κ_{se} from a σ^A -promoter when its repressor AbrB is below its threshold concentration θ_{ab1} (i.e., $s^-(x_{ab}, \theta_{ab1}) = 1$). In addition, for transcription to commence, the sigma factor σ_A encoded by *sigA* needs to be available at a concentration above the threshold θ_{sa1} (i.e., $s^+(x_{sa}, \theta_{sa1}) = 1$). SpoOE degrades at a rate proportional to its own concentration.

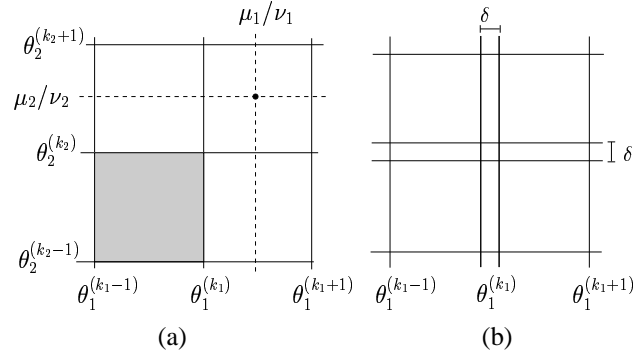


Figure 4: (a) Two-dimensional phase space divided into regulatory domains by the threshold planes. The shaded regulatory domain defined by $\theta_1^{(k1-1)} < x_1 < \theta_1^{(k1)}$ and $\theta_2^{(k2-1)} < x_2 < \theta_2^{(k2)}$ has a target equilibrium in an adjacent regulatory domain. (b) The same phase space with switching zones around the threshold planes.

2.3 Threshold and equilibrium inequalities

In general, a protein encoded by a gene will be involved in different interactions at different threshold concentrations. Although exact numerical values will not usually be available, we can order the p_i threshold concentrations of gene i , which gives the *threshold inequalities*

$$0 < \theta_i^{(1)} < \dots < \theta_i^{(p_i)} < max_{x_i}. \quad (3)$$

The parameter max_{x_i} denotes a maximum concentration for the protein denoted by i .

For the sporulation gene *abrB* two thresholds are defined, θ_{ab1} and θ_{ab2} . AbrB has two threshold concentrations: θ_{ab1} and θ_{ab2} . The first threshold corresponds to the repression of *spoOE*, *sigH*, and other early sporulation genes by AbrB. The second threshold corresponds to the autoregulation of *abrB* during vegetative growth, when AbrB levels are at their highest. This motivates the ordering of the AbrB thresholds in Fig. 3(b): $\theta_{ab1} < \theta_{ab2}$.

The $n - 1$ -dimensional threshold hyperplanes $x_i = \theta_i^{(k_i)}$, $1 \leq k_i \leq p_i$ divide the phase space box into regions, called *regulatory domains* (Fig. 4). Within each regulatory domain, the step function expressions in (2) can be evaluated, which reduces $f_i(\cdot)$ and $g_i(\cdot)$ to sums of rate constants. That is, $f_i(\cdot)$ simplifies to some $\mu_i \in M_i \equiv \{f_i(x) \mid \mathbf{0} \leq x \leq max_{x_i}\}$, and $g_i(\cdot)$ to some $\nu_i \in N_i \equiv \{g_i(x) \mid \mathbf{0} \leq x \leq max_{x_i}\}$. The sets M_i and N_i collect the different synthesis and degradation rates of the protein in different domains of the phase space.

It can be easily shown that all trajectories in a regulatory domain tend towards a single, stable steady state $x = \mu/\nu$, the *target equilibrium*, lying at the intersection of the hyperplanes $x_i = \mu_i/\nu_i$ [Glass and Kauffman, 1973; Mestl *et al.*, 1995; Thomas and d'Ari, 1990]. The target equilibrium level μ_i/ν_i of the protein concentration x_i gives an indication of the strength of gene expression in the regulatory domain.

As in the case of threshold parameters, exact numerical values for the rate constants will not usually be available. However, it is possible to order the possible target equilibrium levels of x_i in different regions of the phase space with respect to the threshold concentrations.¹ The resulting *equilibrium inequalities* define the strength of gene expression in a regulatory domain in a qualitative way, on the scale of ordered threshold concentrations. More precisely, for every $\mu_i \in M_i$, $\nu_i \in N_i$, we specify some l_i , $1 \leq l_i < p_i$, such that

$$\theta_i^{(l_i)} < \mu_i/\nu_i < \theta_i^{(l_i+1)}, \quad (4)$$

with special cases $0 < \mu_i/\nu_i < \theta_i^{(1)}$ and $\theta_i^{(p_i)} < \mu_i/\nu_i < max_{x_i}$.

Inspection of the state equations of AbrB shows that $M_{ab} = \{0, \kappa_{ab}\}$ and $N_{ab} = \{\gamma_{ab}\}$. The equilibrium level

¹The equilibrium inequalities have also been called *nullcline inequalities* [de Jong and Page, 2000], because the equilibrium levels correspond to nullcline hyperplanes in the phase space.

κ_{ab}/γ_{ab} is placed above the highest AbrB threshold, since otherwise the concentration of AbrB would never be able to reach or maintain a level at which negative autoregulation takes place. This leads to the equilibrium inequalities $\theta_{ab_2} < \kappa_{ab}/\gamma_{ab} < \max_{ab}$.

The threshold and equilibrium inequalities determine the sign of \dot{x}_i in a regulatory domain, and hence the local dynamics of the system. In fact, given a regulatory domain defined in dimension i by $\theta_i^{(k_i)} < x_i < \theta_i^{(k_i+1)}$, it can be shown that, if $\mu_i/\nu_i < \theta_i^{(k_i)}$, then $\dot{x}_i < 0$ everywhere inside the regulatory domain. Similarly, if $\mu_i/\nu_i > \theta_i^{(k_i+1)}$, then $\dot{x}_i > 0$. On the other hand, if $\theta_i^{(k_i)} < \mu_i/\nu_i < \theta_i^{(k_i+1)}$, then the sign of \dot{x}_i in the regulatory domain is not unique, written as $\dot{x}_i \lesseqgtr 0$. In particular, $\dot{x}_i < 0$ on one side of the hyperplane $x_i = \mu_i/\nu_i$, $\dot{x}_i > 0$ on the other side of the plane, and $\dot{x}_i = 0$ inside the plane.

2.4 Discontinuities in state equations

The question must be raised what happens on the threshold hyperplanes $x_i = \theta_i^{(k_i)}$ separating the regulatory domains, where the step functions, and hence the state equations, are not defined. Several solutions are possible for this non-trivial problem. In this paper we follow the approach of Plahte *et al.* [1994] and replace the discontinuous step functions by continuous ramp functions (Fig. 5). The solution of the PLDEs with step functions is then defined to be the solution of the DEs with ramp functions considered in the limit $\delta \rightarrow 0$.

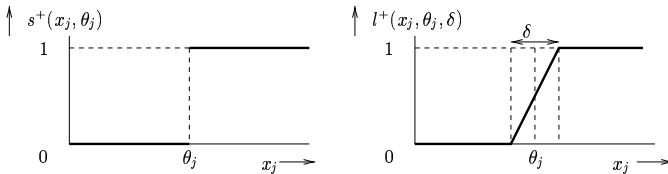


Figure 5: Step function $s^+(x_j, \theta_j)$ and ramp function $l^+(x_j, \theta_j, \delta)$. l^+ approaches s^+ as $\delta \rightarrow 0$.

The use of ramp functions divides the phase space into regulatory domains separated by *switching zones*, regions in which one or more x_i have a value in the δ -interval $[\theta_i^{(k_i)} - \delta/2, \theta_i^{(k_i)} + \delta/2]$ around a threshold $\theta_i^{(k_i)}$. An example phase space with switching zones is shown in Fig. 4. Inside the regulatory domains, the DEs with ramp functions are equivalent to the PLDEs with step functions. Outside the regulatory domains, in the switching zones, the DEs with ramp functions may be nonlinear functions of the concentration variables. The switching zones separating the regulatory domains vanish and approach (intersections of) the threshold hyperplanes as $\delta \rightarrow 0$.

3 Qualitative simulation of genetic regulatory networks

The goal of *qualitative simulation* is to exploit the qualitative constraints on parameter values in order to predict the qualitative dynamics of a regulatory network. More specifically, we would like to know which regulatory domains can

be reached by some solution trajectory starting in the initial regulatory domain, for parameter values consistent with the specified threshold and equilibrium inequalities. A sequence of regulatory domains thus generated gives an indication of the evolution of the functional state of the system, as transitions between regulatory domains reflect changes in the synthesis and degradation rates of proteins.

3.1 Qualitative values, states, and behaviors

The analysis of PLDEs of the form (1) motivates the introduction of the *qualitative value* of a state variable and its derivative, as well as definitions of the *qualitative state* and *qualitative behavior* of the system. Let $\xi(t) = \xi(x, x_0, \theta, \kappa, \gamma, t)$ be the solution of PLDEs with step functions describing a regulatory network on the time-interval $[t_0, t_\infty[$ for given parameter values and initial conditions $x(t_0) = x_0$.

Def. 1 (Qualitative value) Suppose that at some $t > t_0$ it holds that $\xi(t)$ lies in a regulatory domain, such that $\theta_i^{(k_i)} < \xi_i(t) < \theta_i^{(k_i+1)}$ for every i ($1 \leq i \leq n$ and $1 \leq k_i < p_i$). The qualitative value $qv(\xi_i, t)$ of $\xi_i(t)$ is given by the inequalities $\theta_i^{(k_i)} < x_i < \theta_i^{(k_i+1)}$, while the qualitative value $q\dot{v}(\xi_i, t)$ of $\dot{\xi}_i(t)$ is given by one of the inequalities $\dot{x}_i < 0$, $\dot{x}_i > 0$, $\dot{x}_i \lesseqgtr 0$, depending on the sign of $\dot{\xi}_i(t)$ in the regulatory domain.

The definition can be straightforwardly generalized to the case of regulatory domains bounded by $x_i = 0$ or $x_i = \max_{ab}$. Notice that the qualitative value is not defined in the switching zones around the threshold hyperplanes.

Def. 2 (Qualitative state) The qualitative state $qs(\xi, t)$ for ξ at t is given by the vectors $qv(\xi, t)$ and $q\dot{v}(\xi, t)$ of qualitative values.

A qualitative state associated with a regulatory domain can be interpreted as representing a functional state of the regulatory system. Each protein concentration has a value lying between two consecutive thresholds, and is either tending towards one of the threshold values, or evolving towards a value between the thresholds.

Def. 3 (Qualitative behavior) The qualitative behavior $qb(\xi)$ of ξ is given by the sequence of qualitative states $qs(\xi, t)$ on $[t_0, t_\infty[$.

A qualitative behavior defines a succession of qualitative states of the regulatory system. It is not difficult to show that every solution ξ of (1) can be abstracted into a unique qualitative behavior.

3.2 Qualitative simulation algorithm

In terms of the above definitions, the qualitative simulation procedure can be formulated as follows [Kuipers, 1994]. Given initial qualitative values qv_0 , describing the initial protein concentrations x , the simulation algorithm computes the initial qualitative state qs_0 , and then determines all possible *transitions* from qs_0 to successor qualitative states. The generation of successor states is repeated in a recursive manner until all qualitative states reachable from the initial qualitative state have been found.

The possible transitions from a qualitative state are determined by the rule below.

qv_i and $\dot{q}v_i$	qv'_i and $\dot{q}v'_i$
$\theta_i^{(k_i)} < x_i < \theta_i^{(k_i+1)}, \dot{x}_i > 0$	$\theta_i^{(k_i+1)} < x_i < \theta_i^{(k_i+2)}, \dot{x}_i > 0$
$\theta_i^{(k_i)} < x_i < \theta_i^{(k_i+1)}, \dot{x}_i > 0$	$\theta_i^{(k_i+1)} < x_i < \theta_i^{(k_i+2)}, \dot{x}_i \leq 0$
$\theta_i^{(k_i)} < x_i < \theta_i^{(k_i+1)}, \dot{x}_i < 0$	$\theta_i^{(k_i-1)} < x_i < \theta_i^{(k_i)}, \dot{x}_i < 0$
$\theta_i^{(k_i)} < x_i < \theta_i^{(k_i+1)}, \dot{x}_i < 0$	$\theta_i^{(k_i-1)} < x_i < \theta_i^{(k_i)}, \dot{x}_i \leq 0$

Figure 6: Continuity constraints for the qualitative values $qv_i, \dot{q}v_i$ and $qv'_i, \dot{q}v'_i$ of two qualitative states qs and qs' defined on adjacent regulatory domains. Valid for $1 < k_i < p_i - 1$, the constraints can be easily generalized to the case of qualitative values $0 \leq x_i < \theta_i^{(1)}$ and $\theta_i^{(p_i)} < x_i \leq \max_i$.

Def. 4 (State transition) Let qs and qs' be two qualitative states associated with adjacent regulatory domains. A transition from qs to qs' is possible, if for every i , $1 \leq i \leq n$, such that $qv_i \neq qv'_i$, the qualitative values $qv_i, \dot{q}v_i$ and $qv'_i, \dot{q}v'_i$ satisfy the continuity constraints in Fig. 6.

Fig. 7(a) illustrates the application of the rule. Intuitively formulated, the rule says that a transition from one qualitative state to another is possible, if a trajectory may cross the switching zone separating the regulatory domains of the qualitative states.

A simulation algorithm based on Def. 4 is described in [de Jong and Page, 2000]. The qualitative states and transitions generated by the algorithm form a *state transition graph*. The graph may contain cycles and states without successors, which are together referred to as *attractors*. Since the number of possible qualitative states is finite, every path in the state transition graph will reach an attractor at some point. Each path running from the initial qualitative state qs_0 to an attractor forms a possible qualitative behavior of the regulatory system.

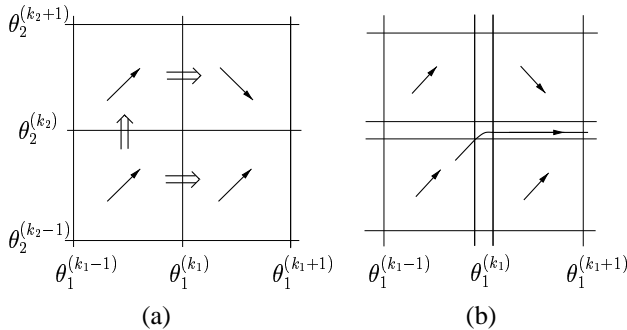


Figure 7: (a) Phase space with derivative vectors \dot{x} in the regulatory domains (\rightarrow) and the state transitions (\Rightarrow) permitted by Def. 4. (b) Solution trajectory escaping through switching zones.

3.3 Properties of simulation algorithm

Given a model and initial qualitative values qv_0 , what can be said about the correctness of the behaviors produced by qualitative simulation? We define a set Ξ of possible solutions of (1) on $[t_0, t_\infty[$, such that for every $\xi \in \Xi$ the numerical values of $\theta, \kappa,$ and γ satisfy (3)-(4), and $qv(\xi, t_0) = qv_0$.

The ultimate aim of qualitative simulation is to determine the set QB of qualitative behaviors, such that (1) for every $\xi \in \Xi$ there is a $b \in QB$, such that $qb(\xi) = b$ (soundness), and (2) for every $b \in QB$ there is a $\xi \in \Xi$, such that $qb(\xi) = b$ (completeness).

Unfortunately, the simulation algorithm based on Def. 4 is not sound. A trajectory may enter a switching zone from a regulatory domain, and then escape through other switching zones to enter another, possibly non-adjacent regulatory domain (Fig. 7(b)). As a consequence, the simulation algorithm may overlook qualitative state transitions. We found that the practical consequences of such omissions are limited, since qualitative states not directly reachable by a transition are often indirectly reachable by a sequence of transitions.

Completeness of the simulation algorithm has neither been proven nor disproven, but seems difficult to guarantee given the behavioral complexity that can be attained by models of the form (1) [Lewis and Glass, 1991].

3.4 Genetic Network Analyzer

The simulation method has been implemented in Java 1.2, in a program called *GNA (Genetic Network Analyzer)*.² The program reads and parses input files specifying the model of the system (state equations, threshold and equilibrium inequalities) and the initial state. From this information it produces a state transition graph. Extensions of the simulation algorithm allow all qualitative states and their transitions to be generated, as well as the completion and simulation of models with unspecified threshold and equilibrium inequalities.

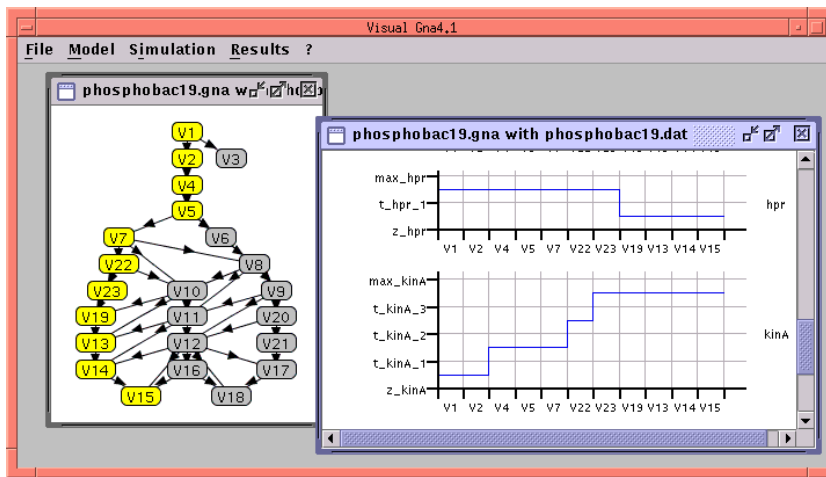
GNA is accessible through a graphical user-interface, which allows the network of interactions between genes to be displayed, as well as the state transition graph resulting from the simulation. In addition, the user can analyze the attractors with their basins of attraction, and focus on qualitative behaviors to study the temporal evolution of protein concentrations in more detail (Fig. 8).

4 Application: sporulation in *B. subtilis*

The method and its implementation have been used to study the regulatory network underlying the initiation of the *sporulation* process in the Gram-positive soil bacterium *Bacillus subtilis* [Grossman, 1995; Hoch, 1993]. Under conditions of nutrient deprivation, *B. subtilis* can decide not to divide and form a dormant, environmentally-resistant spore instead. The decision to either divide or sporulate is made by a complex regulatory network integrating various environmental, cell-cycle, and metabolic signals.

A schematic representation of the core of this network, displaying key genes and their regulatory interactions, is shown in Fig. 1. The central component of the network is a phosphorylation pathway, a *phosphorelay*, which transfers phosphates from the KinA kinase to the Spo0A regulator. Above a certain threshold, the phosphorylated form of Spo0A (Spo0A~P) activates various genes that commit the bacterium to sporulation. An example is the *spoIIA* operon, which encodes the transcription factor σ^F , essential for the development of the forespore. The flux of phosphate across the phosphorelay,

²GNA is available from the authors upon request.



(a)

$0 < \kappa_{se}/\gamma_{se} < \theta_{se1}$
23 states
attractors: sporulation and division
$\theta_{se1} < \kappa_{se}/\gamma_{se} < \theta_{se2}$
44 states
attractors: division
$\theta_{se2} < \kappa_{se}/\gamma_{se} < \theta_{se3}$
65 states
attractors: division
$\theta_{se3} < \kappa_{se}/\gamma_{se} < max_{se}$
1155 states
attractors: division

(b)

Figure 8: (a) GNA output for the equilibrium inequalities $0 < \kappa_{se}/\gamma_{se} < \theta_{se1}$. The left window shows the state transition graph with a qualitative behavior running from the initial qualitative state V1 to the attractor state V15. In the right window the temporal evolution of the qualitative value of Hpr and KinA for this behavior can be followed. (b) Summary of simulation results for an initial state expected to induce sporulation, while varying the equilibrium inequalities for Spo0E. The simulations take between 0.5 and 3 seconds to complete on a Sun Ultra 10 workstation.

and hence the concentration of Spo0A~P, is controlled by various external signals influencing the activity of pathway components. In addition, the flux of phosphate is regulated by Spo0A~P itself, through a number of direct and indirect feedback loops involving *abrB*, *sinI*, *sinR*, and other genes.

Using the extensive literature on *B. subtilis* sporulation, and the Subtilist database at the Institut Pasteur, we have formulated state equations and appropriate parameter inequalities for every gene in the network. In total, the mathematical model consists of 10 state equations, 10 threshold inequalities, and 30 equilibrium inequalities. In the rare cases that the literature did not unambiguously determine the parameter inequalities, we have systematically explored the alternatives and selected those that permit the observed behavior of the bacterium to be reproduced.

The behavior of *B. subtilis* has been simulated from a variety of initial states, reflecting different physiological conditions. For example, fig. 8(a) shows the simulation results for an initial state reflecting a perturbation of the vegetative growth conditions, when the protein kinase KinA autophosphorylates in response to an external signal indicating a state of nutritional deprivation. Under these conditions, a state transition graph with two attractors is produced, corresponding to states in which the bacterium continues to divide (V3) or initiates spore formation (V15 and V18). Both states may be reached, depending on the exact values of the parameters satisfying the threshold and equilibrium inequalities.

In order to obtain result consistent with experimental data, we found that the target equilibrium concentrations of Spo0E have to be placed below the lowest threshold concentrations ($0 < \kappa_{se}/\gamma_{se} < \theta_{se1}$). That is, we need to assume that *spo0E* expression levels are quite weak. When other equilibrium inequalities are chosen, the simulations predict that sporulation cannot be initiated under appropriate conditions, contrary to

what is observed (Fig. 8(b)). In fact, Spo0E mediates the negative autoregulation of Spo0A~P, and thus prevents a critical concentration of Spo0A~P to accumulate.

The above choice of parameter constraints is troublesome, because it implies that Spo0E cannot exert any influence on the decision to sporulate, since its concentration will not reach the threshold levels above which it can block the phosphate flux through the phosphorelay. The simulation results thus suggest that the network in Fig. 1, based on interactions reported in the literature, may be incomplete. As a remedy, we could postulate that an unknown signal decreases the activity of Spo0E at the onset of sporulation. Molecular studies of the interaction of Spo0E with components of the phosphorelay suggest the existence of such a cellular factor which remains as of yet unidentified [Ohlsen *et al.*, 1994].

5 Discussion

We have presented an implemented method for the qualitative simulation of genetic regulatory systems that can handle large and complex networks of genes and interactions. The method is a generalization of the method in [de Jong and Page, 2000], in that it allows a larger class of regulatory relationships between genes to be modeled. In the first place, there are no restrictions on the logical functions that can be represented by (2). This permits complex regulatory interactions to be included in the models, as illustrated by the state equation for *abrB* in Fig. 3(b). In the second place, the regulation of protein degradation can be taken into account. Although this feature of the method has not been used in the sporulation example, it turned out to be crucial in modeling the network controlling the induction of the lytic cycle following phage μ infection of *E. coli* (results not shown here).

The applicability of the method to actual regulatory networks has been demonstrated by an analysis of the large

and complex network underlying the initiation of sporulation in *B. subtilis*. The analysis has resulted in a suggestion to complete the model compiled from the sporulation literature, which shows the potential of the method as a tool to focus further experimentation. To our knowledge, qualitative simulation of genetic regulatory networks of the size and complexity considered in this paper has not been undertaken thus far.

Upscaling of the simulation method is achieved by modeling genetic regulatory systems by a class of piecewise-linear differential equations imposing strong constraints on the local dynamics of the system. Besides in molecular genetics, PLDEs of this form have been used in other biological domains, for instance in population biology. In order to effectively apply the constraints, the representation of the qualitative state of the system and the simulation algorithm are adapted to the mathematical structure of the equations.

Adaptation to a specific class of models is the principal respect in which the method presented in this paper differs from well-known QR methods like QPT and QSIM [Forbus, 1984; Kuipers, 1994]. A major difference with QSIM is that the qualitative state of a regulatory system is described on a higher level of abstraction. In particular, the behavior inside a regulatory domain is abstracted into a single qualitative state, making use of the fact that inside a regulatory domain either $\dot{x}_i < 0$, $\dot{x}_i > 0$, or $\dot{x}_i \leq 0$. In QSIM one would have to distinguish an exponentially growing number of qualitative states inside and on the boundary of a regulatory domain.

Approximating step functions by infinitely steep ramp functions allows a precise definition of the behavior of the system in the threshold planes, and hence of the possible successors of a qualitative state. The state transition rule in Def. 4 is simple and intuitively clear, but does not preserve soundness of the algorithm. Even though the practical limitations of this may be limited, the development of state transition rules guaranteeing soundness is an important topic for further research.

The *B. subtilis* example suggest an approach to validate hypothesized models of genetic regulatory networks. Given temporal gene expression patterns observed under certain physiological conditions in wild-type or mutant strains of the bacterium, one can develop algorithms to systematically search the space of freely adjustable parameter inequalities to find constraints for which the model is able to account for the observations. Extensions of this type would allow the simulation method presented in this paper to evolve into a more general modeling tool.

Acknowledgments The authors would like to thank I. Vatcheva and three anonymous referees for comments on a previous version of this paper.

References

- [Brown and Botstein, 1999] P.A. Brown and D. Botstein. Exploring the new world of the genome with DNA microarrays. *Nat. Genet.*, 21(suppl):33–37, 1999.
- [de Jong and Page, 2000] H. de Jong and M. Page. Qualitative simulation of large and complex genetic regulatory systems. In W. Horn, editor, *Proc. 14th Europ. Conf. Artif. Intell. (ECAI 2000)*, pages 141–145. IOS Press, 2000.
- [Forbus, 1984] K.D. Forbus. Qualitative process theory. *Artif. Intell.*, 24:85–168, 1984.
- [Glass and Kauffman, 1973] L. Glass and S.A. Kauffman. The logical analysis of continuous non-linear biochemical control networks. *J. Theor. Biol.*, 39:103–129, 1973.
- [Grossman, 1995] A.D. Grossman. Genetic networks controlling the initiation of sporulation and the development of genetic competence in *Bacillus subtilis*. *Annu. Rev. Genet.*, 29:477–508, 1995.
- [Heidtke and Schulze-Kremer, 1998] K.R. Heidtke and S. Schulze-Kremer. Design and implementation of a qualitative simulation model of λ phage infection. *Bioinformatics*, 14(1):81–91, 1998.
- [Hoch, 1993] J.A. Hoch. *spo0* genes, the phosphorelay, and the initiation of sporulation. In A.L. Sonenshein et al., editor, *Bacillus subtilis and other Gram-Positive Bacteria*, pages 747–756. AMS, 1993.
- [Kauffman, 1993] S.A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.
- [Kuipers, 1994] B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.
- [Lewin, 1999] B. Lewin. *Genes VII*. Oxford University Press, 1999.
- [Lewis and Glass, 1991] J.E. Lewis and L. Glass. Steady states, limit cycles, and chaos in models of complex biological networks. *Int. J. Bifurcation Chaos*, 1(2):477–483, 1991.
- [Mestl et al., 1995] T. Mestl, E. Plahte, and S.W. Omholt. A mathematical framework for describing and analysing gene regulatory networks. *J. Theor. Biol.*, 176:291–300, 1995.
- [Ohlsen et al., 1994] K.L. Ohlsen, J.K. Grimsley, and J.A. Hoch. Deactivation of the sporulation transcription factor Spo0A by the Spo0E protein phosphatase. *Proc. Natl Acad. Sci. USA*, 91:1756–1760, 1994.
- [Plahte et al., 1994] E. Plahte, T. Mestl, and S.W. Omholt. Global analysis of steady points for systems of differential equations with sigmoid interactions. *Dyn. Stab. Syst.*, 9(4):275–291, 1994.
- [Plahte et al., 1998] E. Plahte, T. Mestl, and S.W. Omholt. A methodological basis for description and analysis of systems with complex switch-like interactions. *J. Math. Biol.*, 36:321–348, 1998.
- [Thomas and d’Ari, 1990] R. Thomas and R. d’Ari. *Biological Feedback*. CRC Press, 1990.
- [Trelease et al., 1999] R.B. Trelease, R.A. Henderson, and J.B. Park. A qualitative process system for modeling NF- κ B and AP-1 gene regulation in immune cell biology research. *Artif. Intell. Med.*, 17:303–321, 1999.
- [Yagil and Yagil, 1971] G. Yagil and E. Yagil. On the relation between effector concentration and the rate of induced enzyme synthesis. *Biophys. J.*, 11:11–27, 1971.

Discrimination of Semi-Quantitative Models by Experiment Selection: Method and Application in Population Biology

Ivayla Vatcheva¹ Olivier Bernard² Hidde de Jong³ Jean-Luc Gouzé² Nicolaas J.I. Mars¹

¹Department of Computer Science
University of Twente
P.O. Box 217, 7500 AE Enschede
The Netherlands

²INRIA Sophia Antipolis
2004 Route des Lucioles, BP 93
06902 Sophia Antipolis
France

³INRIA Rhône-Alpes
655, avenue de l'Europe
38330 Montbonnot Saint Martin
France

Abstract

Modeling an experimental system often results in a number of alternative models that are justified equally well by the experimental data. In order to discriminate between these models, additional experiments are needed. We present a method for the discrimination of models in the form of semi-quantitative differential equations. The method is a generalization of previous work in model discrimination. It is based on an entropy criterion for the selection of the most informative experiment which can handle cases where the models predict multiple qualitative behaviors. The applicability of the method is demonstrated on a real-life example, the discrimination of a set of competing models of the growth of phytoplankton in a bioreactor.

1 Introduction

Obtaining an adequate model of an experimental system is a laborious and error-prone task. In many cases one arrives at a number of rival models that are justified equally well by the experimental data. In order to discriminate between these models, additional experiments are needed. Since in real-life applications the number of experiments to perform may be quite large, and the costs of each of them considerable, it is important that the experiments be selected carefully. In fact, experiments need to be chosen such that the set of possible models is maximally reduced at minimal costs.

For experimental systems described by differential equations, several approaches for *model discrimination* have been proposed in the literature (e.g. [Espie and Macchietto, 1989]). With few exceptions (e.g. [Struss, 1994; Vatcheva *et al.*, 2000]), these methods apply to completely specified quantitative models. That is, they cannot be used when precise numerical values for the parameters are not available and the precise form of functional relations is unknown.

This has motivated the development of a method for model discrimination that is able to handle incompletely specified models in the form of *semi-quantitative differential equations* (SQDEs). The method is based on an entropy criterion for the selection of the most informative discriminatory experiment. This experiment is determined from the behavioral predictions obtained from the competing models through simulation

under various experimental conditions.

In earlier work, we have developed a method for the discrimination of semi-quantitative models [Vatcheva *et al.*, 2000]. However, the previously proposed approach is restricted to the case that all models predict the same qualitative behavior, a situation rarely occurring in the case of more complex models. The method described in this paper is a generalization of the approach above in that it allows one to deal with situations in which multiple qualitative behaviors are predicted.

The applicability of the method is demonstrated on a real problem in population biology, the selection of experiments to discriminate between competing models of the growth of phytoplankton in a bioreactor. The choice of good discriminatory experiments is critical in this application, since the experiments may take several weeks to complete. Semi-quantitative models are appropriate, because the available data is incomplete and imprecise, as for most biological systems. We have compared the optimal experiment as determined by our method with an experiment that has been actually carried out. Furthermore, taking into account the results of the latter experiment, the best next experiment to perform has been suggested.

The paper is organized as follows. The next section deals with the basic concepts of semi-quantitative modeling and simulation. Sec. 3 gives an outline of the method for model discrimination, focusing on the criterion for selecting the most informative experiment. The results of the application of the method to the modeling of phytoplankton growth in a bioreactor are presented in Sec. 4. The final section discusses achievements and limitations of our work in the context of related work on model discrimination and gives some perspectives on further research.

2 Semi-quantitative modeling and simulation

Semi-quantitative differential equations (SQDEs) are abstractions of ordinary differential equations (ODEs) that allow incompletely or imprecisely specified dynamical systems to be modeled [Berleant and Kuipers, 1997]. In an SQDE, bounding envelopes are defined for unknown functions, as well as numerical intervals to bound the values of parameters and initial conditions.

Fig. 1 shows an example of a second-order SQDE describing the growth of the microalgae *Dunaliella tertiolecta* under

$$\begin{aligned}
& \text{Monod} \\
& \dot{x} = \mu(s)x - dx \\
& \dot{s} = d(s_{in} - s) - \rho(s)x \\
& \mu(s) = \mu_{max} \frac{s}{s + k_s} \\
& \rho(s) = \frac{1}{y} \mu(s)
\end{aligned}$$

$$\begin{aligned}
d & \in [0.095, 0.105], s_{in} \in [80, 120] \\
y & \in [0.15, 0.6], \mu_{max} \in [1.2, 1.6], k_s \in [0.01, 0.2]
\end{aligned}$$

Figure 1: An example of an SQDE describing the growth of phytoplankton in a bioreactor, the *Monod* model. The physical meaning of the variables and parameters is given in the caption of Fig. 3.

nutrient limitation in a bioreactor [Monod, 1942]. The state variables are the biomass x and the concentration of the limiting nutrient s . The intervals bounding the model parameters μ_{max} , k_s , and y have been estimated from preliminary experiments.

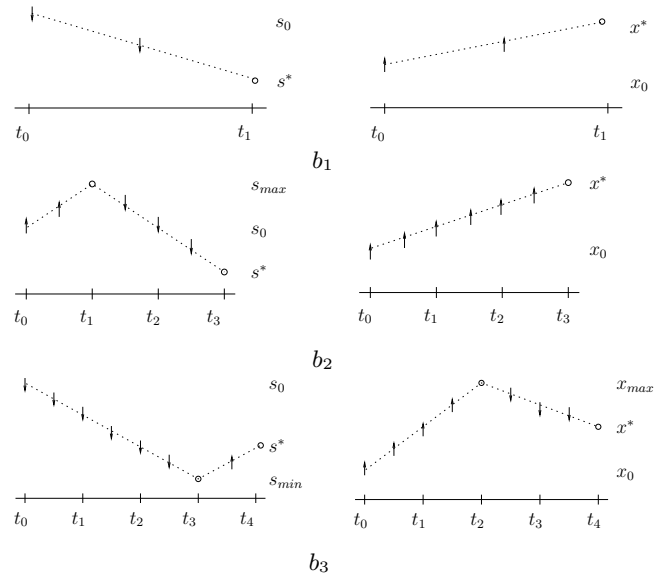
For the simulation of SQDEs we employ the techniques Q2+Q3 [Berleant and Kuipers, 1997], which refine the qualitative behavior tree produced by the QSIM algorithm [Kuipers, 1994]. The results of semi-quantitative simulation consist of one or more qualitative behaviors supplemented by ranges for the values of the variables at qualitatively significant time-points. The behaviors resulting from the simulation of the SQDE in Fig. 1 are shown in Fig. 2. In order to narrow down the interval predictions, we use the comparative analysis technique SQCA [Vatcheva and de Jong, 1999]. SQCA refines the simulation results by comparing a behavior predicted for one experiment with behaviors predicted for related experiments.

3 Method for model discrimination

The predictions obtained through semi-quantitative simulation can be exploited to maximally discriminate a set of competing models against minimal costs. The method for achieving this is based on a generalization of the entropy criterion for the most informative experiment developed in [Vatcheva *et al.*, 2000].

3.1 Model discrimination and behavior predictions

Consider a set M of competing models of an experimental system. Each $m_i \in M$ has a probability $p(m_i)$ to be the correct model of the system, and we assume that $\sum_{m_i \in M} p(m_i) = 1$ [Fedorov, 1972]. The *a priori* model probabilities are estimated from data obtained in preliminary experiments or assumed equal when no such data exist. Each time an experiment is executed, and new data becomes available, the model probabilities are being updated. If the data does not justify the predictions of some m_i , its *a posteriori* probability becomes zero.



	$s^* (\times 10^{-4})$	x^*	s_{max}
b_1	[6.31, 100]	[36, 72]	-
b_2	[14.9, 100]	[36, 72]	[0.005, 0.0335]
b_3	[6.31, 100]	[36, 59]	-
	$s_{min} (\times 10^{-5})$	x_{max}	
b_1	-	-	
b_2	-	-	
b_3	[5.85, 1000]	[36, 121]	

Figure 2: Behaviors resulting from the simulation of the SQDE in Fig. 1, for the initial conditions $x_0 \in [36, 39]$, $s_0 \in [0.005, 0.01]$. Behavior b_1 predicts that the system reaches its equilibrium (x^*, s^*) asymptotically. In b_2 , the nutrient concentration s reaches a maximum, before the system approaches its equilibrium. In b_3 , x reaches first a maximum, followed by a minimum of s and the equilibrium. The table summarizes the interval predictions for each of the three behaviors.

For the discrimination of the models in M , experiments from a set E of experiments need to be selected. The experiment that discriminates best between the models is estimated from the model predictions. For each experiment $e \in E$, the models in M are perturbed according to e , and then simulated to predict the behavior of the system under the experimental conditions. The prediction of m_i for some e is a set of behaviors B_i^e . The set of all qualitatively distinct behaviors resulting from the simulation of the models in M for e is denoted by B^e .

For discrimination purposes, only certain characteristics of the predicted behaviors $b \in B^e$ are taken into account. This gives rise to a set of *behavioral features* F_b for b . The set of behavioral feature consists of minima, maxima and equilibria of the system variables. The behavioral features defined for b_3 in Fig. 2, for instance, are the maximum of x and the minimum of s (x_{max} and s_{min}), and the steady state levels of these variables (s^* and x^*). Here we will assume that the value of a behavioral feature is an interval.

Intuitively, the experiment that can be expected to optimally discriminate between the models is the experiment for which the predicted values of the behavioral features overlap least. This intuition will be formalized below by defining the *most informative* experiment. A set of competing models can then be discriminated by repeatedly determining the most informative experiment, performing this experiment, and updating the model probabilities in the light of the outcomes.

3.2 Criterion for most informative experiment

A standard measure in information theory is the *information increment* of an experiment [Fedorov, 1972]. For every experiment $e \in E$ we define

$$\Delta H(b^e, \mathbf{Y}^e) = - \sum_{m_i \in M} p(m_i) \ln p(m_i) + \sum_{m_i \in M} p(m_i | b^e, \mathbf{Y}^e) \ln p(m_i | b^e, \mathbf{Y}^e), \quad (1)$$

where $p(m_i)$ and $p(m_i | b^e, \mathbf{Y}^e)$ are the *a priori* and *a posteriori* probabilities of m_i . b^e is the behavior of the system observed in e , and \mathbf{Y}^e is the vector of observations for the behavioral features F_{b^e} . The observations are assumed to be intervals $Y_j^e = [y_j^e - \varepsilon_j/2, y_j^e + \varepsilon_j/2]$, where y_j^e is the midpoint, and ε_j is the estimated size of the confidence interval for the j th behavioral feature. For clarity of presentation, we will assume for the moment, that each behavior b is characterized by a single feature.

ΔH reaches its maximum when all posterior probabilities but one are zero. That is, when the observations obtained in e confirm the predictions of a single model. On the other hand, a minimal value is attained, when all posterior probabilities are equal.

Since the *a posteriori* probabilities of the models depend on the outcome of the experiment which is not yet known, ΔH cannot be computed directly. Instead, we can compute its expected value, or the *expected information increment* of e :

$$\Delta J(e) = \sum_{b \in B^e} \int_{y \in D} \Delta H(b, Y) g^e(y, b) dy, \quad (2)$$

where B^e is the set of predicted behaviors, $Y = [y - \varepsilon/2, y + \varepsilon/2]$, D the domain of the behavioral feature, and $g^e(y, b)$ its probability distribution:

$$g^e(y, b) = \sum_{m_i \in M} p(m_i) p(b | m_i) g_i^e(y, b).$$

$p(b | m_i)$ is the probability of behavior b provided m_i is the correct model of the system, and $g_i^e(y, b)$ is the model-specific probability distribution of the behavioral feature, defined by

$$g_i^e(y, b) = \begin{cases} \frac{1}{\varepsilon} \frac{|Y \cap V_i^e|}{|V_i^e|}, & Y \cap V_i^e \neq \emptyset, \\ 0, & Y \cap V_i^e = \emptyset, \end{cases}$$

with V_i^e the interval prediction of m_i for the behavioral feature in experiment e , and $|\cdot|$ denoting interval length.

$g_i^e(y, b)$ expresses the probability that the value of the behavioral feature is Y , if m_i is the correct model of the system and b is the system behavior. If the interval Y does not overlap with the model prediction V_i^e , the probability of the feature having value Y is zero. Otherwise, the probability is weighted according to the size of the overlap between Y and V_i^e .

By substituting the expression for ΔH in (2) and using the Bayes' rule

$$p(m_i | b, Y) = \frac{p(m_i) p(b | m_i) g_i^e(y, b)}{\sum_{m_k \in M} p(m_k) p(b | m_k) g_k^e(y, b)},$$

we arrive at the following expression for the expected information increment of an experiment e :

$$\Delta J(e) = \sum_{m_i \in M} p(m_i) \sum_{b \in B^e} p(b | m_i) \int_{y \in D} g_i^e(y, b) \ln \frac{g_i^e(y, b) p(b | m_i)}{g^e(y, b)} dy. \quad (3)$$

The criterion ranks the experiments in E according to their expected informativeness. The optimal discriminatory experiment will be the *most informative experiment*, that is, the experiments for which $\Delta J(e)$ is maximal. Intuitively, experiments which give rise to predictions as different as possible will be favored. Fig. 4(a)-(b) shows the predictions of the four models given in Figs. 1 and 3 for two different experiments (see next section). In both cases, each of the models predicts two possible qualitative behaviors for the biomass x . The expected information increment for the first experiment, however, is higher than the expected information increment for the second ($\Delta J = 0.5927$ versus $\Delta J = 0.3290$), as the predicted intervals overlap less.

The expression for ΔJ can be simplified in a number of cases. For instance, if all models predict for a given experiment e the same qualitative behavior, (3) can be reduced to

$$\Delta J(e) = \sum_{m_i \in M} p(m_i) \int_{y \in D} g_i^e(y, b) \ln \frac{g_i^e(y, b)}{g^e(y, b)} dy,$$

which is the criterion previously derived by [Vatcheva *et al.*, 2000].

On the other hand, if for a given e , each model predicts a different set of qualitative behaviors, we obtain:

$$\Delta J(e) = - \sum_{m_i \in M} p(m_i) \ln p(m_i),$$

which is the maximum value $\Delta J(e)$ can take.

The criterion (3) is easily generalizable to the case when each behavior b is characterized by more than one feature. In this case we have to substitute the probability distributions by joint probability distributions, and the integral by a multiple integral of the k behavioral features. For computational simplicity, we assume in this article that the behavioral features are independent.

The algorithms for the simulation of SQDEs, outlined in the previous section have been proven sound. That is, all possible predictions are derived from a given SQDE model. If

Contois	Droop	Caperon-Meyer
$\dot{x} = \mu(s)x - dx$	$\dot{x} = \mu(q)x - dx$	$\dot{x} = \mu(q)x - dx$
$\dot{s} = d(s_{in} - s) - \rho(s)x$	$\dot{q} = \rho(s) - \mu(q)q$	$\dot{q} = \rho(s) - \mu(q)q$
$\mu(s) = \mu_{max} \frac{s}{s + k_x}$	$\dot{s} = d(s_{in} - s) - \rho(s)x$	$\dot{s} = d(s_{in} - s) - \rho(s)x$
$\rho(s) = \frac{1}{y}\mu(s)$	$\mu(q) = \bar{\mu}(1 - \frac{k_q}{q})$	$\mu(q) = \mu_{max} \frac{q - k_q}{q - k_q + k_0}$
	$\rho(s) = \rho_{max} \frac{s}{s + k_s}$	$\rho(s) = \rho_{max} \frac{s}{s + k_s}$

$$y \in [0.15, 0.6], k_x \in [0.00014, 0.0167], \bar{\mu} \in [1.7, 2.3], k_q \in [1.6, 2.0], \mu_{max} \in [1.2, 1.6], \rho_{max} \in [9.25, 9.55], k_0 \in [2.0, 2.4]$$

Figure 3: Models for the growth of phytoplankton in a bioreactor. x is the amount of biomass per unit volume, s [$\mu\text{mol}/l$] the nutrient concentration, q [$\mu\text{mol}/l$] the internal quota. The Monod and Contois models assume constant growth yield y [$l/\mu\text{mol}$]. μ_{max} [day^{-1}] is the maximum growth rate of cells, and $\bar{\mu}$ [day^{-1}] a theoretical maximum growth rate obtained for infinite quota. k_s , k_x , and k_0 [$\mu\text{mol}/l$] are half-saturation constants, k_q [$\mu\text{mol}/l$] is the minimum cell quota, ρ_{max} [$\mu\text{mol}/l/\text{day}$] is the maximum uptake rate of nutrients. For all models $s_{in} \in [80, 120]$ [$\mu\text{mol}/l$] is the input nutrient concentration, and d [day^{-1}] the dilution rate to be controlled in the experiments. The initial conditions are $x_0 \in [36.0, 39.0]$, $s_0 \in [0.005, 0.01]$, $q_0 \in [2.4, 2.7]$ which are the equilibrium values reached for the initial dilution rate $d_0 = 0.4$.

the results obtained in the experiment are correct, this guarantees that a model will never be falsely rejected. However, these algorithms do not exclude spurious predictions. As a consequence, an experimental result may corroborate a model while it should be ruled out. Spurious predictions, therefore, may prolong the discrimination process.

3.3 Computation of behavior probabilities

In order to compute $\Delta J(e)$, the conditional behavior probabilities $p(b_j|m_i)$ must be estimated. We have adopted the following approach. Let v_l be a parameter or initial condition in model m_i , and let $\text{range}(v_l, b_j)$ be the interval value of v_l for which behavior b_j is obtained. Define

$$r(b_j|m_i) = \frac{\prod_l |\text{range}(v_l, b_j)|}{\prod_l |\bigcup_{b_k \in B_i^e} \text{range}(v_l, b_k)|}$$

$r(b_j|m_i)$ estimates the fraction of the interval volume of the model parameters which gives rise to b_j . The conditional probability of b_j is now given by normalizing the $r(b_j|m_i)$ s: $p(b_j|m_i) = r(b_j|m_i) / \sum_{b_k \in B_i^e} r(b_k|m_i)$.

Consider, for instance, the three behaviors given in Fig. 2. b_1 , b_2 and b_3 have been obtained for different subintervals of the interval ranges for k_s and y :

beh	k_s	y
b_1	[0.01, 0.158]	[0.3, 0.6]
b_2	[0.0236, 0.158]	[0.3, 0.6]
b_3	[0.01, 0.158]	[0.15, 0.488]

By using these values, the procedure outlined above gives $r(b_1) = 0.67$, $r(b_2) = 0.61$, $r(b_3) = 0.75$. Consequently, the behavior probabilities can be computed: $p(b_1) = 0.33$, $p(b_2) = 0.30$, $p(b_3) = 0.37$.

4 Application: phytoplankton growth

Understanding the regulation of phytoplankton growth is essential for predicting how life in the ocean may respond to climate changes. As these processes are difficult to study in the open sea, the growth conditions are recreated in the laboratory in a bioreactor.

A variety of models can be used to describe the growth of phytoplankton in a bioreactor. Which of these applies best in a given situation cannot be determined on *a priori* grounds. Therefore, experiments need to be performed to discriminate between the alternative models. Unfortunately, these experiments may take weeks to complete and are thus quite costly to perform.

We have applied the method of the previous section in the context of the microalgae *D. tertiolecta*, carried out by population biologists in a marine laboratory. Four alternative models to describe the system have been considered, which are shown in Figs. 1 and 3. The models make different assumptions about the nutrient consumption, the influence of the biomass on the growth rate of the population, and the relation between growth and uptake rates. The models are labeled after their originators: M [Monod, 1942], C [Contois, 1959], D [Droop, 1968], and CM [Caperon and Meyer, 1972].

Because of coarse and noisy data, and evolution of the system in the time frame of the experiment, precise numerical estimations for the values of the parameters cannot be obtained. This motivates the use of semi-quantitative models. The interval values for the parameters required in the SQDEs have been estimated by the biologists, based on the outcome of preliminary experiments (see Fig. 3).

In order to discriminate between the competing models, the value of the dilution rate d can be varied by the experimenter. Starting from an equilibrium, the dilution rate is changed and

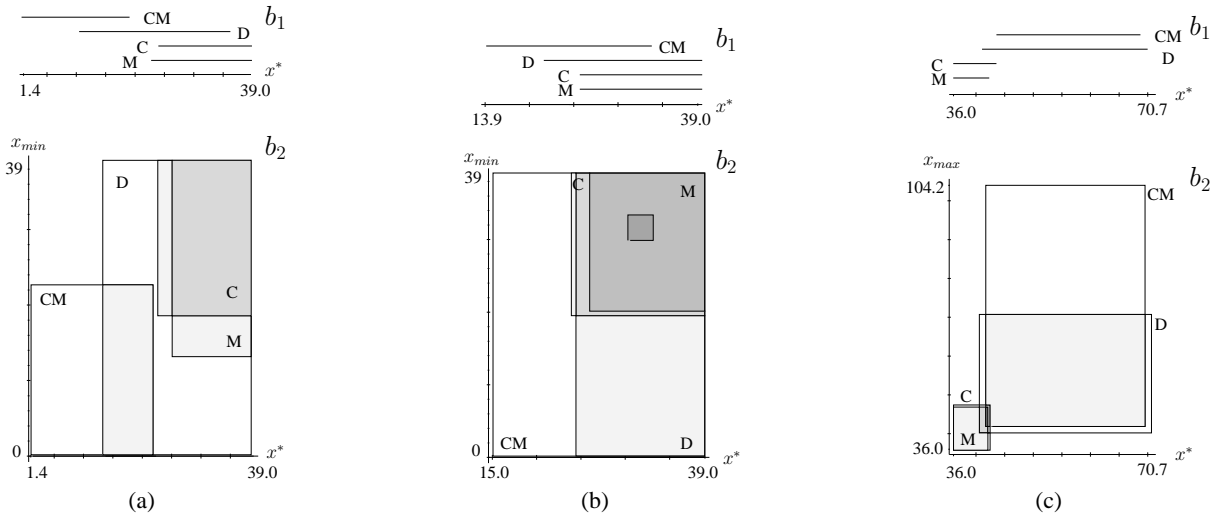


Figure 4: (a) Interval predictions of the four competing models (Figs. 1 and 3) for the behavioral features x^* (behavior b_1), and x^*, x_{min} (behavior b_2) in the predicted optimal discriminatory experiment ($d = 1.1$). (b) Interval predictions for x^* (behavior b_1), and x^*, x_{min} (behavior b_2) in the experiment that has been carried out ($d = 0.7$). The values for x^*, x_{min} measured in the experiment are also shown by a small rectangle. (c) Predictions for the features x^* (behavior b_1), and x^*, x_{max} (behavior b_2) in the next optimal experiment ($d = 0.1$). M, C, D, and CM stand for the Monod, Contois, Droop, and Caperon-Meyer models, respectively.

	$d = 0.1$	$d = 0.2$	$d = 0.3$	$d = 0.5$	$d = 0.6$
ΔJ	0.3204	0.3181	0.3519	0.3547	0.3323
	$d = 0.7$	$d = 0.8$	$d = 0.9$	$d = 1.0$	$d = 1.1$
ΔJ	0.3290	0.3710	0.4668	0.5093	0.5927

Table 1: Values for the expected information increment ΔJ for each of the dilution rate experiments.

	$d = 1.1$		$d = 0.7$		$d = 0.1$	
	$p(b_1)$	$p(b_2)$	$p(b_1)$	$p(b_2)$	$p(b_1)$	$p(b_2)$
M	0.47	0.53	0.47	0.53	0.50	0.50
C	0.47	0.53	0.47	0.53	0.51	0.49
D	0.57	0.43	0.57	0.43	0.54	0.46
CM	0.57	0.43	0.57	0.43	0.27	0.73

Table 2: Conditional probabilities of the behaviors b_1 and b_2 predicted by the four models for the experiments $d = 1.1$, $d = 0.7$, and $d = 0.1$ (see Fig. 4). $p(b_1)$ and $p(b_2)$ have been estimated using the approach in Sec. 3.3.

the transient behavior of the system towards a new equilibrium is observed. We have considered ten experiments, corresponding to equispaced values in the range $[0, 1.2]$: $d = 0.1, d = 0.2, \dots$. Taking into account 5% measurement uncertainty, the values of d become intervals.

The only variable that can be reliably measured in the course of the experiment is the biomass x . This determines the behavioral features that we have considered: the minimum and the maximum value of x (x_{min} and x_{max}), and the equilibrium value of x (x^*). In order to obtain the predicted values of the behavioral features required for the determi-

nation of the most informative experiment, the models have been simulated using the techniques in Sec. 2. For each experiment, all models predict multiple qualitative behaviors as a consequence of the large intervals for the parameter values. In total, four different behaviors for x are predicted. None of these behaviors is spurious, as we have been able to establish by comparing the predictions with the qualitative analysis of [Bernard and Gouzé, 1995].

Starting from the assumption that the models are equiprobable in the beginning, we have calculated the expected information increment (3) for each of the experiments (Table 1). The optimal discriminatory experiment is predicted to be $d = 1.1$. For this experiment, each of the four models predicts two behaviors, b_1 and b_2 , that differ with respect to the observable variable x . In b_1 the equilibrium of the system is reached asymptotically, whereas in b_2 , x reaches a minimum before the equilibrium is attained. Fig. 4(a) shows the interval predictions of the behavioral features for all four models, and Table 2 lists the corresponding conditional behavior probabilities. Notice that in b_1 only one behavioral feature applies (x^*), whereas in b_2 predictions for x^* and x_{min} need to be taken into account.

The experiment $d = 1.1$ has not been performed, but data for the suboptimal experiment $d = 0.7$ was available from an earlier study. The predictions of the behavioral features for this experiment are shown in Fig. 4(b) and the behavior probabilities are given in Table 2.

In the experiment $d = 0.7$, x was found to reach its equilibrium after passing through a minimum. This rules out b_1 . The measured values of the behavioral features, shown in Fig. 4(b), are $x_{min} = [29.2, 32.2]$, and $x^* = [30.5, 33.5]$. Using these results, the *a posteriori* probabilities of the models have been computed via Bayes' rule: $p(M) = 0.34$,

$p(C) = 0.33$, $p(D) = 0.21$, and $p(CM) = 0.12$. In addition, the measurements have allowed the parameter values to be refined by means of the constraint propagation algorithm in Q2 [Berleant and Kuipers, 1997].

The new model probabilities show that experiment $d = 0.7$ has not been very discriminating. Given the new model probabilities and parameter values, what would be the optimal experiment to perform next? The method advises that $d = 0.1$ be tried, as it has the highest expected information increment ($\Delta J = 0.7129$). The predicted values for the behavioral features, again for two behaviors, are shown in Fig. 4(c). Table 2 gives the corresponding behavior probabilities. The experiment $d = 0.1$ has not been performed yet (we recall that each experiment takes weeks to complete). Notice, however, that $d = 0.1$ is likely to rule out at least two of the four models due to the lack of overlap between $M - C$ and $D - CM$.

5 Discussion

We have proposed a method for the discrimination of semi-quantitative models of an experimental system. The method is based on an entropy criterion for the selection of the most informative experiment. The value of ΔJ for a particular experiment is calculated from the model predictions obtained through semi-quantitative simulation. The method generalizes upon a previous method [Vatcheva *et al.*, 2000], in that it can handle cases where the models predict multiple qualitative behaviors. This occurs in the case of the phytoplankton growth models, which predict the biomass to asymptotically approach its equilibrium value or to pass through a maximum or a minimum first.

The applicability of the method has been demonstrated by having it predict the most informative experiment to discriminate between four models of the growth of *D. tertiolecta* in a bioreactor. This has been achieved in the presence of several complicating factors, in particular the nonlinearity of the models, the crude estimations of the parameter values, and the difficulty to observe the behavior of the system. The discrimination of bioreactor models has been attempted before [Espie and Macchietto, 1989; Cooney and McDonald, 1995], but unlike the method discussed in this paper, these approaches require precise numerical data to be available, a requirement that usually cannot be fulfilled in practice.

Within AI, methods for model discrimination have been developed in the field of model-based diagnosis (e.g., [Struss, 1994; de Kleer, 1990]). Basically, these methods determine which inputs need to be applied to a faulty device, and which measurements need to be made, in order to optimally discriminate between a number of diagnoses. In comparison with the approach in this paper, these methods have been adapted to qualitative models. By considering only qualitative distinctions, however, one may fail to discriminate between alternative behaviors. Although two models may predict the same qualitative behavior, their (semi-)quantitative predictions may be different, as clearly shown in Fig. 4.

For the discrimination of the phytoplankton growth models, only one type of experiment was available, a change in the dilution rate. It should be emphasized, though, that the method is not restricted to parameter changes and may even

involve structural changes of the models. A limitation of the method, however, is that the set of experiments needs to be specified in advance. In the case of the dilution rate experiments, for instance, ten possible values from a continuous range have been selected. There is obviously no guarantee that the optimal value is included in the list of prespecified experiments. A subject for further research would therefore be to handle continuous ranges of experiments, and more generally, to move away from the *selection* of experiments to the *design* of experiments.

References

- [Berleant and Kuipers, 1997] D. Berleant and B. Kuipers. Qualitative and quantitative simulation: Bridging the gap. *Artif. Intell.*, 95:215–256, 1997.
- [Bernard and Gouzé, 1995] O. Bernard and J.-L. Gouzé. Transient behavior of biological loop models with application to the Droop model. *Math. Biosci.*, 127:19–43, 1995.
- [Caperon and Meyer, 1972] J. Caperon and J. Meyer. Nitrogen-limited growth of marine phytoplankton. I. Changes in population characteristics with steady-state growth rate. *Deep-Sea Res.*, 19:601–618, 1972.
- [Contois, 1959] D.E. Contois. Kinetics of bacterial growth: relationship between population density and specific growth rate of continuous cultures. *J. Gen. Microbiol.*, 21:40–50, 1959.
- [Cooney and McDonald, 1995] M.J. Cooney and K.A. McDonald. Optimal dynamic experiments for bioreactor model discrimination. *Appl. Microbiol. Biot.*, 43:826–837, 1995.
- [de Kleer, 1990] J. de Kleer. Using crude probability estimates to guide diagnosis. *Artif. Intell.*, 45:381–391, 1990.
- [Droop, 1968] M.R. Droop. Vitamin B12 and marine ecology. IV. The kinetics of uptake growth and inhibition in *Monochrysis lutheri*. *J. Mar. Biol. Assoc.*, 48(3):689–733, 1968.
- [Espie and Macchietto, 1989] D. Espie and S. Macchietto. The optimal design of dynamic experiments. *AIChE*, 35(2):223–229, 1989.
- [Fedorov, 1972] V.V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972.
- [Kuipers, 1994] B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.
- [Monod, 1942] J. Monod. *Recherches sur la croissance des cultures bactériennes*. Herman and Cie, 1942.
- [Struss, 1994] P. Struss. Testing for discrimination of diagnoses. In *Working notes DX-94*, 1994.
- [Vatcheva and de Jong, 1999] I. Vatcheva and H. de Jong. Semi-quantitative comparative analysis. In *Proceedings IJCAI'99*, pages 1034–1040, 1999.
- [Vatcheva *et al.*, 2000] I. Vatcheva, H. de Jong, and N.J.I. Mars. Selection of perturbation experiments for model discrimination. In *Proceedings ECAI-2000*, pages 191–195, 2000.

KNOWLEDGE REPRESENTATION AND REASONING

TEMPORAL REASONING

A Complete Classification of Complexity in Allen's Algebra in the Presence of a Non-Trivial Basic Relation

Andrei Krokhin* and Peter Jeavons*

Oxford University Computing Laboratory
Wolfson Building, Parks Road, OX1 3QD Oxford, UK
{andrei.krokhin, peter.jeavons}@comlab.ox.ac.uk

Peter Jonsson†

Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
email: peter.jonsson@ida.liu.se

Abstract

We study fragments of Allen's algebra that contain a basic relation distinct from the equality relation. We prove that such a fragment is either NP-complete or else contained in some already known tractable subalgebra. We obtain this result by giving a new uniform description of known maximal tractable subalgebras and then systematically using an algebraic technique for description of maximal subalgebras with a given property. This approach avoids the need for extensive computer-assisted search.

1 Introduction

Allen's interval algebra [Allen, 1983] is one of the best established formalisms for temporal reasoning. Such reasoning is an important task in many areas of computer science and artificial intelligence (see, e.g., [Golumbic and Shamir, 1993; Nökel, 1991]). Allen's algebra allows us to specify qualitative information about time intervals. This algebra has also become the kernel of some other formalisms [Meiri, 1996; Angelsmark and Jonsson, 2000]. The interval algebra and some of its extensions are closely related with a number of interval-based temporal logics used for real-time system specification [Bellini *et al.*, 2000].

The basic satisfiability problem in Allen's algebra is NP-complete [Vilain *et al.*, 1989]), so it is unlikely that efficient algorithms exist for reasoning in the full algebra. The computational difficulty has motivated the search for effective heuristics, e.g., [Ladkin and Reinefeld, 1992], and the study of the complexity of fragments of the algebra, e.g., [Golumbic and Shamir, 1993; Nebel and Bürckert, 1995].

*This research is partially supported by the UK EPSRC grants GR/R22704 and GR/M12926.

†This research is partially supported by the Swedish Research Council for Engineering Sciences (TFR) under grants 97-301 and 2000-361.

In this paper we follow the second approach, and we also assume throughout that $P \neq NP$. Research in this direction has focused on identifying *maximal* tractable fragments, i.e., fragments which cannot be extended without losing tractability. So far, eighteen maximal tractable fragments of the algebra have been identified [Drakengren and Jonsson, 1997a; 1997b; Nebel and Bürckert, 1995]. It is still unknown whether any others exist.

A complete classification of complexity within a certain large part of Allen's algebra was obtained in [Drakengren and Jonsson, 1998]. This result (as well as most similar results) was achieved by computer-assisted exhaustive search. However, it was noted in [Drakengren and Jonsson, 1998] that, for further progress, theoretical studies of the structure of Allen's algebra are required. There are some theoretical investigations of the structure of Allen's algebra, (see, e.g., [Hirsch, 1996; Ladkin and Maddux, 1994]). However they generally allow more operations on relations than originally used in [Allen, 1983], which makes them inappropriate for classifying complexity within the interval algebra. In this paper we systematically use an algebraic method that is similar to the approach taken in [Ligozat, 1998].

The first novel element in our approach is a new uniform description for all of the maximal tractable subalgebras of Allen's algebra which have already been identified (Table 2). Then we exhibit several new small NP-complete fragments of the algebra, including the first examples of NP-complete singletons (Proposition 2). Finally, we fully exploit the algebraic properties of Allen's algebra by importing a technique from general algebra. This technique has been used in many other contexts to obtain a description of maximal subalgebras of a given algebra with a given property (e.g., [Szendrei, 1995]). Here, for the first time, we systematically apply this technique to Allen's algebra to obtain a complete classification of the complexity of those fragments of the algebra containing a non-trivial basic relation (Theorem 1). Our method also provides (as a by-product) a new elementary proof of maximality of ten tractable subalgebras. In previous papers this proof of maximality has required extensive computer-aided case anal-

Basic relation	Example	Endpoints
x precedes y	p	$x^+ < y^-$
y preceded by x	p^{-1}	$x^+ < y^-$
x meets y	m	$x^+ = y^-$
y met by x	m^{-1}	$x^+ = y^-$
x overlaps y	o	$x^- < y^- < x^+$,
y overl. by x	o^{-1}	$x^+ < y^+$
x during y	d	$x^- > y^-$,
y includes x	d^{-1}	$x^+ < y^+$
x starts y	s	$x^- = y^-$,
y started by x	s^{-1}	$x^+ < y^+$
x finishes y	f	$x^+ = y^+$,
y finished by x	f^{-1}	$x^- > y^-$
x equals y	\equiv	$x^- = y^-$,
		$x^+ = y^+$

Table 1: The thirteen basic relations. The endpoint relations $x^- < x^+$ and $y^- < y^+$ that are valid for all relations have been omitted.

ysis [Drakengren and Jonsson, 1997a; 1997b].

The paper is organized as follows: in Section 2 we give the basic definitions of Allen’s algebra and present the known maximal tractable subalgebras in the new form. Section 3 contains the new classification result, and Section 4 contains a brief discussion of future extensions.

2 Allen’s Algebra

Allen’s interval algebra [Allen, 1983] is based on the notion of *relations between intervals*. An interval x is represented as a tuple (x^-, x^+) of real numbers with $x^- < x^+$, denoting the left and right endpoints of the interval, respectively. The possible relations between intervals are the 2^{13} possible unions of 13 *basic interval relations*, which are shown in Table 1. Relations between intervals will also be written as collections of basic relations so, for instance, the union of the basic relations p, m and f^{-1} is written $(p m f^{-1})$. Allen’s algebra \mathcal{A} consists of the 2^{13} possible relations between intervals together with the operations *converse* \cdot^{-1} , *intersection* \cap and *composition* \circ which are defined as follows:

$$\forall x, y : xr^{-1}y \Leftrightarrow yrx$$

$$\forall x, y : x(r \cap s)y \Leftrightarrow xry \ \& \ xsy$$

$$\forall x, y : x(r \circ s)y \Leftrightarrow \exists z : (xrz \ \& \ zsy)$$

It follows that the converse of $r = (b_1 \dots b_n)$ is equal to $(b_1^{-1} \dots b_n^{-1})$. The intersection of two relations can be expressed as the usual set-theoretic intersection. Using the definition of composition, it can be derived that

$$(b_1 \dots b_n) \circ (b'_1 \dots b'_m) = \bigcup \{b_i \circ b'_j \mid i \leq n, j \leq m\}.$$

All algebraic calculations given in Section 3 can be done by hand; however, a simple table showing the results of the composition operation on all pairs of basic relations (which can be found, e.g., in [Ladkin and Maddux, 1994]) makes such calculations considerably easier.

The problem of *satisfiability* (\mathcal{A} -SAT) of a set of interval variables with relations between them is that of deciding whether there exists an assignment of intervals on the real line for the interval variables, such that all of the relations between the intervals are satisfied. This is defined as follows.

Definition 1 Let $X \subseteq \mathcal{A}$ be a set of interval relations. An instance I of \mathcal{A} -SAT(X) is a set, V , of variables and a set of constraints of the form xry where $x, y \in V$ and $r \in X$. The question is whether I is satisfiable, i.e., whether there exists a function, f , from V to the set of all intervals such that $f(x) r f(y)$ holds for every constraint xry in I .

If there exists a polynomial-time algorithm solving all instances of \mathcal{A} -SAT(X) then we say that X is tractable. On the other hand, if \mathcal{A} -SAT(X) is NP-complete then we say that X is NP-complete. Since the problem \mathcal{A} -SAT(\mathcal{A}) is NP-complete [Vilain *et al.*, 1989], there arises the question of description of tractable subsets of Allen’s algebra.

Subsets of \mathcal{A} that are closed under the operations of intersection, converse and composition are said to be *subalgebras*. For a given subset X of \mathcal{A} , the smallest subalgebra containing X is called the subalgebra *generated* by X and is denoted by $\langle X \rangle$. It is easy to see that $\langle X \rangle$ is obtained from X by adding all relations that can be obtained from the relations in X by using the three operations of \mathcal{A} .

It is known [Nebel and Bürckert, 1995], and easy to prove, that, for every $X \subseteq \mathcal{A}$, the problem \mathcal{A} -SAT($\langle X \rangle$) is polynomially equivalent to \mathcal{A} -SAT(X). Therefore, to classify the complexity of all subsets of \mathcal{A} it is only necessary to consider *subalgebras* of \mathcal{A} . Obviously, adding relations to a subalgebra can only increase the complexity of the corresponding satisfiability problem. Thus, since \mathcal{A} is finite, the problem of describing tractability in \mathcal{A} can be reduced to the problem of describing the *maximal* tractable subalgebras in \mathcal{A} , i.e., subalgebras that cannot be extended without losing tractability.

The known maximal tractable subalgebras [Drakengren and Jonsson, 1997a; 1997b; Nebel and Bürckert, 1995] are presented in Table 2. In our proofs as well as in Table 2 we use the symbol \pm , which should be interpreted as follows. A condition involving \pm means the conjunction of two conditions: one corresponding to $+$ and one corresponding to $-$. For example, condition $(o)^{\pm} \subseteq r \Leftrightarrow (d)^{\pm} \subseteq r$ means that both $(o) \subseteq r \Leftrightarrow (d) \subseteq r$ and $(o^{-1}) \subseteq r \Leftrightarrow (d^{-1}) \subseteq r$ hold. The main advantage of using the \pm symbol is conciseness: in any subalgebra of \mathcal{A} , the $+$ and the $-$ conditions are satisfied (or not satisfied) simultaneously, and, therefore only one of them needs to be verified.

In previous papers, the subalgebras from Table 2 were defined in other ways. However, in all cases except for \mathcal{H} , it is very straightforward to verify that our definitions are equivalent to the original ones. The subalgebra \mathcal{H} was originally defined as the ‘ORD-Horn algebra’ [Nebel and Bürckert, 1995], but has also been characterized as the algebra of ‘pre-convex’ relations (see, e.g., [Ligozat, 1998]). Using the latter description it is not hard to show that our definition of \mathcal{H} is equivalent. Finally, we note that there is one more known maximal tractable subalgebra that is not included in Table 2. That subalgebra is defined as $\{\emptyset\} \cup \{r \in \mathcal{A} \mid (\equiv) \subseteq r\}$ and the only basic relation it contains is (\equiv) .

$$\begin{aligned}
\mathcal{S}_p &= \{r \mid r \cap (\text{pmod}^{-1}f^{-1})^{\pm 1} \neq \emptyset \Rightarrow (p)^{\pm 1} \subseteq r\} & \mathcal{E}_p &= \{r \mid r \cap (\text{pmods})^{\pm 1} \neq \emptyset \Rightarrow (p)^{\pm 1} \subseteq r\} \\
\mathcal{S}_d &= \{r \mid r \cap (\text{pmod}^{-1}f^{-1})^{\pm 1} \neq \emptyset \Rightarrow (d^{-1})^{\pm 1} \subseteq r\} & \mathcal{E}_d &= \{r \mid r \cap (\text{pmods})^{\pm 1} \neq \emptyset \Rightarrow (d)^{\pm 1} \subseteq r\} \\
\mathcal{S}_o &= \{r \mid r \cap (\text{pmod}^{-1}f^{-1})^{\pm 1} \neq \emptyset \Rightarrow (o)^{\pm 1} \subseteq r\} & \mathcal{E}_o &= \{r \mid r \cap (\text{pmods})^{\pm 1} \neq \emptyset \Rightarrow (o)^{\pm 1} \subseteq r\} \\
\mathcal{A}_1 &= \{r \mid r \cap (\text{pmod}^{-1}f^{-1})^{\pm 1} \neq \emptyset \Rightarrow (s^{-1})^{\pm 1} \subseteq r\} & \mathcal{B}_1 &= \{r \mid r \cap (\text{pmods})^{\pm 1} \neq \emptyset \Rightarrow (f^{-1})^{\pm 1} \subseteq r\} \\
\mathcal{A}_2 &= \{r \mid r \cap (\text{pmod}^{-1}f^{-1})^{\pm 1} \neq \emptyset \Rightarrow (s)^{\pm 1} \subseteq r\} & \mathcal{B}_2 &= \{r \mid r \cap (\text{pmods})^{\pm 1} \neq \emptyset \Rightarrow (f)^{\pm 1} \subseteq r\} \\
\mathcal{A}_3 &= \{r \mid r \cap (\text{pmod}f)^{\pm 1} \neq \emptyset \Rightarrow (s)^{\pm 1} \subseteq r\} & \mathcal{B}_3 &= \{r \mid r \cap (\text{pmod}^{-1}s^{-1})^{\pm 1} \neq \emptyset \Rightarrow (f^{-1})^{\pm 1} \subseteq r\} \\
\mathcal{A}_4 &= \{r \mid r \cap (\text{pmod}f^{-1})^{\pm 1} \neq \emptyset \Rightarrow (s)^{\pm 1} \subseteq r\} & \mathcal{B}_4 &= \{r \mid r \cap (\text{pmod}^{-1}s)^{\pm 1} \neq \emptyset \Rightarrow (f^{-1})^{\pm 1} \subseteq r\} \\
\mathcal{E}^* &= \left\{ r \mid \begin{array}{l} 1) r \cap (\text{pmod})^{\pm 1} \neq \emptyset \Rightarrow (s)^{\pm 1} \subseteq r, \text{ and} \\ 2) r \cap (\text{ff}^{-1}) \neq \emptyset \Rightarrow (\equiv) \subseteq r \end{array} \right\} & \mathcal{S}^* &= \left\{ r \mid \begin{array}{l} 1) r \cap (\text{pmod}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (f^{-1})^{\pm 1} \subseteq r, \text{ and} \\ 2) r \cap (\text{ss}^{-1}) \neq \emptyset \Rightarrow (\equiv) \subseteq r \end{array} \right\} \\
\mathcal{H} &= \left\{ r \mid \begin{array}{l} 1) r \cap (\text{os})^{\pm 1} \neq \emptyset \ \& \ r \cap (\text{o}^{-1}f)^{\pm 1} \neq \emptyset \Rightarrow (d)^{\pm 1} \subseteq r, \text{ and} \\ 2) r \cap (\text{ds})^{\pm 1} \neq \emptyset \ \& \ r \cap (\text{d}^{-1}f^{-1})^{\pm 1} \neq \emptyset \Rightarrow (o)^{\pm 1} \subseteq r, \text{ and} \\ 3) r \cap (\text{pm})^{\pm 1} \neq \emptyset \ \& \ r \not\subseteq (\text{pm})^{\pm 1} \Rightarrow (o)^{\pm 1} \subseteq r \end{array} \right\}
\end{aligned}$$

Table 2: Maximal tractable subalgebras of Allen’s algebra.

3 Main Result

Throughout the paper, \mathcal{S} denotes a subalgebra of \mathcal{A} . We denote by $\text{bas}(\mathcal{S})$ the set of basic relations in \mathcal{S} . Our result is an extension of the following proposition which was proved in [Drakengren and Jonsson, 1998].

Proposition 1

- 1) Let \mathcal{S} be a subalgebra of \mathcal{A} with $|\text{bas}(\mathcal{S})| > 3$. Then \mathcal{S} is tractable if and only if it is contained in one of the following 7 algebras: $\mathcal{S}_p, \mathcal{S}_d, \mathcal{S}_o, \mathcal{E}_p, \mathcal{E}_d, \mathcal{E}_o$, and \mathcal{H} . Otherwise \mathcal{S} is NP-complete.
- 2) Let \mathcal{S} be a subalgebra of \mathcal{A} such that $(m) \in \mathcal{S}$ or $(p) \in \mathcal{S}$. If \mathcal{S} is tractable then $\mathcal{S} \subseteq \mathcal{S}_p$, or $\mathcal{S} \subseteq \mathcal{E}_p$, or $\mathcal{S} \subseteq \mathcal{H}$. Otherwise \mathcal{S} is NP-complete.

We improve this result by showing that any so far unknown tractable subalgebra can contain only a *trivial* basic relation. Throughout the paper, by a *trivial* basic relation we mean the equality relation \equiv .

Theorem 1 Suppose \mathcal{S} is a subalgebra of \mathcal{A} containing a non-trivial basic relation. Then \mathcal{S} is tractable if and only if it is contained in one of the 17 algebras listed in Table 2. Otherwise \mathcal{S} is NP-complete.

We can assume, without loss of generality, that \mathcal{S} contains the relation (\equiv) , since it is easy to show that \mathcal{S} and $\mathcal{S} \cup \{(\equiv)\}$ have the same complexity (up to polynomial-time equivalence). This implies that the size of $\text{bas}(\mathcal{S})$ is odd since \mathcal{S} is closed under converse. By combining Proposition 1(1) and the previous observation, we may consider only the case $|\text{bas}(\mathcal{S})| = 3$. By Proposition 1(2), it suffices to consider the cases where $\text{bas}(\mathcal{S})$ is one of the following sets: $\{\equiv, s, s^{-1}\}$, $\{\equiv, f, f^{-1}\}$, $\{\equiv, d, d^{-1}\}$, or $\{\equiv, o, o^{-1}\}$.

The rest of this section is organized as follows. In Subsection 3.1 we describe some NP-complete sets of relations which we use in our proofs later. In Subsection 3.2, we give a complete proof of Theorem 1 for the cases where $\text{bas}(\mathcal{S}) = \{\equiv, s, s^{-1}\}$ and $\text{bas}(\mathcal{S}) = \{\equiv, f, f^{-1}\}$. Finally, in Subsection 3.3 we give a sketch of a proof of Theorem 1 for the cases where $\text{bas}(\mathcal{S}) = \{\equiv, d, d^{-1}\}$ and $\text{bas}(\mathcal{S}) = \{\equiv, o, o^{-1}\}$.

Finally, we observe that when \mathcal{S} is one of the algebras \mathcal{E}^* , \mathcal{S}^* , \mathcal{A}_i , or \mathcal{B}_i ($1 \leq i \leq 4$), then $\text{bas}(\mathcal{S})$ is either $\{\equiv, s, s^{-1}\}$ or $\{\equiv, f, f^{-1}\}$. With the help of computer-aided exhaustive search, it was proved in [Drakengren and Jonsson, 1997a; 1997b] that these ten algebras are maximal tractable subalgebras of Allen’s algebra. Since we do not use computer-aided results in Subsection 3.2, our results provide a new elementary proof of maximality for these algebras.

Throughout this section we shall assume that a non-trivial relation is also non-empty; this will cause no confusion.

3.1 NP-completeness results

This subsection contains the NP-completeness proofs that we need in the sequel. Our reductions are based on the NP-complete problem BETWEENNESS [Garey and Johnson, 1979]:

INSTANCE: A finite set A and a collection T of ordered triples (a, b, c) of distinct elements from A .

QUESTION: Is there a total ordering $<$ on A such that for each $(a, b, c) \in T$, we have either $a < b < c$ or $c < b < a$?

Let us fix relations $R, R_1, R_2 \in \mathcal{A}$. Then we define $\Gamma(a, b, c, x, y)$ to be the following problem instance over the variables $\{a, b, c, x, y\}$:

$$\{xR_1a, xR_1b, xR_2c, yR_2a, yR_1b, yR_1c\}.$$

Further, define $\Gamma_1 = \Gamma(a, b, c, x, y) \cup \{aRb, bRc\}$, $\Gamma_2 = \Gamma(a, b, c, x, y) \cup \{bRa, bRc, aR \cup R^{-1}c\}$, and $\Gamma_3 = \Gamma(a, b, c, x, y) \cup \{aRb, cRb, aR \cup R^{-1}c\}$.

Lemma 1 Let $R \in \{(p), (d), (o), (s)\}$, $R_1, R_2 \in \mathcal{A}$, and let $\Gamma_1, \Gamma_2, \Gamma_3$ be as above. If Γ_1 is satisfiable while Γ_2 and Γ_3 are not, then $\{R \cup R^{-1}, R_1, R_2\}$ is NP-complete.

Proof. Polynomial-time reduction from BETWEENNESS. Let (A, T) be an arbitrary instance of the problem and construct an instance I of $\mathcal{A}\text{-SAT}(\{R \cup R^{-1}, R_1, R_2\})$ as follows:

- (1) for each pair of distinct elements $a, b \in A$, add the constraint $aR \cup R^{-1}b$ to I ; and

- (2) for each triple $(a, b, c) \in T$, introduce two fresh variables x, y and add $\Gamma(a, b, c, x, y)$ to I .

It is a routine verification to show that I is satisfiable if and only if (A, T) has a solution. \square

The restriction $R \in \{(p), (d), (o), (s)\}$ is imposed to ensure that we get a natural total ordering in the reduction. Many other relations such as (dsf) have this property but there are also relations without this property, e.g., (\equiv) and (mm^{-1}) .

Suppose $X \subseteq \mathcal{A}$ and I is an instance of $\mathcal{A}\text{-SAT}(X)$. Let variables x, y be involved in I . Further, let $r \in \mathcal{A}$ be the relation defined as follows. A basic relation r' is included in r if and only if the instance obtained from I by adding the constraint $xx'y$ is satisfiable. In this case, we say that r is *derived* from X . It can be easily checked that the problems $\mathcal{A}\text{-SAT}(X)$ and $\mathcal{A}\text{-SAT}(X \cup \{r\})$ are polynomially equivalent. Looking now at the definition of the three operations of the interval algebra, it is easy to see that, generally, one can derive more relations from a given $X \subseteq \mathcal{A}$ than one can generate using these operations. On the other hand, derivation is essentially harder to manage while the operations of Allen's algebra give us the advantage of employing algebraic techniques. Therefore, in this paper, we use derivation only once in the proof of the following statement.

Proposition 2 *The following subsets of \mathcal{A} are NP-complete:*

- 1) $\{(d), (oo^{-1})\}$, $\{(d^{-1}), (pp^{-1})\}$, and $\{(o), (dd^{-1})\}$.
- 2) $\{r\}$ with $(ods^{-1}) \subseteq r \subseteq (pmodss^{-1}ff^{-1})$.

Proof. For part (1), apply Lemma 1 with $R = (o)$, $R_1 = (d)$, $R_2 = (oo^{-1})$, or with $R = (p)$, $R_1 = (d^{-1})$, $R_2 = (pp^{-1})$, or with $R = (d)$, $R_1 = (o)$, $R_2 = (dd^{-1})$, respectively.

To prove part (2), let r_3 be the union of all basic relations except for \equiv and s^{-1} , and consider the instance $\Gamma_4 = \{arb, brc, cra, drb, drc\}$ over the variables a, b, c, d . In the cases when $r = (ods^{-1})$ or $r = (pmodss^{-1}ff^{-1})$, it can be shown that $\Gamma_4 \cup \{dr'a\}$ is satisfiable for every basic relation $r' \subseteq r_3$ but not satisfiable for any other choice of r' . Hence, we can derive r_3 from r , and we can also derive the relations $r_4 = r \cap r_3 = r - (s^{-1})$ and $r_5 = r_4 \circ r_4$. It is easy to check that

$$(pmodss) \subseteq r_5 \subseteq (pmodsf^{-1})$$

which implies that $r_5 \cap r^{-1} = (s)$. Furthermore, we have $(s) \circ (s^{-1}) = (\equiv ss^{-1})$, and $r \circ r$ is the disequality relation, so the relation (ss^{-1}) can be obtained from the relation r . If $R = (s)$, $R_1 = r^{-1}$ and $R_2 = r$, then these relations satisfy the conditions of Lemma 1 so $\{(ss^{-1}), r^{-1}, r\}$ is NP-complete. Since all of these relations can be derived from the single relation r , it follows that $\{r\}$ is NP-complete. \square

3.2 The s/f case

We will show that if $bas(\mathcal{S}) = \{\equiv, s, s^{-1}\}$, then either \mathcal{S} is NP-complete or \mathcal{S} is contained in one of the subalgebras $\mathcal{S}_p, \mathcal{S}_d, \mathcal{S}_o, \mathcal{E}^*, \mathcal{H}$, or in one of \mathcal{A}_i , $1 \leq i \leq 4$. By using the obvious symmetry between the relations (s) and (f) , it immediately follows that if $bas(\mathcal{S}) = \{\equiv, f, f^{-1}\}$, then either \mathcal{S} is NP-complete or contained in one of $\mathcal{E}_p, \mathcal{E}_d, \mathcal{E}_o, \mathcal{S}^*, \mathcal{H}$,

or \mathcal{B}_i , $1 \leq i \leq 4$. For the remainder of this subsection, it is assumed that $bas(\mathcal{S}) = \{\equiv, s, s^{-1}\}$.

Given a relation r , we write r^* to denote the relation $r \cap r^{-1}$. Throughout the proofs we use the obvious fact that if $r_1 \subseteq r_2$ then, for any r , we have $r \circ r_1 \subseteq r \circ r_2$ and $r_1 \circ r \subseteq r_2 \circ r$. If b is a basic relation then r_b denotes the least relation $r \in \mathcal{S}$ such that $(b) \subseteq r$, i.e. the intersection of all $r \in \mathcal{S}$ with this property. It is easy to show that the relation r_b exists in \mathcal{S} for every basic relation b , except for two very simple cases: when every $r \in \mathcal{S}$ satisfies exactly one of conditions $r \subseteq (\equiv ss^{-1})$ and $r \subseteq (\equiv ff^{-1})$. In both cases, we have $\mathcal{S} \subseteq \mathcal{H}$.

Lemma 2 *If \mathcal{S} contains the relation (od) , then every relation in \mathcal{S} satisfies condition 3) of \mathcal{H} .*

Proof. Arbitrarily choose $r \in \mathcal{S}$. Since $(od) \in \mathcal{S}$, it follows that $r_d = r_o = (od)$ and $(o)^{\pm 1} \subseteq r \Leftrightarrow (d)^{\pm 1} \subseteq r$. Furthermore, $r_1 = (s) \circ (od) = (pmod) \in \mathcal{S}$ so $r_p \subseteq r_1$ and $r_m \subseteq r_1$. Since $(pm) \circ (pm) = (p)$, we have $r_p \not\subseteq (pm)$ and $r_m \not\subseteq (pm)$, which implies that $r_p \cap (od) \neq \emptyset$ and $r_m \cap (od) \neq \emptyset$. Now it follows that $(od) \subseteq r_p$ and $(od) \subseteq r_m$. Thus, if $r \cap (pm) \neq \emptyset$, then $(od) \subseteq r$ and r satisfies condition 3) of \mathcal{H} . \square

Lemma 3 *If \mathcal{S} contains a non-trivial relation r' with $r' \subseteq (\equiv pp^{-1}mm^{-1}ff^{-1})$ then \mathcal{S} is included in one of $\mathcal{H}, \mathcal{E}^*,$ or \mathcal{S}_p , or else \mathcal{S} is NP-complete.*

Proof. Case 1. $r' = (ff^{-1})$

Since $(ff^{-1}) \circ (s) = (od) \in \mathcal{S}$, it follows that any $r \in \mathcal{S}$ satisfies condition 3) of \mathcal{H} , by Lemma 2. Now it follows that, for any $r \in \mathcal{S}$, we have $(o)^{\pm 1} \subseteq r \Leftrightarrow (d)^{\pm 1} \subseteq r$ and also $(f) \subseteq r \Leftrightarrow (f^{-1}) \subseteq r$

Suppose that $\mathcal{S} \not\subseteq \mathcal{H}$, i.e. some $r \in \mathcal{S}$ fails to satisfy condition 1) or condition 2) of \mathcal{H} . Then, using the facts from the previous paragraph, it can be shown that we can choose r so that $(s^{-1}ff^{-1}) \subseteq r \subseteq (\equiv pmodss^{-1}ff^{-1})$ or $(ods^{-1}) \subseteq r \subseteq (\equiv pmodss^{-1}ff^{-1})$. In both cases, composing the relations with (s) from the left we get

$$(\equiv pmodss^{-1}) \subseteq (s) \circ r \subseteq (\equiv pmodss^{-1}).$$

Therefore $(\equiv pmodss^{-1}) \in \mathcal{S}$, and $(pmodss^{-1})$ can be obtained as

$$(\equiv pmodss^{-1}) \cap ((\equiv pmodss^{-1})^{-1} \circ (od)^{-1}).$$

Then \mathcal{S} is NP-complete, by Proposition 2(2).

Case 2. $r' \subseteq (\equiv ff^{-1})$.

Composing r' and its inverse we get $(\equiv ff^{-1})$, so we may assume that $r' = (\equiv ff^{-1})$. If some relation $r_2 \in \mathcal{S}$ fails to satisfy condition 2) of \mathcal{E}^* , then $r_2 \cap r'$ is either one of (f) and (f^{-1}) , which is impossible under our assumptions, or (ff^{-1}) going back to Case 1. Suppose now that each $r \in \mathcal{S}$ satisfies condition 2) of \mathcal{E}^* .

We have $r' \circ (s) = (ods) \in \mathcal{S}$. If (od) belongs to \mathcal{S} then $r' \cap ((s^{-1}) \circ (od)) = (ff^{-1}) \in \mathcal{S}$, which implies $\mathcal{S} \subseteq \mathcal{H}$. Otherwise we have $r \cap (od)^{\pm 1} \neq \emptyset \Rightarrow (s)^{\pm 1} \subseteq r$ for every $r \in \mathcal{S}$. Assume a relation $r_3 \in \mathcal{S}$ does not satisfy condition 1) of \mathcal{E}^* , that is $r_3 \cap (pmod) \neq \emptyset$ and $(s) \not\subseteq r_3$. Since $(pmodss) = (s) \circ r' \in \mathcal{S}$, we have $r_4 = r_3 \cap (pmodss) \in \mathcal{S}$

and $r_4 \subseteq (\text{pm})$. This implies $r_4 \circ r_4 = (\text{p})$, a contradiction. Therefore the relations in \mathcal{S} must satisfy both conditions of \mathcal{E}^* , that is $\mathcal{S} \subseteq \mathcal{E}^*$.

Case 3. $r' \subseteq (\equiv \text{mm}^{-1}\text{ff}^{-1})$ and $r' \cap (\text{mm}^{-1}) \neq \emptyset$.

Without loss of generality we may assume that $(\text{m}) \subseteq r'$. Then the relation $(r' \cap (r' \circ (\text{s}^{-1})))$ is equal either to (mm^{-1}) or to (m) . In the former case it follows that the relation $(\text{m}) = (\text{mm}^{-1}) \cap ((\text{mm}^{-1}) \circ (\text{s}^{-1}))$ belongs to \mathcal{S} as well. A contradiction.

Case 4. $r' \cap (\text{pp}^{-1}) \neq \emptyset$.

We may assume, without loss of generality, that $(\text{p}) \subseteq r'$. Then $(\text{p}) \subseteq (\text{s}) \circ r' \subseteq (\text{pp}^{-1}\text{mm}^{-1}\text{ods})$. Furthermore, $(\text{p}) \subseteq r_5 = r' \cap ((\text{s}) \circ r') \subseteq (\text{pp}^{-1}\text{mm}^{-1})$, and we have $(\text{p}) \subseteq r_6 = r_5 \circ (\text{s}^{-1}) \subseteq (\text{pp}^{-1}\text{m})$. If $(\text{p}^{-1}) \not\subseteq r_6$ then $r_6 \circ r_6 = (\text{p}) \in \mathcal{S}$, a contradiction. Otherwise we have $(\text{pp}^{-1}) = r_6^* \in \mathcal{S}$. Then $(\text{p}) \subseteq r \Leftrightarrow (\text{p}^{-1}) \subseteq r$ holds for every $r \in \mathcal{S}$.

We have $(\text{s}^{-1}) \circ (\text{pp}^{-1}) = (\text{pp}^{-1}\text{mod}^{-1}\text{f}^{-1}) \in \mathcal{S}$. For every non-empty $r_7 \subseteq (\text{modf}^{-1})$, we have $((\text{s}) \circ r_7) \cap (\text{pp}^{-1}) = (\text{p})$. Therefore no such r_7 belongs to \mathcal{S} . We conclude that, for any $r \in \mathcal{S}$, if $r \cap (\text{mod}^{-1}\text{f}^{-1}) \neq \emptyset$ then $(\text{p}) \subseteq r$, which means that $\mathcal{S} \subseteq \mathcal{S}_\text{p}$. \square

We assume further that, for every non-trivial $r \in \mathcal{S}$, we have $r \cap (\text{oo}^{-1}\text{dd}^{-1}\text{ss}^{-1}) \neq \emptyset$. In the next four lemmas we consider the case when \mathcal{S} contains a non-trivial relation r' with $r' \cap (\text{ss}^{-1}) = \emptyset$. In view of the assumption just made we may consider that r' is either r_d or r_o . It is not hard to verify that these four lemmas indeed cover all possible cases for r_d and r_o .

Lemma 4 Suppose $r_\text{d} \cap (\text{ss}^{-1}) = \emptyset$. If $r_\text{d} \cap (\text{oo}^{-1}\text{dd}^{-1}) = (\text{d})$ or $(\text{dd}^{-1}) \subseteq r_\text{d}$ then $\mathcal{S} \subseteq \mathcal{S}_\text{d}$.

Proof. **Case 1.** $r_\text{d} \cap (\text{oo}^{-1}\text{dd}^{-1}) = (\text{d})$.

By assumption, the relation r_d satisfies condition (d) $\subseteq r_\text{d} \subseteq (\equiv \text{pp}^{-1}\text{mm}^{-1}\text{dff}^{-1})$. Let r_1 be calculated as $r_\text{d} \cap ((\text{s}) \circ r_\text{d})$. Then we have $(\text{d}) \subseteq r_1 \subseteq (\text{pp}^{-1}\text{mm}^{-1}\text{d})$. By minimality of r_d we get $(\text{d}) \subseteq r_\text{d} \subseteq (\text{pp}^{-1}\text{mm}^{-1}\text{d})$. Calculating r_1 again, we get $(\text{d}) \subseteq r_\text{d} \subseteq (\text{pp}^{-1}\text{m}^{-1}\text{d})$. Since $(\text{pp}^{-1}) \not\subseteq \mathcal{S}$, and since we have $(\text{m}^{-1}\text{d}) \subseteq r_\text{d} \circ (\text{s})$ only if $(\text{p}^{-1}) \subseteq r_\text{d}$, we may assume that r_d is either (pd) , or (p^{-1}d) , or $(\text{p}^{-1}\text{m}^{-1}\text{d})$. The first case is impossible because of $(\text{pd}) \cap ((\text{p}^{-1}\text{d}^{-1}) \circ (\text{s})) = (\text{d})$. Let $(\text{p}^{-1}\text{d}) \subseteq r_\text{d} \subseteq (\text{p}^{-1}\text{m}^{-1}\text{d})$. Then $(\text{p}^{-1}\text{m}^{-1}\text{o}^{-1}\text{df}) = r_\text{d} \circ (\text{s}) \in \mathcal{S}$. Suppose \mathcal{S} contains a non-empty subrelation r_2 of $(\text{p}^{-1}\text{m}^{-1}\text{o}^{-1}\text{f})$. Then $r_3 = r_\text{d} \cap (r_2 \circ (\text{s}^{-1}))$ is a non-empty subrelation of $(\text{p}^{-1}\text{m}^{-1})$ implying that $(\text{p}^{-1}) = r_3 \circ r_3 \in \mathcal{S}$, a contradiction. Therefore, for every $r \in \mathcal{S}$, we have $r \cap (\text{p}^{-1}\text{m}^{-1}\text{o}^{-1}\text{f}) \neq \emptyset \Rightarrow (\text{d}) \subseteq r$, which means that $\mathcal{S} \subseteq \mathcal{S}_\text{d}$.

Case 2. $(\text{dd}^{-1}) \subseteq r_\text{d} \cap (\text{oo}^{-1}\text{dd}^{-1})$.

Suppose a relation $r_4 \in \mathcal{S}$ satisfies both $r_4 \not\subseteq (\equiv \text{ss}^{-1})$ and $(\text{dd}^{-1}) \not\subseteq r_4$. Then it is not hard to verify that either $((\text{s}^{-1}) \circ r_4) \cap r_\text{d}$ or $(r_4 \circ (\text{s})) \cap r_\text{d}$ contains exactly one of (d) and (d^{-1}) which contradicts the minimality of r_d . Thus, for every $r \in \mathcal{S}$ such that $r \not\subseteq (\equiv \text{ss}^{-1})$, we have $(\text{dd}^{-1}) \subseteq r$. This implies that $\mathcal{S} \subseteq \mathcal{S}_\text{d}$. \square

Lemma 5 Suppose $r_\text{o} \cap (\text{ss}^{-1}) = \emptyset$. If $r_\text{o} \cap (\text{oo}^{-1}\text{dd}^{-1}) = (\text{o})$ or $(\text{oo}^{-1}) \subseteq r_\text{o}$ then $\mathcal{S} \subseteq \mathcal{S}_\text{o}$.

Proof. Similar to the previous lemma. \square

Lemma 6 If $r_\text{d} = r_{\text{o}^{-1}}$ and $r_\text{d} \cap (\text{oo}^{-1}\text{dd}^{-1}\text{ss}^{-1}) = (\text{o}^{-1}\text{d})$ then $\mathcal{S} \subseteq \mathcal{S}_\text{d}$.

Proof. As in the proof of Lemma 4, we can obtain $(\text{o}^{-1}\text{d}) \subseteq r_\text{d} \subseteq (\text{pp}^{-1}\text{m}^{-1}\text{o}^{-1}\text{df})$. If $(\text{po}^{-1}\text{d}) \subseteq r_\text{d}$ then $(r_\text{d} \circ (\text{s}^{-1}))^* = (\text{pp}^{-1}) \in \mathcal{S}$, a contradiction. Therefore we have $(\text{o}^{-1}\text{d}) \subseteq r_\text{d} \subseteq (\text{p}^{-1}\text{m}^{-1}\text{o}^{-1}\text{df})$, and $(\text{p}^{-1}\text{m}^{-1}\text{o}^{-1}\text{df}) = r_\text{d} \circ (\text{s}^{-1}) \in \mathcal{S}$. If some non-empty subrelation r_1 of $(\text{p}^{-1}\text{m}^{-1}\text{f})$ belongs to \mathcal{S} then $r_2 = r_1 \cap (r_1 \circ (\text{s}^{-1}))$ is a non-empty subrelation of $(\text{p}^{-1}\text{m}^{-1})$. Then $r_2 \circ r_2 = (\text{p}^{-1})$ which contradicts our assumptions. Therefore, for every $r \in \mathcal{S}$, we have $r \cap (\text{p}^{-1}\text{m}^{-1}\text{o}^{-1}\text{f}) \neq \emptyset \Rightarrow (\text{d}) \subseteq r$, which means that $\mathcal{S} \subseteq \mathcal{S}_\text{d}$. \square

Lemma 7 If $r_\text{d} = r_\text{o}$ and $r_\text{d} \cap (\text{oo}^{-1}\text{dd}^{-1}\text{ss}^{-1}) = (\text{od})$ then $\mathcal{S} \subseteq \mathcal{H}$ or \mathcal{S} is NP-complete.

Proof. As in the previous lemmas, it can be shown that $(\text{od}) \subseteq r_\text{d} \subseteq (\text{pp}^{-1}\text{mm}^{-1}\text{od})$. It follows that $r_\text{d}^* = \emptyset$ and $(\text{pmod}) = (\text{s}) \circ r_\text{d} \in \mathcal{S}$. Furthermore, $(\text{oo}^{-1}\text{dd}^{-1}\text{ff}^{-1}) = ((\text{s}^{-1}) \circ (\text{pmod}))^* \in \mathcal{S}$ and $(\text{od}) = (\text{pmod}) \cap (\text{oo}^{-1}\text{dd}^{-1}\text{ff}^{-1}) \in \mathcal{S}$. By Lemma 2, we know that every $r \in \mathcal{S}$ satisfies condition 3) of \mathcal{H} . Suppose some $r_1 \in \mathcal{S}$ does not satisfy condition 1) of \mathcal{H} . Then r_1 can be chosen so that $(\text{s}^{-1}\text{f}^{-1}) \subseteq r_1 \subseteq (\equiv \text{pmodss}^{-1}\text{ff}^{-1})$ or $(\text{os}^{-1}) \subseteq r_1 \subseteq (\equiv \text{pmodss}^{-1}\text{ff}^{-1})$. If $(\text{f}) \subseteq r_1$ then $(\text{ff}^{-1}) = (r_1 \cap (\text{oo}^{-1}\text{dd}^{-1}\text{ff}^{-1}))^* \in \mathcal{S}$ which contradicts our assumptions. Further, if $(\text{od}) \not\subseteq r_1$ then $r_1 \cap (\text{oo}^{-1}\text{dd}^{-1}\text{ff}^{-1}) = (\text{f}^{-1}) \in \mathcal{S}$, a contradiction. Therefore we may assume that $(\text{ods}^{-1}\text{f}^{-1}) \subseteq r_1 \subseteq (\equiv \text{pmodss}^{-1}\text{f}^{-1})$. Now it can be checked that $r_2 = r_1 \cap (r_1^{-1} \circ (\text{o}^{-1}\text{d}^{-1}))$ satisfies $(\text{ods}^{-1}\text{f}^{-1}) \subseteq r_2 \subseteq (\text{pmodss}^{-1}\text{f}^{-1})$. Then $\{r_2\}$ is NP-complete by Proposition 2(2). One can proceed similarly if condition 2) of \mathcal{H} fails in \mathcal{S} . \square

It remains to consider the case when each non-trivial $r \in \mathcal{S}$ satisfies $r \cap (\text{ss}^{-1}) \neq \emptyset$.

Lemma 8 If $r \cap (\text{ss}^{-1}) \neq \emptyset$ for any non-trivial $r \in \mathcal{S}$ then $\mathcal{S} \subseteq \mathcal{A}_i$ for some $1 \leq i \leq 4$.

Proof. **Case 1.** $r_\text{p} \cap (\text{ss}^{-1}) = (\text{s}^{-1})$.

We have $(\text{ps}^{-1}) \subseteq r_\text{p}$ and $(\text{s}) \not\subseteq r_\text{p}$. Let $r_1 = (\text{s}^{-1}) \circ r_\text{p} \in \mathcal{S}$. Then it is easy to check that $(\text{pmod}^{-1}\text{s}^{-1}\text{f}^{-1}) \subseteq r_1$, and that $r_1 \cap (\equiv \text{s}) = \emptyset$. It follows that $r_1 = (\text{pmod}^{-1}\text{s}^{-1}\text{f}^{-1})$, since otherwise r_1^* is non-empty and $r_1^* \cap (\text{ss}^{-1}) = \emptyset$. No non-empty subrelation of $(\text{pmod}^{-1}\text{f}^{-1})$ can belong to \mathcal{S} . Therefore, for any $r \in \mathcal{S}$, $r \cap (\text{pmod}^{-1}\text{f}^{-1}) \neq \emptyset$ implies $(\text{s}^{-1}) \subseteq r$. Hence $\mathcal{S} \subseteq \mathcal{A}_1$.

Case 2. $r_\text{d} \cap (\text{ss}^{-1}) = (\text{s}^{-1})$.

The proof is similar to Case 1; the only change is that $r_1 = r_\text{d} \circ (\text{s}^{-1})$, and we deduce that $r_1 = (\text{p}^{-1}\text{m}^{-1}\text{o}^{-1}\text{ds}^{-1}\text{f})$, and, hence, $\mathcal{S} \subseteq \mathcal{A}_2$.

Case 3. $r_\text{o} \cap (\text{ss}^{-1}) = (\text{s}^{-1})$.

In view of Cases 1 and 2 we may assume that $(\text{os}^{-1}) \subseteq r_\text{o} \subseteq (\equiv \text{p}^{-1}\text{mm}^{-1}\text{od}^{-1}\text{s}^{-1}\text{ff}^{-1})$. Let $r_1 = (\text{s}^{-1}) \circ r_\text{o}$. It is easy to check that

$$(\text{od}^{-1}\text{s}^{-1}\text{f}^{-1}) \subseteq r_1 \subseteq (\text{p}^{-1}\text{m}^{-1}\text{oo}^{-1}\text{d}^{-1}\text{s}^{-1}\text{f}^{-1}).$$

Since $r_1^* \neq (oo^{-1})$, we obtain $r_1 \subseteq (p^{-1}m^{-1}od^{-1}s^{-1}f^{-1})$. It can straightforwardly be verified that if $r_1 \neq (od^{-1}s^{-1}f^{-1})$, then $r_2 = (r_1 \circ r_1)^* \in \mathcal{S}$, but $r_2 \cap (pp^{-1}ss^{-1}) = (pp^{-1})$, which contradicts the assumption made. Therefore, $r_1 = (od^{-1}s^{-1}f^{-1})$, and $(pmod^{-1}s^{-1}f^{-1}) = r_1 \circ r_1 \in \mathcal{S}$. Hence, for any $r \in \mathcal{S}$, $r \cap (pmod^{-1}f^{-1}) \neq \emptyset$ implies $(s^{-1}) \subseteq r$, and we have $\mathcal{S} \subseteq \mathcal{A}_1$.

Case 4. $r_m \cap (ss^{-1}) = (s^{-1})$.

Similarly to Case 3, we infer that $\mathcal{S} \subseteq \mathcal{A}_1$.

Case 5. (s) is contained in each of r_p, r_d, r_o , and r_m .

We have $r_f \cap (ss^{-1}) \neq \emptyset$. Then it follows that if $(s) \subseteq r_f$ then $\mathcal{S} \subseteq \mathcal{A}_3$. Otherwise, $(s^{-1}) \subseteq r_f$ and we have $\mathcal{S} \subseteq \mathcal{A}_4$. \square

3.3 The d/o case

Suppose $bas(\mathcal{S}) = \{\equiv, d, d^{-1}\}$. We give a sketch of a proof that either \mathcal{S} is contained in \mathcal{S}_d or in \mathcal{E}_d (and therefore is tractable), or else it contains (oo^{-1}) or (pp^{-1}) (and then it is NP-complete by Proposition 2(1)).

Let $sym(\mathcal{S}) = \{r^* | r \in \mathcal{S}\}$. First it can be shown that either $sym(\mathcal{S})$ is contained in \mathcal{S}_d or in \mathcal{E}_d , or else \mathcal{S} contains (oo^{-1}) or (pp^{-1}) . Then one can check, similarly to Lemma 3, that if there exists a non-trivial $r \in \mathcal{S}$ such that $r \cap (dd^{-1}) = \emptyset$ then r satisfies $r \subseteq (\equiv ss^{-1})$ or $r \subseteq (\equiv ff^{-1})$, and in this case, once again, \mathcal{S} is contained in \mathcal{S}_d or in \mathcal{E}_d , respectively, or else \mathcal{S} contains (oo^{-1}) or (pp^{-1}) . Finally, we assume that each non-trivial $r \in \mathcal{S}$ satisfies $r \cap (dd^{-1}) \neq \emptyset$, and then the required result can be proved by analysing the minimal relations r_p, r_m, r_o, r_s , and r_f , as in the proof of Lemma 8.

The case $bas(\mathcal{S}) = \{\equiv, o, o^{-1}\}$ is very similar to the previous one. The only essential difference is that the relation (pp^{-1}) cannot belong to \mathcal{S} ; otherwise we get $(o) \circ (o) = (pmo)$ and $(p) = (pp^{-1}) \cap (pmo) \in \mathcal{S}$, a contradiction.

4 Conclusion

We have proved that any so far unknown tractable fragment of Allen's algebra can contain no other basic relation than (\equiv) . We believe that, by extending the algebraic techniques used to establish our result, it will be possible to show that there are in fact no other forms of tractability in Allen's algebra. In other words, we conjecture that all maximal tractable subalgebras are already identified. However, the proof of this conjecture requires a more elaborate algebraic argument than the proofs in this paper, since in the general case we have no particular relation to start with. Algebraic techniques similar to the one used in this paper can also be applied to the study of the complexity in other temporal and spatial formalisms.

References

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Angelsmark and Jonsson, 2000] Ola Angelsmark and Peter Jonsson. Some observations on durations, scheduling and Allen's algebra. In *Proceedings of the 6th Conference on Constraint Programming (CP'00)*, volume 1894 of *Lecture Notes in Computer Science*, pages 484–488. Springer-Verlag, 2000.

[Bellini et al., 2000] P. Bellini, R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys*, 32(1):12–42, 2000.

[Drakengren and Jonsson, 1997a] Thomas Drakengren and Peter Jonsson. Eight maximal tractable subclasses of Allen's algebra with metric time. *Journal of Artificial Intelligence Research*, 7:25–45, 1997.

[Drakengren and Jonsson, 1997b] Thomas Drakengren and Peter Jonsson. Twenty-one large tractable subclasses of Allen's algebra. *Artificial Intelligence*, 93(1-2):297–319, 1997.

[Drakengren and Jonsson, 1998] Thomas Drakengren and Peter Jonsson. A complete classification of tractability in Allen's algebra relative to subsets of basic relations. *Artificial Intelligence*, 106(2):205–219, 1998.

[Garey and Johnson, 1979] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[Golumbic and Shamir, 1993] Martin C. Golumbic and Ron Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM*, 40(5):1108–1133, 1993.

[Hirsch, 1996] Robin Hirsch. Relation algebras of intervals. *Artificial Intelligence*, 83(2):1–29, 1996.

[Ladkin and Maddux, 1994] Peter B. Ladkin and Roger D. Maddux. On binary constraint problems. *Journal of the ACM*, 41(3):435–469, 1994.

[Ladkin and Reinefeld, 1992] Peter B. Ladkin and Alexander Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1):105–124, 1992.

[Ligozat, 1998] Gérard Ligozat. "Corner" relations in Allen's algebra. *Constraints*, 3(2-3):165–177, 1998.

[Meiri, 1996] Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence*, 87(1-2):343–385, 1996.

[Nebel and Bürckert, 1995] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.

[Nökel, 1991] Klaus Nökel. *Temporally Distributed Symptoms in Technical Diagnosis*, volume 517 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1991.

[Szendrei, 1995] Ágnes Szendrei. Maximal non-affine reducts of simple affine algebras. *Algebra Universalis*, 34(1):144–174, 1995.

[Vilain et al., 1989] Marc B. Vilain, Henry A. Kautz, and Peter G. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In Daniel S. Weld and Johan de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, San Mateo, CA, 1989.

Interval-based Temporal Reasoning with General TBoxes

Carsten Lutz

LuFG Theoretical Computer Science
RWTH Aachen, Germany
lutz@cs.rwth-aachen.de

Abstract

Until now, interval-based temporal Description Logics (DLs) did—if at all—only admit TBoxes of a very restricted form, namely acyclic macro definitions. In this paper, we present a temporal DL that overcomes this deficiency and combines interval-based temporal reasoning with general TBoxes. We argue that this combination is very interesting for many application domains. An automata-based decision procedure is devised and a tight EXPTIME-complexity bound is obtained. Since the presented logic can be viewed as being equipped with a concrete domain, our results can be seen from a different perspective: we show that there exist interesting concrete domains for which reasoning with general TBoxes is decidable.

1 Motivation

Description Logics (DLs) are a family of formalisms well-suited for the representation of and reasoning about conceptual knowledge. Whereas most Description Logics represent only static aspects of the application domain, recent research resulted in the exploration of various Description Logics that allow to, additionally, represent temporal information, see, e.g., [Artale and Franconi,2000] for an overview. One approach for temporal reasoning with DLs is to use so-called concrete domains. Concrete domains have been proposed as an extension of Description Logics that allows reasoning about “concrete qualities” of entities of the application domain such as sizes, weights or temperatures [Baader and Hanschke,1991]. As was first described in [Lutz *et al.*,1997], if a “temporal” concrete domain is employed, then Description Logics with concrete domains are a very useful tool for temporal reasoning. Ontologically, temporal reasoning with concrete domains is usually interval-based but may also be point-based or even both.

In this paper, we define a temporal Description Logic based on concrete domains which uses points as its basic temporal entity, but which may also be used as a full-fledged interval-based temporal DL. More precisely, the presented logic \mathcal{TDL} extends the basic Description Logic \mathcal{ALC} with a concrete domain that is based on the rationals and predicates $<$ and $=$. The well-known Allen relations can be defined in terms of

their endpoints [Allen,1983] thus allowing for (qualitative) interval-based temporal reasoning. Since it is an important feature of DLs that reasoning should be decidable, we prove decidability of the standard reasoning tasks by using an automata-theoretic approach which also yields a tight EXPTIME complexity bound.

Most DLs allow for some kind of TBox formalism that is used to represent terminological knowledge as well as background knowledge about the application domain. However, there exist various flavours of TBoxes with vast differences in expressivity. To the best of our knowledge, all interval-based DLs and all DLs with concrete domains defined in the literature admit only a very restricted form of TBox, i.e., sets of acyclic macro definitions. Compared to existing Description Logics that are interval-based or include concrete domains, the distinguishing feature of our logic is that it is equipped with a very general form of TBoxes that allows arbitrary equations over concepts. Thus, the presented work overcomes a major limitation of both families of Description Logics.

Our results can be viewed from the perspective of interval-based temporal reasoning and from the perspective of concrete domains. For the temporal perspective, we claim that the combination of general TBoxes and interval-based temporal reasoning is important for many application areas. In this paper, we present process engineering as an example. From the concrete domain perspective, our results can be viewed as follows: in [Lutz,2001], it is shown that, even for very simple concrete domains, reasoning with general TBoxes is undecidable. It was an open question whether there exist interesting concrete domains for which reasoning with general TBoxes is decidable. In this paper, we answer this question to the affirmative. This paper is accompanied by a technical report containing full proofs [Lutz,2000].

2 Syntax and Semantics

In this section, we introduce syntax and semantics of the Description Logic \mathcal{TDL} .

Definition 1. Let N_C , N_R , and N_{cF} be mutually disjoint and countably infinite sets of *concept names*, *roles*, and *concrete features*. Furthermore, let N_{aF} be a countably infinite subset of N_R . The elements of N_{aF} are called *abstract features*. A *path* u is a composition $f_1 \cdots f_n g$ of $n \geq 0$ abstract fea-

tures f_1, \dots, f_n and one concrete feature g . The set of \mathcal{TDL} -concepts is the smallest set such that

1. every concept name is a concept
2. if C and D are concepts, R is a role, g is a concrete feature, u_1, u_2 are paths, and $P \in \{<, =\}$, then the following expressions are also concepts: $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\exists u_1, u_2.P$, and $g\uparrow$.

A *TBox axiom* is an expression of the form $C \sqsubseteq D$ with C and D are concepts. A finite set of TBox axioms is a *TBox*.

Throughout this paper, we will denote atomic concepts by the letter A , (possibly complex) concepts by the letters C, D, E , roles by the letter R , abstract features by the letter f , concrete features by the letter g , paths by the letter u , and elements of the set $\{<, =\}$ by the letter P . We will sometimes call the TBox formalism introduced above *general TBoxes* to distinguish it from other, weaker formalisms such as the ones in [Nebel,1990]. As most Description Logics, \mathcal{TDL} is equipped with a Tarski-style semantics.

Definition 2. An *interpretation* \mathcal{I} is a pair $(\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta_{\mathcal{I}}$ is a set called the *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function* mapping each concept name C to a subset $C^{\mathcal{I}}$ of $\Delta_{\mathcal{I}}$, each role name R to a subset $R^{\mathcal{I}}$ of $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$, each abstract feature f to a partial function $f^{\mathcal{I}}$ from $\Delta_{\mathcal{I}}$ to $\Delta_{\mathcal{I}}$, and each concrete feature g to a partial function $g^{\mathcal{I}}$ from $\Delta_{\mathcal{I}}$ to the rationals \mathbb{Q} . For paths $u = f_1 \cdots f_n g$, we set $u^{\mathcal{I}}(a) := g^{\mathcal{I}}(f_n^{\mathcal{I}}(\cdots(f_1^{\mathcal{I}}(a))\cdots))$. The interpretation function is extended to arbitrary concepts as follows:

$$\begin{aligned} (\neg C)^{\mathcal{I}} &:= \Delta_{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}} \quad (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}} \\ (\exists R.C)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \{b \mid (a, b) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}} \neq \emptyset\} \\ (\forall R.C)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \{b \mid (a, b) \in R^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}\} \\ (\exists u_1, u_2.P)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \exists q_1, q_2 \in \mathbb{Q} : u_1^{\mathcal{I}}(a) = q_1, \\ &\quad u_2^{\mathcal{I}}(a) = q_2, q_1 P q_2\} \\ (g\uparrow)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid g^{\mathcal{I}}(a) \text{ is undefined}\} \end{aligned}$$

An interpretation \mathcal{I} is a *model* of a TBox \mathcal{T} iff it satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all axioms $C \sqsubseteq D$ in \mathcal{T} . \mathcal{I} is a *model* of a concept C iff $C^{\mathcal{I}} \neq \emptyset$.

If $g(a) = x$ for some $g \in N_{cF}$, $a \in \Delta_{\mathcal{I}}$, and $x \in \mathbb{Q}$, then we call x a *concrete successor* of a in \mathcal{I} . We write \top for $A \sqcup \neg A$ and \perp for $\neg \top$, where A is a concept name. Moreover, we write $u\uparrow$ with $u = f_1 \cdots f_k g$ for $\forall f_1. \cdots \forall f_k. g\uparrow$.

Definition 3 (Inference Problems). Let C and D be concepts and \mathcal{T} be a TBox. C *subsumes* D w.r.t. \mathcal{T} (written $D \sqsubseteq_{\mathcal{T}} C$) iff $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ for all models \mathcal{I} of \mathcal{T} . C is *satisfiable* w.r.t. \mathcal{T} iff there exists a model of both \mathcal{T} and C .

It is well-known that (un)satisfiability and subsumption can be mutually reduced to each other: $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. \mathcal{T} and C is satisfiable w.r.t. \mathcal{T} iff $C \not\sqsubseteq_{\mathcal{T}} \perp$.

We now discuss the relationship between \mathcal{TDL} and Description Logics with concrete domains.

Definition 4 (Concrete Domain). A *concrete domain* \mathcal{D} is a pair $(\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$, where $\Delta_{\mathcal{D}}$ is a set called the domain, and $\Phi_{\mathcal{D}}$

is a set of predicate names. Each predicate name $P \in \Phi_{\mathcal{D}}$ is associated with an arity n and an n -ary predicate $P^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^n$.

The concrete domain is usually integrated into the logic by a concept constructor $\exists u_1, \dots, u_n.P$, with semantics

$$\begin{aligned} (\exists u_1, \dots, u_n.P)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid u_i^{\mathcal{I}}(a) = q_i \text{ for } 1 \leq i \leq n \\ &\quad \text{and } (q_1, \dots, q_n) \in P^{\mathcal{D}}\}. \end{aligned}$$

It is obvious that \mathcal{TDL} can be viewed as being equipped with the concrete domain $\mathcal{D}_{<} := (\mathbb{Q}, \{<, =\})$, where $<$ and $=$ are binary predicates with the usual semantics. For most DLs with concrete domains, it is required that the set of predicates is closed under negation and contains a name $\top_{\mathcal{D}}$ for $\Delta_{\mathcal{D}}$. This property ensures that every concept can be converted into an equivalent one in the so-called negation normal form (NNF). The NNF of concepts, in turn, is used as a starting point for devising satisfiability algorithms. It is not hard to see that $\mathcal{D}_{<}$ is not admissible in this sense. However, as we will see in Section 4, the conversion of \mathcal{TDL} -concepts into equivalent ones in NNF is nevertheless possible.

3 Temporal Reasoning with \mathcal{TDL}

Although \mathcal{TDL} does only provide the relations “=” and “<” on time points, it is not hard to see that the remaining relations can be defined by writing, e.g., $\exists u_2, u_1.< \sqcup \exists u_1, u_2.=$ for $\exists u_1, u_2.\geq$. However, we claim that \mathcal{TDL} cannot only be used for point-based temporal reasoning but also as a full-fledged interval-based temporal Description Logic. As observed by Allen [1983], there are 13 possible relationships between two intervals such as, for example, the meets relation: two intervals i_1 and i_2 are related by meets iff the right endpoint of i_1 is identical to the left endpoint of i_2 —see [Allen,1983] for an exact definition of the other relations. As we shall see, these 13 Allen relations (as well as the additional relations from the Allen algebra, see [Allen,1983]) can be defined in \mathcal{TDL} . In the following, we present a framework for mixed interval- and point-based reasoning in \mathcal{TDL} and apply this framework in the application area of process engineering

The representation framework consists of several conventions and abbreviations. We assume that each entity of the application domain is either temporal or atemporal. If it is temporal, its temporal extension may be either a time point or an interval. Left endpoints of intervals are represented by the concrete feature ℓ , right endpoints of intervals are represented by the concrete feature r , and time-points not related to intervals are represented by the concrete feature t . All this can be expressed by the following TBox \mathcal{T}^* :

$$\begin{aligned} \text{Point} &\doteq \exists t, t.= \sqcap \ell\uparrow \sqcap r\uparrow \quad \text{Interval} \doteq \exists \ell, r.< \sqcap t\uparrow \\ \text{Atemporal} &\doteq t\uparrow \sqcap \ell\uparrow \sqcap r\uparrow \quad \text{Interval} \supseteq \exists \ell, \ell, = \sqcup \exists r, r.= \end{aligned}$$

Here, $C \doteq D$ is an abbreviation for $\{C \sqsubseteq D, D \sqsubseteq C\}$. Let us now define the Allen relations as abbreviations. For example, $\exists(F, F').\text{contains}$ is an abbreviation for $\exists F\ell, F'\ell.< \sqcap \exists F'r, F'r.<$ where $F, F' \in (N_{aF})^*$, i.e., F and F' are words over the alphabet N_{aF} . Note that

$$\exists(F, F').\text{contains} \sqsubseteq_{\mathcal{T}^*} \exists F.\text{Interval} \sqcap \exists F'.\text{Interval}.$$

Similar abbreviations are introduced for the other Allen relations. We use self to denote the empty word. For example,

$\exists(F, \text{self}).\text{starts}$ is an abbreviation for $\exists F\ell, \ell. = \sqcap \exists Fr, r. <$. Intuitively, self refers to the interval associated with the abstract object at which the $\exists(F, \text{self}).\text{starts}$ concept is “evaluated”.

Since we have intervals *and* points available, we should also be able to talk about the relationship of points and intervals. More precisely, there exist 5 possible relations between a point and an interval [Vilain,1982], one example being startsp that holds between a point p and an interval i if p is identical to the left endpoint of i . Hence, we can define $\exists(Ft, F').\text{startsp}$ as an abbreviation for $\exists Ft, F'\ell. =$ and similar abbreviations for beforep , duringp , finishesp , and afterp .

This finishes the definition of the framework. We claim that the combination of interval-based reasoning and general TBoxes is important for many application areas such as reasoning about action and plans [Artale and Franconi,2000]. The examples presented here are from the area of process engineering that was first considered by Sattler in a DL context [Sattler,1998]. However, Sattler’s approach does not take into account temporal aspects of the application domain. We show how this can be done using \mathcal{TDL} thus refining Sattler’s proposal.

Assume that our goal is to represent information about an automated chemical production process that is carried out by some complex technical device. The device operates each day for some time depending on the number of orders. It needs a complex startup and shutdown process before resp. after operation. Moreover, some weekly maintenance is needed to keep the device functional. Let us first represent the underlying temporal structure that consists of weeks and days.

$$\begin{aligned} \text{Week} \doteq & \text{Interval} \sqcap \prod_{1 \leq i \leq 7} \exists \text{day}_i. \text{Day} \sqcap \\ & \exists(\text{day}_1, \text{self}).\text{starts} \sqcap \exists(\text{day}_7, \text{self}).\text{finishes} \sqcap \\ & \prod_{1 \leq i < 7} \exists(\text{day}_i, \text{day}_{i+1}).\text{meets} \sqcap \\ & \exists \text{next}. \text{Week} \sqcap \exists(\text{self}, \text{next}).\text{meets} \end{aligned}$$

The axiom states that each week consists of seven days, where the i ’th day is accessible from the corresponding week via the abstract feature day_i . The temporal relationship between the days are as expected: Monday starts the week, Sunday finishes it, and each day temporally meets the succeeding one. Note that this implies that days 2 to 6 are during the corresponding week although this is not explicitly stated. Moreover, each week has a successor week that it temporally meets. We now describe the startup, operation, shutdown, and maintenance phases.

$$\begin{aligned} \text{Day} \doteq & \text{Interval} \sqcap \\ & \exists \text{start}. \text{Startup} \sqcap \exists \text{op}. \text{Operation} \sqcap \exists \text{shut}. \text{Shutdn} \sqcap \\ & \exists \text{start} \circ \ell, \ell. \geq \sqcap \exists(\text{start}, \text{op}).\text{meets} \sqcap \\ & \exists(\text{op}, \text{shut}).\text{meets} \sqcap \exists \text{shut} \circ r, r. \leq \end{aligned}$$

$$\text{Week} \sqsubseteq \exists \text{maint}. \text{Maintenance} \sqcap \exists(\text{self}, \text{maint}).\text{contains}$$

Here start , op , shut , and maint are abstract features and “ \circ ” is used for better readability (i.e., paths $f_1 \cdots f_k g$ are written as $f_1 \circ \cdots \circ f_k \circ g$). The TBox implies that phases are related to the corresponding day as follows: startup via starts or during , shutdown via during or finishes , and operation via

during . Until now, we did not say anything about the temporal relationship of maintenance and operation. This may be inadequate, if, for example, maintenance and operation are mutually exclusive. We can take this into account by using additional axioms

$$\text{Week} \sqcap \bigsqcup_{1 \leq i \leq 7} \exists(\text{maint}, \text{day}_i \circ \text{op}). \text{OVLP} \sqsubseteq \perp \quad (*)$$

where OVLP is replaced by equal, overlaps, overlapped-by, during, contains, starts, started-by, finishes, or finished-by yielding 9 axioms.

Until now, we have modelled the very basic properties of our production process. Let us define some more advanced concepts to illustrate reasoning with \mathcal{TDL} . For example, we could define a busy week as

$$\text{BusyWeek} \doteq \text{Week} \sqcap \prod_{1 \leq i \leq 7} (\exists \text{day}_i \circ \text{start}, \text{day}_i.\text{starts} \sqcap \exists \text{day}_i \circ \text{shut}, \text{day}_i.\text{finishes})$$

i.e., each day, the startup process starts at the beginning of the day and the shutdown finishes at the end of the day. Say now that it is risky to do maintenance during startup and shutdown phases and define

$$\text{RiskyWeek} \doteq \text{Week} \sqcap \neg \prod_{1 \leq i \leq 7} (\exists \text{maint}, \text{day}_i \circ \text{start}. \text{before} \sqcup \exists \text{maint}, \text{day}_i \circ \text{shut}. \text{after})$$

expressing that, in a risky week, the maintenance phase is not strictly separated from the startup and shutdown phases. A \mathcal{TDL} reasoner could be used to detect that $\text{BusyWeek} \sqsubseteq \text{RiskyWeek}$, i.e., every busy week is a risky week: in a busy week, the week is partitioned into startup, shutdown, and operation phases. Since maintenance may not OVLP with operation phases (see (*)), it must OVLP with startup and/or shutdown phases which means that it is a risky week. We can further refine this model by using mixed point-based and interval-based reasoning, see [Lutz,2000] for examples.

4 The Decision Procedure

In this section, we prove satisfiability of \mathcal{TDL} -concepts w.r.t. TBoxes to be decidable and obtain a tight EXPTIME complexity bound. This is done using an automata-theoretic approach: first, we abstract models to so-called Hintikka-trees such that there exists a model for a concept C and a TBox \mathcal{T} iff there exists a Hintikka-tree for C and \mathcal{T} . Then, we build, for each \mathcal{TDL} -concept C and TBox \mathcal{T} , a looping automaton $\mathcal{A}_{(C, \mathcal{T})}$ that accepts exactly the Hintikka-trees for (C, \mathcal{T}) . In particular, this implies that $\mathcal{A}_{(C, \mathcal{T})}$ accepts the empty language iff C is unsatisfiable w.r.t. \mathcal{T} .

Definition 5. Let M be a set and $k \geq 1$. A k -ary M -tree is a mapping $T : \{1, \dots, k\}^* \rightarrow M$ that labels each node $\alpha \in \{1, \dots, k\}^*$ with $T(\alpha) \in M$. Intuitively, the node αi is the i -th child of α . We use ϵ to denote the empty word (corresponding to the root of the tree).

A *looping automaton* $\mathcal{A} = (Q, M, I, \Delta)$ for k -ary M -trees is defined by a set Q of states, an alphabet M , a subset $I \subseteq Q$ of initial states, and a transition relation $\Delta \subseteq Q \times M \times Q^k$. A *run* of \mathcal{A} on an M -tree T is a mapping $r : \{1, \dots, k\}^* \rightarrow Q$ with $r(\epsilon) \in I$ and $(r(\alpha), T(\alpha), r(\alpha 1), \dots, r(\alpha k)) \in \Delta$ for each $\alpha \in$

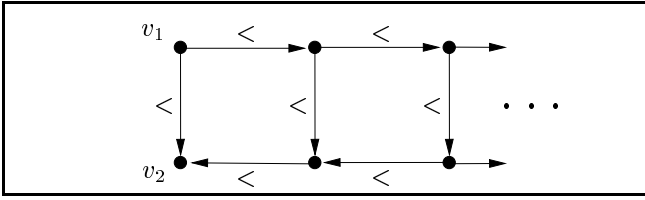


Figure 1: A constraint graph containing no $<$ -cycle that is unsatisfiable over \mathbb{N} .

$\{1, \dots, k\}^*$. A looping automaton accepts all those M -trees for which a run exists, i.e., the language $\mathcal{L}(\mathcal{A})$ of M -trees accepted by \mathcal{A} is

$$L(\mathcal{A}) = \{T \mid \text{there is a run of } \mathcal{A} \text{ on } T\}.$$

In [Vardi and Wolper,1986], it is proved that the emptiness problem for looping automata is decidable in polynomial time.

A Hintikka-tree for C and \mathcal{T} corresponds to a canonical model for C and \mathcal{T} . Apart from describing the abstract domain $\Delta_{\mathcal{T}}$ of the corresponding canonical model \mathcal{I} together with the interpretation of concepts and roles, each Hintikka-tree induces a directed graph whose edges are labelled with predicates from $\{<, =\}$. These constraint graphs describe the “concrete part” of \mathcal{I} (i.e., concrete successors of domain objects and their relationships).

Definition 6. A *constraint graph* is a pair $G = (V, E)$, where V is a countable set of *nodes* and $E \subseteq V \times V \times \{=, <\}$ a set of *edges*. We generally assume that constraint graphs are *equality closed*, i.e., that $(v_1, v_2, =) \in E$ implies $(v_2, v_1, =) \in E$. A constraint graph $G = (V, E)$ is called *satisfiable over M* , where M is a set equipped with a total ordering $<$, iff there exists a total mapping δ from V to M such that $\delta(v_1) P \delta(v_2)$ for all $(v_1, v_2, P) \in E$. In this case, δ is called a *solution* for G .

A *$<$ -cycle* Q in G is a finite non-empty sequence of nodes $v_0, \dots, v_{k-1} \in V$ such that (1) for all i with $i < k$, we have $(v_i, v_{i \oplus_k 1}, P) \in E$, where $P \in \{<, =\}$ and \oplus_k denotes addition modulo k and (2) $(v_i, v_{i \oplus_k 1}, <) \in E$ for some $i < k$.

The following theorem will be crucial for proving that, for every Hintikka-tree, there exists a corresponding canonical model. More precisely, it will be used to ensure that the constraint graph induced by a Hintikka-tree, which describes the concrete part of the corresponding model, is satisfiable.

Theorem 7. A constraint graph G is satisfiable over M with $M \in \{\mathbb{Q}, \mathbb{R}\}$ iff G does not contain a $<$ -cycle.

Note that Theorem 7 does *not* hold if satisfiability over \mathbb{N} is considered due to the absence of density: if there exist two nodes v_1 and v_2 such that the length of $<$ -paths (which are defined in the obvious way) between v_1 and v_2 is unbounded, a constraint graph is unsatisfiable over \mathbb{N} even if it contains no $<$ -cycle, see Figure 1.

The decidability procedure works on \mathcal{TDL} -concepts and TBoxes that are in a certain syntactic form.

Definition 8 (Path Normal Form). A \mathcal{TDL} -concept C is in *path normal form* (PNF) if, for all subconcepts $\exists u_1, u_2. P$ of

C , we have either (1) $u_1 = g_1$ and $u_2 = g_2$, (2) $u_1 = f g_1$ and $u_2 = g_2$, or (3) $u_1 = g_1$ and $u_2 = f g_2$ for some $f \in N_{aF}$ and $g_1, g_2 \in N_{cF}$. A \mathcal{TDL} TBox \mathcal{T} is in *path normal form* iff all concepts appearing in \mathcal{T} are in path normal form.

Lemma 9. Satisfiability of \mathcal{TDL} -concepts w.r.t. \mathcal{TDL} -TBoxes can be reduced to satisfiability of \mathcal{TDL} -concepts in PNF w.r.t. \mathcal{TDL} -TBoxes in PNF.

Proof Let C be a \mathcal{TDL} concept. For every path $u = f_1 \dots f_n g$ in C , we assume that $[g], [f_n g], \dots, [f_1 \dots f_n g]$ are concrete features. We inductively define a mapping λ from paths u in C to concepts as follows:

$$\lambda(g) = \top \quad \lambda(fu) = (\exists[fu], f[u]. =) \sqcap \exists f. \lambda(u)$$

For any \mathcal{TDL} -concept C , a corresponding concept $\rho(C)$ in PNF is obtained by replacing all subconcepts $\exists u_1, u_2. P$ of C with $\exists[u_1], [u_2]. P \sqcap \lambda(u_1) \sqcap \lambda(u_2)$ and $g \uparrow$ with $[g] \uparrow$. We extend the mapping ρ to TBoxes in the obvious way. It is easy to check that the translation is polynomial and that a concept C is satisfiable w.r.t. a TBox \mathcal{T} iff $\rho(C)$ is satisfiable w.r.t. $\rho(\mathcal{T})$. \square

Hence, it suffices to prove that satisfiability of concepts in PNF w.r.t. TBoxes in PNF is decidable. Moreover, we can assume all concepts and TBoxes to be in NNF.

Definition 10 (NNF). A concept C is in *negation normal form* (NNF) if negation occurs only in front of concept names. Every concept can be transformed into an equivalent one in NNF by eliminating double negation and using de Morgan’s law, the duality between \exists and \forall , and the following equivalences:

$$\begin{aligned} \neg(\exists u_1, u_2. P) &\equiv \exists u_1, u_2. \tilde{P} \sqcup \exists u_2, u_1. < \sqcup u_1 \uparrow \sqcup u_2 \uparrow \\ \neg(g \uparrow) &\equiv \exists g, g. = \end{aligned}$$

where $\tilde{\cdot}$ denotes the exchange of predicates, i.e., $\tilde{<} \text{ is } =$ and $\tilde{=} \text{ is } <$. With $\text{nfn}(C)$, we denote the equivalent of C in NNF. A TBox \mathcal{T} is in NNF iff all concepts in \mathcal{T} are in NNF.

Note that transformation to NNF preserves PNF. We often refer to TBoxes \mathcal{T} in their *concept form* $C_{\mathcal{T}}$:

$$C_{\mathcal{T}} = \bigcap_{C \sqsubseteq D \in \mathcal{T}} \text{nfn}(\neg C \sqcup D).$$

We now define Hintikka-trees for concepts C and TBoxes \mathcal{T} (in PNF and NNF) and show that there exists Hintikka-tree for C and \mathcal{T} iff there exists a model for C and \mathcal{T} .

Let C be a concept and \mathcal{T} a TBox. With $\text{cl}(C, \mathcal{T})$, we denote the set of subconcepts of C and $C_{\mathcal{T}}$. We assume that existential concepts $\exists R. D$ in $\text{cl}(C, \mathcal{T})$ with $R \in N_R \setminus N_{aF}$ are linearly ordered, and that $\mathcal{E}(C, \mathcal{T}, i)$ yields the i -th existential concept in $\text{cl}(C, \mathcal{T})$. Furthermore, we assume the abstract features used in $\text{cl}(C, \mathcal{T})$ to be linearly ordered and use $\mathcal{F}(C, \mathcal{T}, i)$ to denote the i -th abstract feature in $\text{cl}(C, \mathcal{T})$. The set of concrete features used in $\text{cl}(C, \mathcal{T})$ is denoted with $\mathcal{G}(C, \mathcal{T})$. Hintikka-pairs are used as labels of the nodes in Hintikka-trees.

Definition 11 (Hintikka-set, Hintikka-pair). Let C be a concept and \mathcal{T} be a TBox. A set $\Psi \subseteq \text{cl}(C, \mathcal{T})$ is a *Hintikka-set* for (C, \mathcal{T}) iff it satisfies the following conditions:

- (H1) $C_{\mathcal{T}} \in \Psi$,
- (H2) if $C_1 \sqcap C_2 \in \Psi$, then $\{C_1, C_2\} \subseteq \Psi$,
- (H3) if $C_1 \sqcup C_2 \in \Psi$, then $\{C_1, C_2\} \cap \Psi \neq \emptyset$,
- (H4) $\{A, \neg A\} \not\subseteq \Psi$ for all concept names $A \in \text{cl}(C, \mathcal{T})$,
- (H5) if $g \uparrow \in \Psi$, then $\exists u_1, u_2. P \notin \Psi$ for all concepts $\exists u_1, u_2. P$ with $u_1 = g$ or $u_2 = g$.

We say that $f \in N_{aF}$ is *enforced* by a Hintikka-set Ψ iff either $\exists f.C \in \Psi$ for some concept C or $\{\exists f.g_1, g_2.P, \exists g_1, f.g_2.P\} \cap \Psi \neq \emptyset$ for some $g_1, g_2 \in N_{cF}$ and $P \in \{<, =\}$. A *Hintikka-pair* (Ψ, χ) for (C, \mathcal{T}) consists of a Hintikka-set Ψ for (C, \mathcal{T}) and a set χ of tuples (g_1, g_2, P) with $g_1, g_2 \in \mathcal{G}(C, \mathcal{T})$ such that

- (H6) if $(g_1, g_2, P) \in \chi$, then $\{g_1 \uparrow, g_2 \uparrow\} \cap \Psi = \emptyset$.

With $\Gamma_{(C, \mathcal{T})}$, we denote the set of all Hintikka-pairs for (C, \mathcal{T}) . A path u (of length 1 or 2) is *enforced* by (Ψ, χ) iff either u is a node in χ or $\{\exists u, u'. P, \exists u', u. P\} \cap \Psi \neq \emptyset$ for some path u' and $P \in \{<, =\}$.

Intuitively, each node α of a (yet to be defined) Hintikka-tree T corresponds to a domain object a of the corresponding canonical model \mathcal{I} . The first component Ψ_α of the Hintikka-pair labelling α is the set of concepts from $\text{cl}(C, \mathcal{T})$ satisfied by a . The second component χ_α states restrictions on the relationship between concrete successors of a . If, for example, $(g_1, g_2, <) \in \chi_\alpha$, then we must have $g_1^{\mathcal{I}}(a) < g_2^{\mathcal{I}}(a)$. Note that the restrictions in χ_α are independent from concepts $\exists g_1, g_2. P \in \Psi_\alpha$. As will become clear when Hintikka-trees are defined, the restrictions in χ_α are used to ensure that the constraint graph induced by the Hintikka-tree T , which describes the concrete part of the model \mathcal{I} , does not contain a $<$ -cycle, i.e., that it is satisfiable. This induced constraint graph can be thought of as the union of smaller constraint graphs, each one being described by a Hintikka-pair labelling a node in T . These pair-graphs are defined next.

Definition 12 (Pair-graph). Let C be a concept, \mathcal{T} a TBox, and $p = (\Psi, \chi)$ a Hintikka-pair for (C, \mathcal{T}) . The *pair-graph* $G(p) = (V, E)$ of p is a constraint graph defined as follows:

1. V is the set of paths enforced by p
2. $E = \chi \cup \{(u_1, u_2, P) \mid \exists u_1, u_2. P \in \Psi\}$.

An *edge extension* of $G(p)$ is a set $E' \subseteq V \times V \times \{<, =\}$ such that for all $f.g_1, f.g_2 \in V$, we have either $(f.g_2, f.g_1, <) \in E'$ or $(f.g_1, f.g_2, P) \in E'$ for some $P \in \{<, =\}$. If E' is an edge extension of $G(p)$, then the graph $(V, E \cup E')$ is a *completion* of $G(p)$.

Note that, due to path normal form and the definitions of Hintikka-pairs and pair-graphs, we have $E' \cap E = \emptyset$ for every edge extension E' of a pair-graph (V, E) .

We briefly comment on the connection of completions and the χ -component of Hintikka-pairs. Let α and β be nodes in a Hintikka-tree T and let a and b be the corresponding domain objects in the corresponding canonical model \mathcal{I} . Edges in Hintikka-trees represent role-relationships, i.e., if β is successor of α in T , then there exists an $R \in N_R$ such that $(a, b) \in R^{\mathcal{I}}$. Assume β is successor of α and the edge between α and β represents relationship via the abstract feature f , i.e., we have $f^{\mathcal{I}}(a) = b$. The second component χ_β of

the Hintikka-pair labelling β fixes the relationships between all concrete successors of b that “ a talks about”. For example, if $(\exists f.g_1, g_2. =) \in \Psi_\alpha$ and $(\exists f.g_3, g_2. <) \in \Psi_\alpha$, where Ψ_α is the first component of the Hintikka-pair labelling α , then “ a talks about” the concrete g_1 -successor and the concrete g_3 -successor of b . Hence, χ_β either contains $(g_3, g_1, <)$ or (g_1, g_3, P) for some $P \in \{<, =\}$. This is formalized by demanding that the pair-graph $G(T(\alpha))$ of the Hintikka-pair labelling α together with all the edges from the χ -components of the successors of α are a completion of $G(T(\alpha))$. Moreover, this completion has to be satisfiable, which is necessary to ensure that the constraint graph induced by T does not contain a $<$ -cycle. An appropriate way of thinking about the χ -components is as follows: at α , a completion of $G(T(\alpha))$ is “guessed”. The additional edges are then “recorded” in the χ -components of the successor-nodes of α .

Definition 13 (Hintikka-tree). Let C be a concept, \mathcal{T} be a TBox, k the number of existential subconcepts in $\text{cl}(C, \mathcal{T})$, and ℓ be the number of abstract features in $\text{cl}(C, \mathcal{T})$. A $1+k+\ell$ -tuple of Hintikka-pairs $(p_0, \dots, p_{k+\ell})$ with $p_i = (\Psi_i, \chi_i)$ and $G(p_0) = (V, E)$ is called *matching* iff

- (H7) if $\exists R.D \in \Psi_0$ and $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D$, then $D \in \Psi_i$
- (H8) if $\{\exists R.D, \forall R.E\} \subseteq \Psi_0$ and $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D$, then $E \in \Psi_i$
- (H9) if $\exists f.D \in \Psi_0$ and $\mathcal{F}(C, \mathcal{T}, i) = f$, then $D \in \Psi_{k+i}$.
- (H10) if f is enforced by Ψ_0 , $\mathcal{F}(C, \mathcal{T}, i) = f$, and $\forall f.D \in \Psi_0$, then $D \in \Psi_{k+i}$.
- (H11) the constraint graph $(V, E \cup E')$ is a satisfiable completion of $G(p_0)$, where E' is defined as

$$\bigcup_{1 \leq i \leq \ell} \{(f.g_1, f.g_2, P) \mid \mathcal{F}(C, \mathcal{T}, i) = f, (g_1, g_2, P) \in \chi_{k+i}\}.$$

A $k + \ell$ -ary $\Gamma_{(C, \mathcal{T})}$ -tree T is a *Hintikka-tree* for (C, \mathcal{T}) iff $T(\alpha)$ is a Hintikka-pair for (C, \mathcal{T}) for each node α in T , and T satisfies the following conditions:

- (H12) $C \in \Psi_\epsilon$, where $T(\epsilon) = (\Psi_\epsilon, \chi_\epsilon)$,
- (H13) for all $\alpha \in \{1, \dots, k + \ell\}^*$, the tuple $(T(\alpha), T(\alpha_1), \dots, T(\alpha_j))$ with $j = k + \ell$ is matching.

For a Hintikka-tree T and node $\alpha \in \{1, \dots, k + \ell\}^*$ with $T(\alpha) = (\Psi, \chi)$, we use $T_{\triangleleft}(\alpha)$ to denote Ψ and $T_{\triangleright}(\alpha)$ to denote χ . Moreover, if $G(\alpha) = (V, E)$, we use $\text{cpl}(T, \alpha)$ to denote the constraint graph $(V, E \cup E')$ as defined in (H11).

Whereas most properties of Hintikka-trees deal with concepts, roles, and abstract features and are hardly surprising, (H11) ensures that constraint graphs induced by Hintikka-trees contain no $<$ -cycle. By “guessing” a completion as explained above, possible $<$ -cycles are anticipated and can be detected locally, i.e., it suffices to check that the completions $\text{cpl}(T, \alpha)$ are satisfiable as demanded by (H11). Indeed, it is crucial that the cycle detection is done by a *local* condition since we need to define an automaton which accepts exactly Hintikka-trees and automata work locally. It is worth noting that the localization of cycle detection as expressed by (H11) crucially depends on path normal form.

The following two lemmas show that Hintikka-trees are appropriate abstractions of models.

Lemma 14. *A concept C is satisfiable w.r.t. a TBox \mathcal{T} iff there exists a Hintikka-tree for (C, \mathcal{T}) .*

To prove decidability, it remains to define a looping automaton $\mathcal{A}_{(C, \mathcal{T})}$ for each concept C and TBox \mathcal{T} such that $\mathcal{A}_{(C, \mathcal{T})}$ accepts exactly the Hintikka-trees for (C, \mathcal{T}) .

Definition 15. Let C be a concept, \mathcal{T} be a TBox, k the number of existential subconcepts in $\text{cl}(C, \mathcal{T})$, and ℓ be the number of abstract features in $\text{cl}(C, \mathcal{T})$. The looping automaton $\mathcal{A}_{(C, \mathcal{T})} = (Q, \Gamma_{(C, \mathcal{T})}, \Delta, I)$ is defined as follows:

- $Q = \Gamma_{(C, \mathcal{T})}$, $I = \{(\Psi, \chi) \in Q \mid C \in \Psi\}$, and
- $((\Psi, \chi), (\Psi', \chi'), (\Psi_1, \chi_1), \dots, (\Psi_k, \chi_{k+\ell})) \in \Delta$ iff $(\Psi, \chi) = (\Psi', \chi')$ and $((\Psi, \chi), (\Psi_1, \chi_1), \dots, (\Psi_k, \chi_{k+\ell}))$ is matching.

Note that every state is an accepting state, and, hence, every run is accepting. The following lemma is easily obtained.

Lemma 16. *T is Hintikka-tree for (C, \mathcal{T}) iff $T \in L(\mathcal{A}_{C, \mathcal{T}})$.*

Since the size of $\text{cl}(C, \mathcal{T})$ is linear in the size of C and \mathcal{T} , it is straightforward to verify that the size of $\mathcal{A}_{(C, \mathcal{T})}$ is exponential in the size of C and \mathcal{T} . This, together with Lemmas 9, 14, and 16, and the polynomial decidability of the emptiness problem of looping automata [Vardi and Wolper, 1986], implies the upper bound given in the following theorem which states the main result of this paper. The lower bound is an immediate consequence of the fact that \mathcal{ALC} with general TBoxes is EXPTIME-hard [Schild, 1991].

Theorem 17. *Satisfiability and subsumption of \mathcal{TDL} -concepts w.r.t. TBoxes are EXPTIME-complete.*

5 Conclusion

There are several perspectives for future work of which we highlight three rather interesting ones: firstly, the presented decision procedure is only valid if a dense strict linear order is assumed as the underlying temporal structure. For example, the concept \top is satisfiable w.r.t. the TBox

$$\mathcal{T} = \{ \top \sqsubseteq \exists g_1, g_2, < \sqcap \exists g_1, f g_1, < \sqcap \exists f g_2, g_2, < \}$$

over the temporal structures \mathbb{Q} and \mathbb{R} (with the natural orderings) but not over \mathbb{N} . To see this, note that \mathcal{T} induces a constraint graph as in Figure 1. Hence, it would be interesting to investigate if and how the presented algorithm can be modified for reasoning with the temporal structure \mathbb{N} .

Secondly, \mathcal{TDL} does only allow for *qualitative* temporal reasoning. It would be interesting to extend the logic to mixed qualitative and quantitative reasoning by additionally admitting unary predicates $<_q$ and $=_q$ for each $q \in \mathbb{Q}$.

Thirdly, we plan to extend \mathcal{TDL} to make it suitable for reasoning about entity relationship (ER) diagrams. As demonstrated in, e.g., [Calvanese *et al.*, 1998; Artale and Franconi, 1999], Description Logics are well-suited for this task. By using an appropriate extension of \mathcal{TDL} , one should be able to capture a new kind of temporal reasoning with ER diagrams, namely reasoning over ER diagrams with “temporal” integrity constraints. For example, a temporal integrity constraint could state that employees birthdays should be before their employment date. An appropriate extension of

\mathcal{TDL} for this task could be by (unqualified) number restrictions, inverse roles, and a generalized version of the concrete domain constructor $\exists u_1, u_2.P$. An extension of the presented automata-theoretic decision procedure to this more complex logic seems possible.

Acknowledgements My thanks go to Franz Baader, Ulrike Sattler, and Stephan Tobies for fruitful discussions. The author was supported by the DFG Project BA1122/3-1 “Combinations of Modal and Description Logics”.

References

- [Allen, 1983] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.
- [Artale and Franconi, 1998] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research (JAIR)*, (9), 1998.
- [Artale and Franconi, 1999] A. Artale and E. Franconi. Temporal ER modeling with description logics. In *Proc. of ER'99*, Paris, France, 1999. Springer-Verlag.
- [Artale and Franconi, 2000] A. Artale and E. Franconi. Temporal description logics. In *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. MIT Press, To appear.
- [Baader and Hanschke, 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, pages 452–457, Sydney, Australia, 1991. Morgan Kaufmann Publ. Inc.
- [Calvanese *et al.*, 1998] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.
- [Lutz *et al.*, 1997] C. Lutz, V. Haarslev, and R. Möller. A concept language with role-forming predicate restrictions. Technical Report FBI-HH-M-276/97, University of Hamburg, Computer Science Department, Hamburg, 1997.
- [Lutz, 2000] C. Lutz. Interval-based Temporal Reasoning with General TBoxes. LTCS-Report LTCS-00-06, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See <http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html>.
- [Lutz, 2001] C. Lutz. NExpTime-complete description logics with concrete domains. In *Proc. of IJCAR 2001*, LNCS, Siena, Italy, 2001. Springer-Verlag.
- [Nebel, 1990] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
- [Sattler, 1998] U. Sattler. *Terminological knowledge representation systems in a process engineering application*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, 1998.
- [Schild, 1991] K. D. Schild. A correspondence theory for terminological logics. In *Proc. of IJCAI-91*, pages 466–471, Sidney, Australia, 1991. Morgan Kaufmann Publ. Inc.
- [Vardi and Wolper, 1986] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.
- [Vilain, 1982] M. Vilain. A system for reasoning about time. In *Proceedings of the Second AAAI*, pages 197–201, Pittsburgh, Pennsylvania, 1982.

KNOWLEDGE REPRESENTATION AND REASONING

BELIEF REVISION

On the Semantics of Knowledge Update

Chitta Baral

Dept of Computer Science & Engineering
Arizona State University
Tempe, AZ 85287, USA
E-mail: chitta@asu.edu

Yan Zhang

School of Computing & IT
University of Western Sydney
Penrith South DC, NSW 1797, Australia
E-mail: yan@cit.uws.edu.au

Abstract

We consider the problem of how an agent’s knowledge can be updated. We propose a formal method of knowledge update on the basis of the semantics of modal logic S5. In our method, an update is specified according to the minimal change on both the agent’s actual world and knowledge. We then investigate the semantics of knowledge update and characterize several specific forms of knowledge update which have important applications in reasoning about change of agents’ knowledge. We also discuss the persistence property of knowledge and ignorance associated with knowledge update.

1 Introduction and Motivation

The well-studied issues of belief updates and belief revision [Katsuno and Mendelzon, 1991] are concerned with the update and revision aspects of an agent’s belief with respect to new beliefs. The notion of belief update has been used, and often serves as a guideline [Karthi and Lifschitz, 1994], in reasoning about the effect of (*world altering*) actions on the state of the world. Thus if ϕ represents the agent’s belief about the world and the agent does an action that is supposed to make ψ true in the resulting world, then the agent’s belief about the resulting world can be described by $\phi \diamond \psi$, where \diamond is the update operator of choice.

Now let us consider reasoning about sensing actions [Scherl and Levesque, 1993; Son and Baral, 2000], which in their pure form, when executed, do not change the world, but change the agent’s knowledge about the world. Let *sense_f* be a sensing action whose effect is that after it is executed the agent knows whether f is true or not. This can be expressed as $Kf \vee K\neg f$, where K is the modal operator *Knows*. The current theory of belief updates does not tell us how to do updates with respect to such gain in knowledge due to a sensing action. (Note that we can not just have $\psi \equiv f \vee \neg f$ and use the the notion of belief update, as $f \vee \neg f$ is a tautology). One of our *goals* in this paper is to define a notion of *knowledge update*, analogous to belief update, where the original theory (ϕ) and the new theory (ψ) are in a language that can express knowledge. Such a notion would not only serve as a guideline to reason about pure and mixed sensing actions

in presence of constraints, but also allow us to reason about actions corresponding to *forgetting*, and *ignorance*.

The structure of the rest of the paper is as follows. In Section 2 we start with describing the particular modal logic that we plan to use in expressing knowledge, and describe the notion of k-models analogous to ‘models’ in classical logic. We then define closeness between k-models and use it to define a particular notion of knowledge update. In Section 3 we present alternative characterizations of four particular knowledge update cases – *gaining knowledge*, *ignorance*, *sensing*, and *forgetting*, and show their equivalence to our original notion of knowledge update. Some of these alternative characterizations are based on the formulation of reasoning about sensing actions, and thus our equivalence results serve as justification of the intuitiveness of our definition of knowledge update. In Section 4 we explore sufficiency conditions that guarantee persistence of knowledge (or ignorance) during a knowledge update. In Section 5, we conclude and discuss future directions.

2 Closeness between k-models and Knowledge Update

In this section, we describe formal definitions for knowledge update. Our formalization will be based on the semantics of the propositional modal logic S5 with a single agent. In general, under Kripke semantics, a *Kripke structure* is a triple (W, R, π) , where W is a set of possible worlds, R is an equivalence relation on W , and π is a truth assignment function that assigns a propositional valuation to each world in W . Given a Kripke structure $S = (W, R, \pi)$, a *Kripke interpretation* is a pair $M = (S, w)$, where $w \in W$ is referred to the *actual world* of M . The entailment relation \models between Kripke interpretations and formulas is defined to provide semantics for formulas of S5 [Fagin and et al, 1995].

In the case of single agent, however, we may restrict ourselves to those S5 structures in which the relation R is universal, i.e. each world is accessible from every world, and worlds are identified with the set of atoms true at the worlds [Meyer and van der Hoek, 1995]. To simplify a comparison between two worlds (e.g. Definition 2), we may view an atom $p \in w$ iff $w \models p$. Therefore, in our context a Kripke structure (W, R, π) is uniquely characterized by W and we may simplify a Kripke interpretation as a pair (W, w) which

we call a k -model, where w indicates the actual world of the agent and W presents all possible worlds that the agent may access. Note that w is in W for any k -model (W, w) .

In our following description, we use a, b, c, p, \dots to denote primitive propositional atoms; ϕ, ψ, v, \dots to denote propositional formulas without including modalities (we also call them *objective* formulas); and $\alpha, \beta, \mu, \dots$ and T to denote formulas that may contain modal operator K . For convenience, we also use $T \equiv \alpha_1 \wedge \dots \wedge \alpha_k$ to represent a finite set of formulas $\{\alpha_1, \dots, \alpha_k\}$ and call T a (*knowledge*) *set*.

Definition 1 (S5 Semantics) Let \mathcal{P} be the set of all primitive propositions in the language. The entailment relation \models under normal S5 semantics is defined as follows:

1. $(W, w) \models p$ iff p is primitive (i.e. $p \in \mathcal{P}$) and $w \models p$;
2. $(W, w) \models \alpha \wedge \beta$ iff $(W, w) \models \alpha$ and $(W, w) \models \beta$;
3. $(W, w) \models \neg \alpha$ iff it is not the case that $(W, w) \models \alpha$;
4. $(W, w) \models K\alpha$ iff $(W, w') \models \alpha$ for all $w' \in W$.

Given a formula T , $M = (W, w)$ is called a k -model of T if $M \models T$. We use $KMod(T)$ to denote the set of all k -models of T . For objective formulas ϕ , $Mod(\phi)$ denotes the set of models of ϕ , and $w \models \phi$ denotes that w is a *model* of ϕ . For a formula α , we say that T *entails* α , denoted as $T \models \alpha$, if for every k -model M of T , $M \models \alpha$.

Now the basic problem of knowledge update that we would like to investigate is formally described as follows: given a k -model $M = (W, w)$, that is usually viewed as a *knowledge state* of an agent, and a formula μ - the agent's new knowledge that may contain modal operator K , how do we update M to another k -model $M' = (W', w')$ such that $M' \models \mu$ and M' is *minimally different* from M with respect to some criterion. To approach this problem, we first need to provide a definition of *closeness* between two k -models with respect to a given k -model.

Definition 2 (k -model Closeness) Let $M = (W, w)$, $M_1 = (W_1, w_1)$ and $M_2 = (W_2, w_2)$ be three k -models. We say M_1 is as close to M as M_2 , denoted as $M_1 \leq_M M_2$, if:

1. $(w_1 \setminus w \cup w \setminus w_1) \subseteq (w_2 \setminus w \cup w \setminus w_2)$; or
2. $w_1 = w_2$ and one of the following conditions holds:
 - (i) if $W \subseteq W_1$, then (a) there exist some ϕ and ψ such that $M \models K\phi$ and $M_2 \not\models K\phi$ and $M \not\models K\psi$ and $M_2 \models K\psi$, or (b) for any ϕ if $M \models K\phi$ and $M_1 \not\models K\phi$, then $M_2 \not\models K\phi$;
 - (ii) if $W_1 \subset W$, then condition (a) above is satisfied, or (c) for any ϕ if $M \not\models K\phi$ and $M_1 \models K\phi$, then $M_2 \models K\phi$;
 - (iii) if $W \not\subseteq W_1$ and $W_1 \not\subseteq W$, then conditions (b) and (c) above are satisfied.

We denote $M_1 <_M M_2$ if $M_1 \leq_M M_2$ and $M_2 \not\leq_M M_1$.

In the above definition, condition 1 simply says that the symmetric differences between w and w_1 is not bigger than that between w and w_2 , while in condition 2, (i), (ii) and (iii) express that different preferences are applied to compare knowledge between M_1 and M_2 with respect to M . For convenience, given a k -model M , if we denote $KM =$

$\{\phi \mid \text{for all } w \in W, w \models \phi\}$, then (a) is equivalent to $KM \setminus KM_2 \neq \emptyset$ and $KM_2 \setminus KM \neq \emptyset$; (b) is equivalent to $KM \setminus KM_1 \subseteq KM \setminus KM_2$; and (c) is equivalent to $KM_1 \setminus KM \subseteq KM_2 \setminus KM$. Also (b) and (c) together present a difference on both knowledge decrease and increase between M_1 and M_2 in terms of M . It is also easy to see that \leq_M is a partial ordering.

Note that during the comparison between two k -models, we give preference to the change of the actual world over the change of the knowledge about the world. (The closeness criterion between actual worlds that we use is the commonly used criterion [Winslett, 1988] based on symmetric difference). For instance, if the actual world of a k -model M_1 is closer to the actual world of M than the actual world of another k -model M_2 , we will think M_1 is closer to M and the comparison of knowledge between M_1 and M_2 is ignored. Only when both M_1 and M_2 have the same actual world, we will compare the knowledge of M_1 and M_2 in terms of M . This seems to be intuitive to us. In fact, the comparison between actual worlds determines the *actual distance* between two k -models to the given k -model M . If M_1 and M_2 have the same actual distance to M , the *knowledge distance* is then taken into account.

Basically, condition (i) (or (ii) resp.) in Definition 2 defines a knowledge preference based on knowledge decrease (or increase resp.). That is, if M_1 *only* loses knowledge from M (or *only* gains some knowledge to M , resp.), then M_1 is preferred over those k -models that have both knowledge decrease and increase from M , i.e. (a), and also preferred over those k -models that only lose more knowledge from M (or add more knowledge to M , resp.) than M_1 does, i.e. (b) or (c) respectively. Condition (iii), on the other hand, deals with the mixed situation that M_1 has both knowledge decrease and increase from M . In this case, a combined difference on knowledge decrease and increase is applied to determine the knowledge distance, i.e. (b) and (c). Conditions (i), (ii) and (iii) can be illustrated by the following figures respectively.

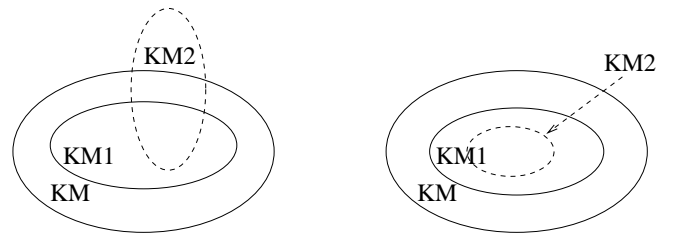


Figure 1: $M_1 \leq_M M_2$ under the condition $w_1 = w_2$ and $W \subseteq W_1$: (a) or (b) holds.

Definition 3 (k -model Update) Let $M = (W, w)$ be a k -model and μ a formula. A k -model $M' = (W', w')$ is called a possible resulting k -model after updating M with μ if and only if the following conditions hold:

1. $M' \models \mu$;
2. there does not exist another k -model $M'' = (W'', w'')$ such that $M'' \models \mu$ and $M'' <_M M'$.

We denote the set of all possible resulting k -models after updating M with μ as $Res(M, \mu)$.

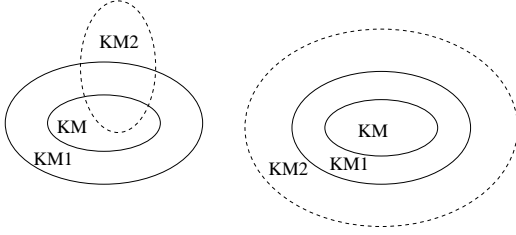


Figure 2: $M_1 \leq_M M_2$ under the condition $w_1 = w_2$ and $W_1 \subset W$: (a) or (c) holds.

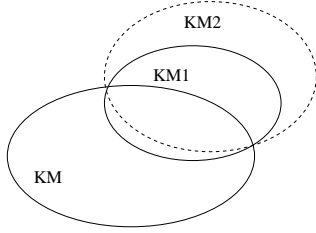


Figure 3: $M_1 \leq_M M_2$ under the condition $w_1 = w_2$, $W \not\subseteq W_1$ and $W_1 \not\subseteq W$: (b) and (c) hold.

Example 1 Let $T \equiv Kc \wedge \neg Ka \wedge \neg Kb \wedge K(a \vee b)$ and $\mu \equiv K\neg c$. We denote

$$\begin{aligned} w_0 &= \{a, b, c\}, w_1 = \{a, c\}, w_2 = \{b, c\}, \\ w_3 &= \{c\}, w_4 = \{a, b\}, w_5 = \{a\}, \\ w_6 &= \{b\}, w_7 = \emptyset. \end{aligned}$$

Clearly, $M_0 = (\{w_0, w_1, w_2\}, w_0)$ is a k -model of T . Consider the update of M_0 with μ . Let $M_1 = (\{w_4, w_5, w_6\}, w_4)$. Now we show that M_1 is a possible resulting k -model after updating M_0 with μ .

Since $(w_0 \setminus w_4 \cup w_4 \setminus w_0) = \{c\}$, we first consider any possible k -model $M' = (W', w')$ such that $(w_0 \setminus w' \cup w' \setminus w_0) \subset \{c\}$. Clearly, the only possible w' would be w_0 itself. Let $M' = (W', w_0)$, where W' is a subset of $\{w_0, \dots, w_7\}$. However, since $c \in w_0$, there does not exist any W' such that $M' \models K\neg c$. Therefore, from Definition 2, only condition 2 can be used to find a possible M' such that $M' <_M M_1$. So we assume $M' = (W', w_4)$. On the other hand, from M_0 and M_1 , it is easy to see that $KM_0 = \{c, a \vee b\}$ and $KM_1 = \{\neg c, a \vee b\}$ ¹. Then we have $KM_0 \setminus KM_1 = \{c\}$ and $KM_1 \setminus KM_0 = \{\neg c\}$. Ignoring the detailed verifications, we can show that there does not exist such $M' = (W', w_4)$ satisfying $KM_0 \setminus KM' = KM' \setminus KM_0 = \emptyset$. ■

Based on the k -model update, updating a formula (knowledge set) T in terms of another formula μ is then achieved by updating every k -model of T with μ .

¹For simplicity, here we only consider the *prime* formulas ϕ in KM in the sense that if $\phi \in KM$, then there is not another ψ such that $\models \psi \supset \phi$ and $\psi \in KM$.

Definition 4 (Knowledge Update) Let T and μ be two formulas. The update of T with μ , denoted as $T \diamond \mu$, is defined by $KMod(T \diamond \mu) = \bigcup_{M \in KMod(T)} Res(M, \mu)$.

3 Characterizing Knowledge Update

In this section, we first investigate basic properties of k -models, then provide alternative characterizations for different types of knowledge update and then show their equivalence with respect to our original characterization.

Proposition 1 Let $M_1 = (W_1, w_1)$ and $M_2 = (W_2, w_2)$ be two k -models. Then the following properties hold:

1. $\phi \in KM_1$ iff $W_1 \subseteq Mod(\phi)$;
2. $W_1 \subseteq W_2$ iff $KM_2 \subseteq KM_1$;
3. $KM_1 = KM_2$ iff $W_1 = W_2$;
4. Let $M' = (W_1 \cup W_2, w')$, then $KM' = KM_1 \cap KM_2$;
5. Let $w' \in W_1 \cap W_2$ and $M' = (W_1 \cap W_2, w')$, then $KM_1 \cup KM_2 \subseteq KM'$.

Given a set of k -models \mathcal{S} and a k -model M , let \leq_M be an ordering on \mathcal{S} as we defined in Definition 2. By $Min(\mathcal{S}, \leq_M)$ we mean the set of all elements in \mathcal{S} that are minimal with respect to ordering \leq_M . The following theorem shows that our knowledge update can be characterized by a minimal change criterion based on an ordering on k -models.

Theorem 1 Let T and μ be two formulas. Then $KMod(T \diamond \mu) = \bigcup_{M \in KMod(T)} Min(KMod(\mu), \leq_M)$.

Proof: To prove the result, we only need to show that for each k -model M of T , $Res(M, \mu) = Min(KMod(\mu), \leq_M)$. Let $M' \in Res(M, \mu)$. Since $M' \models \mu$, $M' \in KMod(\mu)$. On the other hand, according to Definition 3, for any $M'' \in KMod(\mu)$, we have $M' \leq_M M''$. That is, $M' \in Min(KMod(\mu), \leq_M)$. So $Res(M, \mu) \subseteq Min(KMod(\mu), \leq_M)$. Similarly, we can show $Min(KMod(\mu), \leq_M) \subseteq Res(M, \mu)$. ■

While the above theorem presents a general minimal change property of our knowledge update, we now give alternative characterizations of the update of T with μ when μ has some specific forms. These specific forms present some of the important features of knowledge update, and their alternative characterization is handy when the use of the notion of knowledge update becomes an overkill. For example, the alternative characterization of *sensing update* below is a much simpler characterization that is used in reasoning about sensing actions [Scherl and Levesque, 1993; Son and Baral, 2000]. We now introduce a notation that will be useful in our following discussions. Let W be a set of worlds and $w \in W$. By $W^{(w, \phi)}$, we denote the set $\{w' \mid w' \in W \text{ and } w' \models \phi \text{ iff } w \models \phi\}$.

Proposition 2 Given T and $K\phi$ where ϕ is objective and $T \models \phi$. Then $M' = (W', w')$ is a k -model of $T \diamond K\phi$ if and only if there exists a k -model $M = (W, w)$ of T such that $w = w'$ and $W' = W^{(w, \phi)}$.

The above proposition reveals an important property about knowledge update: to know some fact, the agent only needs to restrict the current possible worlds in each of her k -models, if this fact itself is already entailed by her current knowledge set. We call this kind of knowledge update *gaining knowledge update*.

Example 2 Let $T \equiv a \wedge \neg Ka \wedge Kb$. Suppose $w_0 = \{a, b\}$, $w_1 = \{a\}$, $w_2 = \{b\}$ and $w_3 = \emptyset$. Then T has one k -model $M = (\{w_0, w_2\}, w_0)$. Updating M with Ka , according to our k -model update definition, we have a unique resulting k -model $M' = (\{w_0\}, w_0)$. Indeed, this result is also obtained from Proposition 2. ■

As a contrary case to the gaining knowledge update, we now characterize an agent ignoring a fact from her knowledge set which we call *ignorance update*, i.e. updating T with $\neg K\phi$. From Definition 1, it is easy to see that $T \diamond \neg\phi \models \neg K\phi$. However, it should be noted that updating $T \diamond \neg\phi$ can *not* be used to achieve $T \diamond \neg K\phi$. Consider a k -model $M = (\{\{a, b\}, \{a\}\}, \{a, b\})$. Updating M with $\neg Ka$ we have a possible resulting k -model $M' = (\{\{a, b\}, \{a\}, \{b\}\}, \{a, b\})$, while updating M with $\neg a$ will lead to a possible result $M'' = (\{\{a, b\}, \{a\}, \{b\}\}, \{b\})$. Note that both M' and M'' entail $\neg Ka$, but $M' <_M M''$ according to Definition 2.

Proposition 3 Given T and $\mu \equiv \neg K\phi$ where ϕ is objective. $M' = (W', w')$ is a k -model of $T \diamond \neg K\phi$ if and only if there exists a k -model $M = (W, w)$ of T such that

- (i) if $M \models K\phi$, then $w' = w$ and $W' = W \cup \{w^*\}$, where $w^* \models \neg\phi$;
- (ii) otherwise, $w' = w$ and $W' = W$.

Example 3 Suppose $T \equiv \neg Ka \wedge \neg Kb \wedge K(a \vee b) \wedge Kc$ and the agent wants to ignore c . Let $w_0 = \{a, b, c\}$, $w_1 = \{a, c\}$, $w_2 = \{b, c\}$, $w_3 = \{c\}$, $w_4 = \{a, b\}$, $w_5 = \{a\}$, $w_6 = \{b\}$, $w_7 = \emptyset$. Clearly, T has three k -models: $M_0 = (\{w_0, w_1, w_2\}, w_0)$, $M_1 = (\{w_0, w_1, w_2\}, w_1)$, and $M_2 = (\{w_0, w_1, w_2\}, w_2)$. From Proposition 3, $T \diamond \neg Kc$ has the following twelve k -models: $(\{w_0, w_1, w_2, w_i\}, w_j)$, where $i = 4, 5, 6, 7$ and $j = 0, 1, 2$. ■

Now we consider the case when μ is of the form $K\phi \vee K\neg\phi$ where ϕ is objective. Updating T with this type of μ is particularly useful in reasoning about sensing actions [Scherl and Levesque, 1993; Son and Baral, 2000] where $K\phi \vee K\neg\phi$ represents the effect of a sensing action after whose execution, the agent will know either ϕ or its negation. We refer to such an update as a *sensing update*. The following proposition characterizes the update of T with a formula of the form $K\phi \vee K\neg\phi$. It is interesting to note that the sufficient and necessary condition for a k -model of $T \diamond (K\phi \vee K\neg\phi)$ is similar to the one presented in Proposition 2.

Proposition 4 Given T and $\mu \equiv K\phi \vee K\neg\phi$ where ϕ is objective. $M' = (W', w')$ is a k -model of $T \diamond (K\phi \vee K\neg\phi)$ if and only if there exists a k -model $M = (W, w)$ of T such that $w = w'$ and $W' = W^{(w, \phi)}$ or $w = w'$ and $W' = W^{(w, \neg\phi)}$

Proof: Let $\mu \equiv K\phi \vee K\neg\phi$ and $M' = (W^{(w, \phi)}, w)$. From the definition of $W^{(w, \phi)}$, it is easy to see that $M' \models \mu$.

Consider any $M'' = (W'', w'')$ where $M'' \models \mu$ and $w'' \neq w$. According to the condition 1 of Definition 2, $M' <_M M''$. So M'' can not be a k -model of $T \diamond \mu$. In other words, each k -model of $T \diamond \mu$ must have a form of $M'' = (W'', w)$. Then from Theorem 1, to prove the result, it is sufficient to prove for k -model $M'' = (W'', w)$ where $W'' \neq W^{(w, \phi)}$ or $W'' \neq W^{(w, \neg\phi)}$, $M' \leq_M M''$. This can be shown in the same way as in the proof of Proposition 2. ■

Example 4 Suppose $T \equiv Kb \wedge \neg Ka \wedge \neg K\neg a$ represents the current knowledge of an agent. Note that T implies that the agent does not have any knowledge about a . Consider the update of T with $\mu \equiv Ka \vee K\neg a$ which can be thought of as the agent trying to reason – in the planning or plan verification stage – about a sensing action² that will give her the knowledge about a . Let $w_0 = \{a, b\}$, $w_1 = \{b\}$, $w_2 = \{a\}$ and $w_3 = \emptyset$. It is easy to see that $M_0 = (\{w_0, w_1\}, w_0)$ and $M_1 = (\{w_0, w_1\}, w_1)$ are two k -models of T . Then according to the above proposition, it is obtained that $M'_0 = (\{w_0\}, w_0)$ and $M'_1 = (\{w_1\}, w_1)$ are the two k -models of $T \diamond \mu$. ■

As another important type of knowledge update, we consider the update of T with $\mu \equiv \neg K\phi \wedge \neg K\neg\phi$. This update can be thought of as the result of an agent *forgetting* her knowledge about the fact ϕ . We will refer to such an update as a *forgetting update*. The following proposition shows that in order to forget ϕ from T , for each k -model of the current knowledge set, the agent only needs to expand the set of possible worlds of this model with exactly *one specific* world.

Proposition 5 Given T and $\mu \equiv \neg K\phi \wedge \neg K\neg\phi$ where ϕ is objective. $M' = (W', w')$ is a k -model of $T \diamond \mu$ if and only if there exists a k -model $M = (W, w)$ of T such that

- (i) if $M \models K\phi$, then $w' = w$ and $W' = W \cup \{w^*\}$, where $w^* \models \neg\phi$;
- (ii) if $M \models K\neg\phi$, then $w' = w$ and $W' = W \cup \{w^*\}$, where $w^* \models \phi$;
- (iii) otherwise, $w' = w$ and $W' = W$.

Proof: Let $\mu \equiv \neg K\phi \wedge \neg K\neg\phi$ and $M = (W, w)$ be a k -model of T . Then it is easy to see that for two k -models $M' = (W', w')$ and $M'' = (W'', w'')$ such that $M' \models \mu$ and $M'' \models \mu$, and $w' = w$ and $w'' \neq w$, $M' <_M M''$. So M'' can not be a k -model of $T \diamond \mu$. In other words, a k -model of $T \diamond \mu$ must have a form $M' = (W', w)$.

From Theorem 1, to prove the result, we only need to show that for any k -model $M'' = (W'', w)$ such that $M'' \models \mu$ and $W'' \neq W \cup \{w^*\}$, $M' \leq_M M''$.

Let $M = (W \cup \{w^*\}, w)$, where $w^* \models \neg\phi$ if $M \models K\phi$ and $w^* \models \phi$ if $M \models K\neg\phi$. We first prove that for any k -model $M'' = (W'', w)$ such that $M'' \models \mu$ and W'' does not have a form of $W \cup \{w_i\}$, $M' \leq_M M''$.

Suppose $M \models K\phi$. Clearly $M' \models \mu$.

Case I. Consider a k -model $M'' = (W'', w)$ where $W' =$

²Such reasoning is necessary in creating plans with sensing actions or verifying such plans. On the other hand after the execution of a sensing action the agent exactly knows either a or $\neg a$, and can simply use the notion of belief update.

$W \cup \{w^*\} \subseteq W''$. Note that $M'' \models \mu$ as well. However, from Proposition 1, we have $KM'' \subseteq KM' \subseteq KM$. So $M' \leq_M M''$ according to Definition 2 (i.e. condition (b)).

Case 2. Suppose $W'' \subset W'$ (proper set inclusion). Without loss of generality, we assume that $W'' = W \cup \{w^*\} \setminus \{w_j\}$ where $w_j \in W$. This follows that $W \not\subseteq W''$ and $W'' \not\subseteq W$. So it is the case that $KM \setminus KM'' \neq \emptyset$ and $KM'' \setminus KM \neq \emptyset$. On the other hand, we have $W \subseteq W'$, from Definition 2 (i.e. condition (a)), we have $M' \leq_M M''$.

Case 3. Now suppose $W'' \not\subseteq W'$ and $W' \not\subseteq W''$. Without loss of generality, we can assume that $W'' = W \cup \{w^*, w_i\} \setminus \{w_j\}$, where $w_j \in W$ and $w^*, w_i \notin W$. Again, this results to the situation that $W \not\subseteq W''$ and $W'' \not\subseteq W$. From the above discussion, it implies that $M' \leq_M M''$.

Following the same way as above, we can prove that under the condition that $M \models K\neg\phi$ and $M' = (W \cup \{w^*\}, w)$ where $w^* \models \phi$, for any k -model $M'' = (W'', w)$ such that $M'' \models \mu$ and W'' does not have a form of $W \cup \{w_j\}$ $M' \leq_M M''$.

Now we show that for any k -model M'' that is of the form $M'' = (W \cup \{w_i\}, w)$ and w_i is any world such that $w_i \models \neg\phi$ if $M \models K\phi$ or $w_i \models \phi$ if $M \models K\neg\phi$, $M' \not\leq_M M''$ and $M'' \not\leq_M M'$. Suppose $M' \leq_M M''$. Since $W \subseteq W \cup \{w^*\}$, then according to Definition 2, condition (a) or (b) should be satisfied. As $W \subseteq W \cup \{w_i\}$, condition (a) can not be satisfied. So condition (b) must be satisfied. That is, for any ψ such that $M \models K\psi$ and $M' \not\models K\psi$, $M'' \models K\psi$. However, this implies that $KM \setminus KM' \subseteq KM'' \setminus KM'$, and also $KM \cap KM' \subseteq KM \cap KM''$. From Proposition 1 (Results 2 and 4), it follows that $W \cup W' \subseteq W \cup W''$, that is, $W \cup \{w^*\} \subseteq W \cup \{w_i\}$. Obviously, this is not true. Similarly, we can show that $M'' \not\leq_M M'$. That means, both M' and M'' are in $Res(M, \mu)$. This completes our proof. ■

Example 5 Suppose $T \equiv Kb \wedge (Ka \vee K\neg a)$ represents the current knowledge of an agent. After executing a forgetting action the agent now would like to update her knowledge with $\mu \equiv \neg Ka \wedge \neg K\neg a$. Let $w_0 = \{a, b\}$, $w_1 = \{b\}$, $w_2 = \{a\}$, $w_3 = \emptyset$. It is easy to see that $M_0 = (\{w_0\}, w_0)$ and $M_1 = (\{w_1\}, w_1)$ are the two k -models of T . Then using Proposition 5, we conclude that $M'_0 = (\{w_0, w_1\}, w_0)$, $M'_1 = (\{w_0, w_3\}, w_0)$, $M'_2 = (\{w_1, w_0\}, w_1)$, and $M'_3 = (\{w_1, w_2\}, w_1)$ are the four k -models of $T \diamond \mu$. Note that $(\{w_0, w_2\}, w_0)$ cannot be a k -model of $T \diamond \mu$ according to Proposition 5. ■

4 Persistence of Knowledge and Ignorance

Like most systems that do dynamic modeling, the knowledge update discussed previously is non-monotonic in the sense that while adding new knowledge into a knowledge set, some previous knowledge in the set might be lost. However, it is important to investigate classes of formulas that are persistent with respect to an update, as this may partially simplify the underlying inference problem [Engelfreit, 1998; Zhang, 1999]. Given T and μ , a formula α is said to be *persistent* with respect to the update of T with μ , if $T \models \alpha$ implies $T \diamond \mu \models \alpha$. If α is of the form $K\phi$, we call this persistence as *knowledge persistence*, while if α is of the form $\neg K\phi$, we

call it *ignorance persistence*. The question that we address now is that under what conditions, a formula α is persistent with respect to the update of T with μ .

As the update of T with μ is achieved based on the update of every k -model of T with μ , our task reduces to the study of persistence with respect to a k -model update. This is defined in the following definition.

Definition 5 (Persistence with respect to k -model Update)

Let μ and α be two formulas and M be a k -model. α is persistent with respect to the update of M with μ if for any $M' \in Res(M, \mu)$, $M \models \alpha$ implies $M' \models \alpha$.

Clearly a formula α is persistent with respect to the update of T with μ if and only if for each k -model M of T , α is persistent with respect to the update of M with μ . To characterize the persistence property with respect to k -model updates, we first define a preference ordering on k -models in terms of a formula.

Definition 6 (Formula Closeness) Let μ be a formula and M_1 and M_2 be two k -models. We say M_1 is as close to μ as M_2 , denoted as $M_1 \leq_\mu M_2$, if one of the following conditions holds:

1. $M_1 \in KMod(\mu)$;
2. if $M_1, M_2 \notin KMod(\mu)$, then for any $M \in KMod(\mu)$, $M_1 \leq_M M_2$.

We denote $M_1 <_\mu M_2$ if $M_1 \leq_\mu M_2$ and $M_2 \not\leq_\mu M_1$.

Intuitively, the above definition specifies a partial ordering to measure the closeness between two k -models to a formula. In particular, if M_1 is a k -model of μ , then M_1 is closer to μ than all other k -models (i.e. condition 1). If neither M_1 nor M_2 is a k -model of μ , then the comparison between M_1 and M_2 with respect to μ is defined based on the k -model preference ordering \leq_M for each k -model M of μ (i.e. condition 2). Note that if both M_1 and M_2 are k -models of μ , we have $M_1 \leq_\mu M_2$ and $M_2 \leq_\mu M_1$, and both of them are equally close to μ .

Example 6 Let $\mu \equiv Ka \wedge Kb$, $w_0 = \{a, b\}$, $w_1 = \{b\}$, $w_2 = \{a\}$ and $w_3 = \emptyset$. Clearly, μ has one k -model $M = (\{w_0\}, w_0)$. Consider two k -models $M_1 = (\{w_0, w_1\}, w_0)$ and $M_2 = (\{w_1, w_2\}, w_1)$. Now let us compare which one of them is closer to μ . Since neither M_1 nor M_2 is a k -model of μ , we can use condition 2 in Definition 6 to compare M_1 and M_2 . According to Definition 2, it is easy to see that $M_1 \leq_M M_2$ as $w_0 \setminus w_1 \cup w_1 \setminus w_0 = \{a\} \neq \emptyset$. Therefore, we conclude $M_1 \leq_\mu M_2$. Furthermore, we also have $M_1 <_\mu M_2$. ■

Proposition 6 Let μ be a formula. For any two k -models M_1 and M_2 , if $M_1 \leq_\mu M_2$, then $M_2 \models \mu$ implies $M_1 \models \mu$.

Proof: Suppose $M_2 \models \mu$. Then $M_2 \in KMod(\mu)$. From Definition 6, we know that for any other k -model M' , $M_2 \leq_M M'$. So $M_2 \leq_\mu M_1$. But we have $M_1 \leq_\mu M_2$. This implies that both M_1 and M_2 are equally close to μ . Hence, $M_1 \models \mu$. ■

Given a formula μ and a sequence of k -models M_1, \dots, M_k , if the relation $M_1 \leq_\mu M_2 \leq_\mu \dots \leq_\mu M_k$ holds, then it means that M_i is closer to μ than M_j , where

$i < j$. Now under this condition, if there is another formula α which satisfies the property that $M_j \models \alpha$ implies $M_i \models \alpha$ whenever $i < j$, we say that formula α is *persistent* with respect to formula μ . In other words, when k -models move closer to μ , α 's truth value is preserved in these k -models. The following definition formalizes this idea.

Definition 7 (\leq_μ -persistence) Let α, μ be two formulas. We say that α is \leq_μ -persistent if for any two k -models M_1 and M_2 , $M_2 \models \alpha$ and $M_1 \leq_\mu M_2$ implies $M_1 \models \alpha$.

Now we have the following important relationship between \leq_μ -persistence and k -model update persistence.

Theorem 2 Let α and μ be two formulas and M be a k -model. α is persistent with respect to the update of M with μ if α is \leq_μ -persistent.

Proof: Let M' be a k -model in $Res(M, \mu)$. Then we have $M' \in KMod(\mu)$. So for any k -model M'' , we have $M' \leq_\mu M''$. So $M' \leq_\mu M$. Now suppose α is μ -persistent. It follows that $M \models \alpha$ implies $M' \models \alpha$. As M' is an arbitrary k -model in $Res(M, \mu)$, we can conclude that α is persistent with respect to the update of M with μ . ■

From Theorem 2, we have that \leq_μ -persistence is a sufficient condition to guarantee a formula persistence with respect to a k -model update. As will be shown next, we can provide a unique characterization for μ -persistence. We first define the notion of ordering preservation as follows.

Definition 8 (Ordering Preservation) Given two formulas α and β . We say that ordering \leq_α preserves ordering \leq_β if for any two k -models M_1 and M_2 , $M_1 \leq_\alpha M_2$ implies $M_1 \leq_\beta M_2$.

The intuition behind ordering preservation is clear. That is, if \leq_α preserves \leq_β , then for any two k -models M_1 and M_2 , whenever M_1 is closer to α than M_2 , M_1 will be closer to β than M_2 as well. Finally, we have the following important result to characterize μ -persistence.

Theorem 3 Given two formulas α and μ , α is \leq_μ -persistent if and only if \leq_μ preserves \leq_α .

Proof: (\Rightarrow) Suppose α is \leq_μ -persistent. That is, for any two k -models M_1 and M_2 , $M_1 \leq_\mu M_2$ and $M_2 \models \alpha$ implies $M_1 \models \alpha$. So under the constraint that α is μ -persistent, whenever $M_1 \leq_\mu M_2$, we have $M_1 \leq_\alpha M_2$. That means, \leq_μ preserves \leq_α .

(\Leftarrow) Suppose \leq_μ preserves \leq_α . From Definition 8, we have that for any two k -models M_1 and M_2 , $M_1 \leq_\mu M_2$ implies $M_1 \leq_\alpha M_2$. Now suppose $M_1 \leq_\mu M_2$. So we have $M_1 \leq_\alpha M_2$. From Proposition 6, we have that $M_2 \models \alpha$ implies $M_1 \models \alpha$. From this it follows that α is \leq_μ -persistent. ■

5 Conclusion and Future Directions

In this paper we developed the notion of knowledge update – an analogous notion to belief update, that is useful in characterizing the knowledge change of an agent in presence of new knowledge. Our notion is particularly relevant in reasoning about actions and plan verifications when there are sensing

actions. We presented simpler alternative characterization of knowledge update for particular cases, and showed its equivalence to the original characterization. Finally we discussed when particular knowledge (or ignorance) persists with respect to a knowledge update.

We believe our work here to be a starting point on knowledge update, and as evident from the research in belief update and revision in the past decade. A lot remains to be done in knowledge update. For example, issues such as iterative knowledge update, abductive knowledge update, minimal knowledge in knowledge update, etc. remain to be explored. Similarly, in regards to reasoning about actions, additional specific cases of knowledge update need to be identified and simpler alternative characterization for them will be needed to be developed. Finally, we are currently working on establishing complexity results for knowledge update.

Acknowledgement

The authors thank Norman Foo and Abhaya Nayak for useful discussions on this topic. The research of the first author was supported by NSF under grants IRI 9501577 and 0070463, and part of this research was performed while he was visiting the University of Western Sydney. The research of the second author was supported in part by Australian Research Council under grant A49803542.

References

- [Engelfreit, 1998] J. Engelfreit. Monotonicity and persistence in preferential logics. *Journal of Artificial Intelligence Research*, 8:1–12, 1998.
- [Fagin and et al, 1995] R. Fagin and et al. *Reasoning about Knowledge*. MIT press, 1995.
- [Kartha and Lifschitz, 1994] G. Kartha and V. Lifschitz. Actions with indirect effects: Preliminary report. In *Proceedings of KR-94*, pages 341–350, 1994.
- [Katsuno and Mendelzon, 1991] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of KR-91*, pages 387–394, 1991.
- [Meyer and van der Hoek, 1995] J-J. Meyer and W. van der Hoek. Epistemic logics for computer science and artificial intelligence. In *Cambridge Tracts in Theoretical Computer Science*, 41. Cambridge University Press, 1995.
- [Scherl and Levesque, 1993] R. Scherl and H. Levesque. The frame problem and knowledge producing actions. In *Proceedings of AAAI-93*, pages 689–695, 1993.
- [Son and Baral, 2000] T. Son and C. Baral. Formalizing sensing actions: a transition function based approach. *Artificial Intelligence*, 125:19–92, 2000.
- [Winslett, 1988] M. Winslett. Reasoning about action using a possible models approach. In *Proceedings of AAAI-88*, pages 89–93, 1988.
- [Zhang, 1999] Y. Zhang. Monotonicity in rule based update. In *Proceedings of the 1999 International Conference on Logic Programming (ICLP'99)*, pages 471–485. MIT Press, 1999.

Resource-bounded inference from inconsistent belief bases

Pierre Marquis and Nadège Porquet

CRIL / Université d'Artois
rue de l'Université - S.P. 16
F-62307 Lens Cedex, France
{marquis, porquet}@cril.univ-artois.fr

Abstract

A family of resource-bounded paraconsistent inference relations is introduced. These relations are based on $S - 3$ entailment, an inference relation logically weaker than classical entailment and parametrized by a set S of variables. Their properties are investigated, especially from the computational complexity point of view. Among the strong features of our framework is the fact that tractability is ensured each time $|S|$ is bounded and that binary connectives behave in a classical manner. Moreover, our family is large enough to include both $S - 3$ inference, the standard inference relations based on the selection of consistent subbases and some additional forms of paraconsistent reasoning as specific cases.

1 Introduction

Few would dispute the fact that classical entailment is not suited to common-sense inference. One of its main drawbacks is its inability to deal with inconsistency. Indeed, any formula is a logical consequence of a contradiction. Such a trivialization of inference is often referred to as *ex falso quodlibet sequitur* and classical entailment is said to be explosive (i.e., not paraconsistent).

Many approaches have been proposed so far to address this issue. Some of them, like belief revision, aim at preventing inconsistencies from being generated; other approaches aim at removing inconsistencies once they appeared. Complementary to them, many approaches deal with inconsistencies. Among them are various paraconsistent logics, argumentative logics, belief merging and the so-called coherence-based approach¹ to inconsistency handling. Unfortunately, existing paraconsistent inference relations are typically intractable (see [Cayrol *et al.*, 1998] [Nebel, 1998]).

In this paper, we show that trivialization in presence of inconsistency and intractability can be handled within a uniform framework. A family of paraconsistent resource-bounded inference relations is presented. Our starting point

¹In this approach, beliefs are represented by a belief base B , i.e., a finite set of propositional formulas; some consistent subbases of B that are preferred w.r.t. a given selection policy are first selected, then inference amounts to classical inference from some of them [Pinkas and Loui, 1992] [Benferhat *et al.*, 1993].

is the notion of $S - 3$ inference [Schaerf and Cadoli, 1995] and the coherence-based approach to inconsistency handling [Pinkas and Loui, 1992] [Benferhat *et al.*, 1993]. In $S - 3$ logic, every propositional symbol is interpreted in a weak way, allowing it to be both true and false at the same time, except for the variables of a given set S that are interpreted classically. Tractability is ensured by limiting the size of S . Interestingly, $S - 3$ entailment proves sufficient to handle some inconsistencies, those that are not reachable when classical inference is limited to the variables of S . In order to handle the remaining inconsistencies while avoiding trivialization, we introduce a family of inference relations which are to $S - 3$ entailment what the inference relations of the standard coherence-based approach are to classical entailment. Specifically, consistency is restored by removing variables (from S) instead of removing some explicit beliefs from the belief base. The computational complexity of these inference relations is identified, and some other properties are investigated. Among the strong features of our framework is the fact that tractability is ensured each time a computational bound $|S|$ is set. Moreover, binary connectives behave in a classical manner while it is not the case for many paraconsistent logics. Finally, our framework is general enough to encompass both $S - 3$ logic [Schaerf and Cadoli, 1995], the standard coherence-based approach to inconsistency handling [Pinkas and Loui, 1992] [Benferhat *et al.*, 1993] and some systems for paraconsistent reasoning given in [Priest, 1991] [Besnard and Schaub, 1998] as specific cases.

2 Formal preliminaries

In the following, we consider a propositional language $PROP_{PS}$ inductively defined from a finite set PS of propositional symbols, the boolean constants *true* and *false* and the connectives \neg, \wedge, \vee . For every formula ϕ from $PROP_{PS}$, $Var(\phi)$ denotes the symbols of PS occurring in ϕ . As usual, every finite set of formulas is considered as the conjunctive formula whose conjuncts are the elements of the set.

Every formula ϕ from $PROP_{PS}$ is interpreted in a classical way, unless stated otherwise. In order to avoid any ambiguity, we sometimes refer to 2-interpretations, 2-models, 2-consistent and 2-inconsistent formulas instead of the usual notions of (respectively) interpretations, models, consistent formulas and inconsistent formulas.

For the sake of simplicity, we assume that every formula ϕ of $PROP_{PS}$ is in Negation Normal Form (NNF), i.e., only

variables are in the scope of any occurrence of \neg in ϕ . For instance, $\neg a \vee (b \wedge \neg c)$ is in NNF while the (classically equivalent) formula $\neg(a \wedge \neg(b \wedge \neg c))$ is not. Nevertheless, this assumption could be given up easily (cf. Section 4.4).

We assume the reader familiar with the complexity classes \mathcal{P} , $\Delta_2^p[\mathcal{O}(\log n)]$, Δ_2^p , and Π_2^p of the polynomial hierarchy (see [Papadimitriou, 1994] for details).

3 $S - 3$ inference

Our family of inference relations is based on $S - 3$ logic [Schaerf and Cadoli, 1995], a multivalued logic which can be viewed as a generalization of Levesque's logic of limited inference [Levesque, 1984]. In $S - 3$ logic, every propositional variable is interpreted in a weak way, allowing it to be both true and false at the same time, except for the variables of a given set S .

Definition 3.1 Let S be a subset of PS .

- An $S - 3$ -interpretation over PS is a mapping I from the set L_{PS} of all literals over PS to $BOOL = \{0, 1\}$ s.t. for every literal l of L_{PS} , we never have $I(l) = I(\neg l) = 0$, and we always have $I(l) = 1 - I(\neg l)$ whenever the variable of l belongs to S . true is always interpreted as 1 while false is always interpreted as 0.
- An $S - 3$ -interpretation I is an $S - 3$ -model of a NNF formula Σ iff the formula obtained by replacing in Σ every occurrence of $\neg l$ by false (resp. true) when $I(\neg l) = 0$ (resp. 1), then every occurrence of l by false (resp. true) when $I(l) = 0$ (resp. 1) evaluates classically to 1.
- Let B be a belief base (i.e., a conjunctively-interpreted finite set of formulas from $PROPPS$). Let γ be a formula from $PROPPS$. γ is an $S - 3$ consequence of B , noted $B \models_S^3 \gamma$, iff every $S - 3$ -model of B is an ($S - 3$) model of γ .

A formula is said to be $S - 3$ consistent when it has an $S - 3$ -model. A 3-interpretation (resp. 3-model) is just an $\emptyset - 3$ -interpretation (resp. $\emptyset - 3$ -model) and a formula is 3-consistent when it has a 3-model. Every formula from $PROPPS$ which does not contain any occurrence of false is 3-consistent (the 3-interpretation I s.t. $I(l) = 1$ for every literal l of L_{PS} is a 3-model of it).

As shown in [Schaerf and Cadoli, 1995], for every $S \subseteq PS$, the corresponding inference relation \models_S^3 is an approximation of \models by below: if $B \models_S^3 \gamma$ then $B \models \gamma$.

Interestingly, the limited power of $S - 3$ inference enables some inconsistencies to be handled:

Example 3.1 Let $B = \{(\neg a \vee b), a, (\neg c \vee b), (\neg b \vee d), (c \wedge \neg d), e, (\neg e \vee f)\}$. B is (2-)inconsistent.

- Let $S = \{c, d, e\}$. We have $B \models_S^3 a \wedge c \wedge \neg d \wedge b \wedge \neg b \wedge e \wedge f$ but $B \not\models_S^3 d$.
- Let $S = \{b, c, d, e\}$. We have $B \models_S^3$ false. Here, $B \models_S^3$ is explosive: any formula is an $S - 3$ consequence of B .

Tractability of $S - 3$ entailment is ensured by limiting the size of S . To be more specific, the time complexity of $S - 3$ inference is in $\mathcal{O}(2^{|S|} \cdot |B| \cdot |\gamma|)$ when B is a CNF formula²

²Combining both Theorem A.3 from [Schaerf and Cadoli, 1995], Lemma 3.1 and Theorem 4.1 from [Cadoli and Schaerf, 1996], this

and γ is a clause [Schaerf and Cadoli, 1995]. Accordingly, the complexity of $S - 3$ inference depends essentially on the number of variables in S .

4 Resource-bounded paraconsistent inference

4.1 Dealing with inconsistency

Adhering to $S - 3$ entailment is a way to prevent some inconsistencies from being harmful, namely those which are derivable only if classical reasoning over some variables outside S is performed. In the previous example with $S = \{c, d, e\}$, while we have both $B \models_S^3 b$ and $B \models_S^3 \neg b$, we do **not** have $B \models_S^3 d$.

However, restricting S to an arbitrary subset of the variables occurring in B is not sufficient to avoid trivialization in every situation (see the previous example). In order to deal with the remaining inconsistencies, we suggest to focus on some subsets S' of S , those for which the corresponding inference relations are not explosive. The approach is similar to the standard coherence-based approach to inconsistency handling, except that the inference relation is weakened by removing variables from S instead of removing explicit beliefs from B . Formally, we first need a notion playing a role similar to the notion of consistent subbase in the standard coherence-based approach.

Definition 4.1 Let B be a belief base. Let $S \subseteq PS$ and $S_0 \subseteq S$ s.t. $B \not\models_{S_0}^3$ false. A consistent subset S' of S w.r.t. B and S_0 is a subset of S containing S_0 and s.t. $B \not\models_{S'}^3$ false. $\mathcal{S}(B, S, S_0)$ denotes the set of all consistent subsets of S w.r.t. B and S_0 .

In this definition, S_0 is a given set of variables which *must* be interpreted classically. In our framework, S_0 plays a role similar to the one played by integrity constraints. Requiring the existence of such a set S_0 (possibly empty) imposes that B is 3-consistent (which is not a strong assumption).

In the following, we adhere to a skeptical approach (every preferred consistent subset is considered):

Definition 4.2 Let B be a belief base and S_0 be a subset of PS s.t. $B \not\models_{S_0}^3$ false. Let \mathcal{P} be a selection policy s.t. $\mathcal{S}_{\mathcal{P}}(B, S, S_0)$ is a subset of $\mathcal{S}(B, S, S_0)$ and let γ be a formula from $PROPPS$. γ is a consequence of B w.r.t. \mathcal{P} and S_0 , noted $B \approx_{S, S_0}^{\mathcal{P}} \gamma$, iff $\forall S' \in \mathcal{S}_{\mathcal{P}}(B, S, S_0)$, $B \models_{S'}^3 \gamma$.

Now, we can define selection policies for consistent subsets that are similar to the ones defined for consistent subbases in the standard coherence-based approach [Pinkas and Loui, 1992] [Benferhat et al., 1993] [Benferhat et al., 1995].

Definition 4.3 Let B be a belief base. Let $S = (S_0, \dots, S_j)$ be a stratification³ of a given subset of PS where S_0 is s.t. $B \not\models_{S_0}^3$ false.

- The set $\mathcal{S}_{\mathcal{P}\mathcal{O}}(B, S, S_0)$ of all preferred consistent subsets of S w.r.t. B and S_0 for the possibilistic policy is the singleton $\{\bigcup_{i=0}^{s-1} S_i\}$, where s is either the smallest index ($1 \leq s \leq j$) s.t. $\bigcup_{i=0}^s S_i$ is an inconsistent subset of S w.r.t. B and S_0 or $j + 1$ if S is consistent w.r.t. B .

result can be easily generalized to the case where B is any NNF formula and γ is a CNF formula; in this situation, $S - 3$ inference can still be decided in time linear in the size of the belief base.

³A stratification of S is a totally ordered partition of it.

- The set $\mathcal{S}_{\mathcal{LO}}(B, S, S_0)$ of all preferred consistent subsets of S w.r.t. B and S_0 for the linear order policy is the singleton $\{\bigcup_{i=0}^j S'_i\}$, where S'_i ($i \in 0, \dots, j$) is defined by $S'_i = S_i$ if $S_i \cup \bigcup_{l=0}^{i-1} S'_l$ is a consistent subset of S w.r.t. B and S_0 , \emptyset otherwise.
- The set $\mathcal{S}_{\mathcal{IP}}(B, S, S_0)$ of all preferred consistent subsets of S w.r.t. B and S_0 for the inclusion-preference policy is $\{S' \mid S' \text{ is a consistent subset of } S \text{ w.r.t. } B \text{ and } S_0 \text{ and } \forall S'' \neq S' \text{ s.t. } S'' \in \mathcal{S}(B, S, S_0), \forall i \in 0, \dots, j ((\forall l < i (S' \cap S_l = S'' \cap S_l)) \Rightarrow S' \cap S_i \not\subseteq S'' \cap S_i)\}$.
- The set $\mathcal{S}_{\mathcal{LE}}(B, S, S_0)$ of all preferred consistent subsets of S w.r.t. B and S_0 for the lexicographic policy is $\{S' \mid S' \text{ is a consistent subset of } S \text{ w.r.t. } B \text{ and } S_0 \text{ and } \forall S'' \neq S' \text{ s.t. } S'' \in \mathcal{S}(B, S, S_0), \forall i \in 0, \dots, j ((\forall l < i (|S' \cap S_l| = |S'' \cap S_l|)) \Rightarrow |S' \cap S_i| \geq |S'' \cap S_i|)\}$.

These definitions assume that a stratification of the given set S of variables is available. Such a stratification is the formal counterpart of the preferential information used to discriminate among consistent subsets.

Here is an example showing the variety of conclusions that can be drawn in our framework (if both a formula γ and its negation are consequences of B , the truth of γ given B must be considered as doubtful):

Example 4.1 Let $B = \{(\neg a \vee b), a, (\neg c \vee b), (\neg b \vee d), (c \wedge \neg d), e, (\neg e \vee f)\}$. With $S = (\emptyset, \{a, b, c, d, e\})$ or $S = (\emptyset, \{a\}, \{b, c, d\}, \{e\})$, $a, c, \neg d$ and e are consequences of B (but their negations are not) whatever \mathcal{P} among the four policies considered above. Moreover:

- With $S = (\emptyset, \{a, b, c, d, e\})$:
 - w.r.t. \mathcal{PO} (or \mathcal{LO}), none of $b, \neg b, f$ and $\neg f$ are consequences of B .
 - w.r.t. \mathcal{IP} , f is a consequence of B , but b and $\neg b$ are not.
 - w.r.t. \mathcal{LE} , b and f are consequences of B , but their negations are not.
- With $S = (\emptyset, \{a\}, \{b, c, d\}, \{e\})$:
 - w.r.t. \mathcal{PO} , b is a consequence of B , but f and $\neg f$ are not.
 - w.r.t. $\mathcal{LO}, \mathcal{IP}$ or \mathcal{LE} , b and f are consequences of B , but their negations are not.

Clearly enough, in the case where $B \not\models_S^3$ false, all our inference relations $B \approx_S^{\mathcal{P}, S_0}$ coincides with $B \models_S^3$. Accordingly, $S - 3$ inference can be recovered as a specific case in our framework in this situation.

The next proposition states that one of our basic objectives (avoiding trivialization) is always reached.

Proposition 4.1 None of the relations $B \approx_S^{\mathcal{P}, S_0}$ with \mathcal{P} among $\mathcal{PO}, \mathcal{LO}, \mathcal{IP}, \mathcal{LE}$ is explosive.

4.2 Instantiating our framework

There are many ways to define a stratified set S from a stratified belief base (SBB) B ⁴. First, S_0 can be defined as the set

⁴Upstream to our work are several techniques that aim at deriving a SBB from a “flat” belief base, see [Benferhat, 2000] for a survey.

of all variables from the certain beliefs (i.e., pieces of knowledge) of B when available, provided that such a set is small enough and satisfies $B \not\models_{S_0}^3$ false. Otherwise, S_0 is defined as \emptyset .

Defining S is known as a difficult problem in the general case [Schaerf and Cadoli, 1995]. Interestingly, when B is a SBB, it is possible to exploit the given preferential information so as to define S . A very natural way consists in considering in priority the variables of the most plausible beliefs until a preset bound b on the size of S is reached. Formally, we consider the following set of variables S^b :

Definition 4.4 Let $B = (\Delta_1, \dots, \Delta_k)$ be a SBB. Let b be a positive integer. Let S_0 be a subset of PS s.t. $B \not\models_{S_0}^3$ false and $|S_0| \leq b$. The inference level⁵ of B given b and S_0 , noted i_B^b is the smallest i ($i \in 0..k-1$) s.t. $S_i^b = S_{i+1}^b$, where the sequence (S_i^b) is inductively defined by $S_0^b = S_0$ and $(\forall i \in 1, \dots, k-1) S_i^b = S_{i-1}^b \cup Var(\Delta_i)$ if $|S_{i-1}^b \cup Var(\Delta_i)| \leq b$, $S_i^b = S_{i-1}^b$ otherwise. We define S^b as $S_{i_B^b}^b$.

Finally, once both S_0 and a flat set S are available, a stratification of S can be easily derived from the given stratification of B . Basically, once S_0 has been taken into account, it is reasonable to give more importance to the variables of S that belong to the more plausible beliefs. Loosely speaking, we want to reason as much as classically as possible from the most plausible beliefs in order to take into account their, possibly conflicting, classical consequences: inconsistencies must not be ignored when supported by plausible beliefs.

Definition 4.5 Let $B = (\Delta_1, \dots, \Delta_k)$ be a SBB, S a subset of PS and S_0 a subset of S . The stratification of S induced by B given S_0 is $S = (S_0, \dots, S_{k+1})$ s.t. $S_i = (S \cap Var(\Delta_i)) \setminus \bigcup_{l=0}^{i-1} S_l$ for every $i \in 1..k$, and $S_{k+1} = S \setminus \bigcup_{l=0}^k S_l$.

There are many other ways to derive a stratification of S . For instance, in the situation where we are ready to give more credit to facts than to rules, we can take advantage of the notion of degree of definiteness⁶ of a variable (as defined in [Besnard and Schaub, 1998]) to achieve this goal from a CNF belief base B .

4.3 Logical properties and cautiousness

Some logical properties. As in the standard coherence-based approach, it is easy to give a preferential models semantics to each of our relations $\approx_S^{\mathcal{P}, S_0}$. Indeed, $B \approx_S^{\mathcal{P}, S_0} \gamma$ holds iff every preferred $(S_0 -)$ 3-model of B w.r.t. \mathcal{P} is a $(S_0 -)$ 3-model of γ , where the preferred $(S_0 -)$ 3-models of B w.r.t. \mathcal{P} are exactly the $S' - 3$ -models of B with $S' \in \mathcal{S}_{\mathcal{P}}(B, S, S_0)$. Thus, when $S = (\emptyset, PS)$, the preferred 3-models of B w.r.t. \mathcal{IP} are the 3-models of it encoding worlds that are “as normal as possible” (i.e., as close as possible to 2-interpretations).

⁵At a first glance, the computation of the inference level of a belief base looks like the computation of the inconsistency level for possibilistic belief bases. However, the former is guided by computational motivations, only; especially, it is not the case that $B \not\models_{S^b}^3$ false in the general case.

⁶Roughly, every variable x of S occurring in B can be associated with the size of one of the smallest clauses of B containing x ; a stratification of S is obtained by considering x before y whenever the score attached to x is smaller than the score attached to y .

Another strong point of our approach is that our inference relations exhibit a limited form of syntax-sensitivity. Indeed, the binary connectives \wedge and \vee of our language behave in a classical manner: the set of $S-3$ models of $\phi \wedge \psi$ (resp. $\phi \vee \psi$) is the intersection (resp. union) of the set of $S-3$ models of ϕ and the set of $S-3$ models of ψ . As a consequence, given a stratification $S = (S_0, \dots, S_j)$, every formula from B (and B itself!) can be turned into a CNF formula without modifying the corresponding inference relations $\approx_S^{\mathcal{P}, S_0}$. The possibility of normalizing the belief base under CNF is particularly important for readability reasons and the practical computing of the inference relations $\approx_S^{\mathcal{P}, S_0}$. Moreover, this valuable property is usually not satisfied by paraconsistent logics (e.g., the one given in [da Costa, 1974], the one given in [Lin, 1987], or the standard coherence-based approach). Contrastingly, like any paraconsistent inference relation, our relations do not satisfy left logical equivalence. For the same reason, they do not satisfy right weakening (since they satisfy reflexivity). Finally, unlike $S-3$ inference [Schaerf and Cadoli, 1995], they are not monotonic⁷ (neither in S nor in B) in the general case.

Example 4.2 Let $B = \{a, (\neg a \vee b), (\neg a \vee c), \neg c\}$, $B' = \{a, (\neg a \vee b), (\neg a \vee c), \neg c, \neg a\}$, $S = (\emptyset, \{a\})$, and $S' = (\emptyset, \{a, c\})$. We have $B \approx_S^{\mathcal{I}\mathcal{P}, S_0} b$ but $B' \not\approx_S^{\mathcal{I}\mathcal{P}, S_0} b$ and $B \not\approx_{S'}^{\mathcal{I}\mathcal{P}, S_0} b$. Similar counter-examples can be easily found for the other policies.

This is not very surprising: while monotonicity is a highly desirable property when B is consistent, this is not the case otherwise. It is natural that the epistemic status of beliefs changes when more evidence is got through the incorporation of additional explicit beliefs or through an improvement of inferential capabilities. Specifically, increasing the computational effort can lead to discover inconsistencies that were hidden before.

Cautiousness. In the spectrum of inference relations in our framework, one extreme bound corresponds to the case $S = \emptyset$, while at the other extreme bound, we have $S = PS$. None of these two extreme cases can be considered as a “reasonable” inference relation for common-sense reasoning. On the one hand, the latter case corresponds to an ideal agent, with unlimited inferential capabilities: \models_{PS}^3 coincides with \models . Accordingly, if B is 2-consistent, all our inference relations $B \approx_{PS}^{\mathcal{P}, S_0}$ coincide with $B \models$ (admittedly, this contrasts with many paraconsistent inference relations). On the other hand, the former case corresponds to an agent with very limited reasoning capabilities. Indeed, if $S = \emptyset$, all our inference relations correspond to Levesque’s 3-entailment [Levesque, 1984]. This inference relation is tractable and avoids trivialization (as long as B does not contain any occurrence of *false*). However, with this inference relation, disjunctive syllogism can never be applied: it is not possible to conclude b from $a \wedge (\neg a \vee b)$. This suggests that cautiousness must be taken into account as an important criterion in the choice of a relation.

⁷Nevertheless, we have a “stratumwise” monotonicity property: for $\mathcal{P} \in \{\mathcal{P}\mathcal{O}, \mathcal{L}\mathcal{O}, \mathcal{I}\mathcal{P}, \mathcal{L}\mathcal{E}\}$, for every $i \in 0, \dots, j-1$, if $B \approx_{(S_0, \dots, S_i)}^{\mathcal{P}, S_0} \gamma$, then $B \approx_{(S_0, \dots, S_{i+1})}^{\mathcal{P}, S_0} \gamma$. This property is helpful to check whether $B \approx_S^{\mathcal{P}, S_0} \gamma$ in an incremental fashion.

In the case where S is a consistent subset, all our inference relations coincide with \models_S^3 , so they are just as cautious as \models_S^3 , and the size of S can be viewed as a degree of cautiousness. In the remaining case, our inference relations are more cautious relations than \models_S^3 but this is what is expected since trivialization must be avoided. In this situation, our inference relations do not coincide in the general case, and results similar to the corresponding ones in the standard coherence-based approach can be obtained; especially, the less cautious inference relations correspond to the $\mathcal{L}\mathcal{O}$ and $\mathcal{L}\mathcal{E}$ policies.

Proposition 4.2

- $\approx_S^{\mathcal{P}\mathcal{O}, S_0}$ is more cautious than $\approx_S^{\mathcal{L}\mathcal{O}, S_0}$ and $\approx_S^{\mathcal{I}\mathcal{P}, S_0}$, but the converse does not always hold.
- $\approx_S^{\mathcal{I}\mathcal{P}, S_0}$ is more cautious than $\approx_S^{\mathcal{L}\mathcal{E}, S_0}$ but the converse does not always hold.
- $\approx_S^{\mathcal{L}\mathcal{O}, S_0}$ cannot be compared with $\approx_S^{\mathcal{I}\mathcal{P}, S_0}$ and $\approx_S^{\mathcal{L}\mathcal{E}, S_0}$ w.r.t. cautiousness in the general case.

4.4 A general framework

Our framework is quite general. As evoked previously, it includes $S-3$ inference as a specific case. As a consequence, it also includes the inference relation \vdash_{LP} of the three-valued logic LP ⁸ [Priest, 1989] with $\{1, \frac{1}{2}\}$ as designated truth values, defined by:

Definition 4.6

- An *LP-interpretation* I over PS is a mapping from PS to $THREE = \{0, \frac{1}{2}, 1\}$.
- The semantics of a formula in an *LP-interpretation* I is defined inductively by:
 - $I(\text{true}) = 1$ and $I(\text{false}) = 0$;
 - $I(\neg\phi) = 1 - I(\phi)$;
 - $I(\phi \wedge \psi) = \min(I(\phi), I(\psi))$;
 - $I(\phi \vee \psi) = \max(I(\phi), I(\psi))$.
- I is an *LP-model* of Σ iff $I(\Sigma) > 0$.
- $\Sigma \vdash_{LP} \gamma$ holds iff every *LP-model* of Σ is an (*LP-*)*model* of γ .

Indeed, it is not very difficult to prove that \vdash_{LP} coincides with \models_{\emptyset}^3 . More interestingly, let \leq be the preference ordering over *LP-interpretations* defined by $I \leq J$ iff $\{x \in PS \mid I(x) = \frac{1}{2}\} \subseteq \{x \in PS \mid J(x) = \frac{1}{2}\}$; then, the relation \vdash_{LP_m} defined by $\Sigma \vdash_{LP_m} \gamma$ iff every minimal *LP-model* of Σ w.r.t. \leq is an (*LP-*)*model* of γ , can be recovered as a specific case of our $\approx_S^{\mathcal{I}\mathcal{P}, S_0}$ relation:

Proposition 4.3 Let Σ and γ be two formulas from $PROP_{PS}$. We have $\Sigma \vdash_{LP_m} \gamma$ iff $\{\Sigma\} \approx_{PS}^{\mathcal{I}\mathcal{P}, \emptyset} \gamma$.

Our work is also closely related to the approach to paraconsistent reasoning given in [Besnard and Schaub, 1998]. In this work, every formula Σ of $PROP_{PS}$ is associated to a default theory $\langle \Sigma^\pm, D_{PS} \rangle$ where:

⁸ LP is also known as J_3 , see [Epstein, 1990].

- Σ^\pm is a formula in the language $PROP_{PS^\pm}$ where $PS^\pm = \{x^+ \mid x \in PS\} \cup \{x^- \mid x \in PS\}$; Σ^\pm is obtained by replacing in Σ every occurrence of a positive literal x by the positive literal x^+ and every occurrence of a negative literal $\neg x$ by the positive literal x^- .
- $D_{PS} = \{\delta_x \mid x \in PS\}$ is a set of default rules

$$\delta_x = \frac{x^+ \Leftrightarrow \neg x^-}{(x \Leftrightarrow x^+) \wedge (\neg x \Leftrightarrow x^-)}$$

Among the inference relations that can be defined from $\langle \Sigma^\pm, D_{PS} \rangle$ is skeptical signed inference; a formula γ of $PROP_{PS}$ is a skeptical signed consequence of Σ , noted $\Sigma \vdash_s^\pm \gamma$ iff γ^\pm belongs to every extension of $\langle \Sigma^\pm, D_{PS} \rangle$.

Interestingly, the relation \vdash_s^\pm can be recovered as a specific case of our $\approx_S^{\mathcal{P}, S_0}$ relation.

Proposition 4.4 *Let Σ and γ be two formulas from $PROP_{PS}$. We have $\Sigma \vdash_s^\pm \gamma$ iff $\{\Sigma\} \approx_{PS}^{\mathcal{P}, \emptyset} \gamma$.*

Finally, our approach encompasses as well the skeptical inference relations $\vdash^{\mathcal{P}}$ of the standard coherence-based approach⁹ as defined in [Pinkas and Loui, 1992] [Benferhat *et al.*, 1993] [Benferhat *et al.*, 1995].

Proposition 4.5 *Let $B = (\Delta_1, \dots, \Delta_k)$ be a SBB. We associate to B in polynomial time the belief base $B_{new} = \{new_{i,j}, \neg new_{i,j} \vee \phi_{i,j} \mid \phi_{i,j} \in \Delta_i\}$ and the stratification $S = (Var(B), \{new_{1,j} \mid \phi_{1,j} \in \Delta_1\}, \dots, \{new_{k,j} \mid \phi_{k,j} \in \Delta_k\})$ where all the $new_{i,j}$ are new variables (not occurring in B). For every selection policy \mathcal{P} among \mathcal{PO} , \mathcal{LO} , \mathcal{IP} , \mathcal{LE} and every formula γ not containing a new variable, we have $B \vdash^{\mathcal{P}} \gamma$ iff $B_{new} \approx_S^{\mathcal{P}, Var(B)} \gamma$.*

Focusing on variables instead of beliefs gives less syntax-sensitivity and some flexibility that is hardly achieved by the standard coherence-based approach. For instance, let us assume that B gathers beliefs stemming from two different sources of information. The first source states $a \wedge b$ while the second source states $\neg a \wedge \neg b$. The two sources are equally reliable; however, the first source is more reliable to what concerns b while the second source is more reliable to what concerns a . With $\mathcal{P} \in \{\mathcal{IP}, \mathcal{LE}\}$, the standard coherence-based approach gives $(\{a \wedge b, \neg a \wedge \neg b\}) \vdash^{\mathcal{P}} a \vee \neg b$, but the expected conclusion is just the *negation* of $a \vee \neg b$! Indexing variables with sources, we can express in our framework that b from source 1. is more reliable than a from source 1. and the opposite holds when source 2. is considered. Indeed, with $B = \{(\neg a_1 \vee a) \wedge (\neg a \vee a_1) \wedge (\neg a_2 \vee a) \wedge (\neg a \vee a_2) \wedge (\neg b_1 \vee b) \wedge (\neg b \vee b_1) \wedge (\neg b_2 \vee b) \wedge (\neg b \vee b_2) \wedge a_1 \wedge b_1 \wedge \neg a_2 \wedge \neg b_2\}$, $\mathcal{P} \in \{\mathcal{PO}, \mathcal{LO}, \mathcal{IP}, \mathcal{LE}\}$, and $S = (\{a, b\}, \{b_1, a_2\}, \{a_1, b_2\})$, we have $B \approx_S^{\mathcal{P}, S_0} \neg a \wedge b$ and we do **not** have $B \approx_S^{\mathcal{P}, S_0} a \vee \neg b$.

4.5 Dealing with intractability

Despite the generality of our framework, our inference relations are just as hard as the corresponding ones in the standard coherence-based approach when the size of S is not bounded.

⁹A converse polynomial encoding based on \pm transformation exists as well; we omit it here due to space limitations.

\mathcal{P}	CLAUSE / LITERAL $\approx_S^{\mathcal{P}, S_0}$
\mathcal{PO}	$\Delta_2^{\mathcal{P}}[\mathcal{O}(\log n)]$ -complete
\mathcal{LO}	$\Delta_2^{\mathcal{P}}$ -complete
\mathcal{IP}	$\Pi_2^{\mathcal{P}}$ -complete
\mathcal{LE}	$\Delta_2^{\mathcal{P}}$ -complete

Table 1: Complexity of inference.

Definition 4.7 *For every selection policy \mathcal{P} , CLAUSE $\approx_S^{\mathcal{P}, S_0}$ is the following decision problem:*

- **Input:** A belief base B , a subset S of PS , a subset S_0 of S s.t. $B \not\models_{S_0}^3$ false and a CNF formula γ .
- **Query:** Does $B \approx_S^{\mathcal{P}, S_0} \gamma$ hold?

LITERAL $\approx_S^{\mathcal{P}, S_0}$ is the restriction of this problem to the case where γ is a conjunction of literals.

Proposition 4.6 *The complexity of CLAUSE $\approx_S^{\mathcal{P}, S_0}$ and LITERAL $\approx_S^{\mathcal{P}, S_0}$ is reported in Table 1.*

Interestingly, limiting the size of S is sufficient to ensure tractability. The next proposition shows that our second objective is reached in this case:

Proposition 4.7 *The complexity of CLAUSE $\approx_S^{\mathcal{P}, S_0}$ from a belief base B given a stratification $S = (S_0, \dots, S_j)$ of a bounded set of propositional variables is in \mathbf{P} .*

The main reason is that when $|S| \leq b$, there is only 2^b subsets of S , which is a constant when b is a constant. Accordingly, the naive algorithm consisting in generating $\mathcal{S}_{\mathcal{P}}(B, S, S_0)$ by filtering out the preferred consistent elements of 2^S , and testing for every of the resulting elements S' whether or not $B \models_{S'} \gamma$ holds runs in time $\mathcal{O}(2^{2b} \cdot |B| \cdot |\gamma|)$, hence in $\mathcal{O}(|B| \cdot |\gamma|)$ for a constant b , for every B and a CNF query γ .

Of course, this is a “trick” in some sense but we can hardly do better: unless $\mathbf{P} = \mathbf{NP}$, there is no tractable inference relation that is general enough to include \models as a specific case. In addition, we argue that focusing on $B \models_S^3$ as a tractable approximation of $B \models$ in order to design a paraconsistent inference relation is not so bad, especially when compared with other possible choices. Indeed, let $B \vdash$ be a tractable approximation by below of $B \models$ upon which we want to build a tractable paraconsistent inference relation. Requiring that such a tractable paraconsistent relation coincides with $B \vdash$ whenever $B \vdash$ is not explosive and is obtained by weakening $B \vdash$ through the removal of some beliefs of B otherwise limits the spectrum of interesting relations \vdash . For instance, considering the restriction $B \vdash_{unit}$ of $B \vdash$ where only unit-refutation derivable consequences of B are considered would lead to a complex semantics with respect to which \wedge and \vee do not behave classically [Crawford and Etherington, 1998]; considering the restriction $B \vdash_{Horn}$ of $B \vdash$ where B_{Horn} is the subset of Horn clauses of B would not lead to a tractable relation [Cayrol *et al.*, 1998] [Nebel, 1998].

5 Other related work

A close look at the AI literature shows that both trivialization and intractability can be handled by considering *infraclassical* inference relations, i.e., approximations by below of $B \models$.

What is quite surprising is the fact that the two issues have usually **not** been addressed together.

On the one hand, the proof theory of any paraconsistent logic limits the set of admissible proofs to a strict subset (i.e., an approximation by below) of the classical proofs in the objective of avoiding trivialization. In the standard coherence-based approach, the focus is laid on some preferred consistent subbases of B , which can be considered as approximations of B by below. Preferred subbases can also be defined by splitting the belief base B into several micro-theories characterized by their sets of variables, and selected in a dynamic way by looking at the variables occurring in the queries [Chopra and Parikh, 1999]. Unfortunately, for all these approaches, the corresponding inference relations are typically at least as hard as classical inference in the general case.

On the other hand, many approximation techniques have been developed so far to deal with the intractability issue (e.g., [Levesque, 1984] [Crawford and Kuipers, 1989] [Schaerf and Cadoli, 1995] [Selman and Kautz, 1996] [Dalal, 1996] [Crawford and Etherington, 1998]); in such approaches, the inference relation or the knowledge base itself are approximated into computationally easier one(s). However, the trivialization issue is typically not addressed. For instance, the inference relation defined in [Selman and Kautz, 1996] is always explosive when B is inconsistent, while the ones given in [Schaerf and Cadoli, 1995] [Crawford and Etherington, 1998] are explosive in the general case when B is inconsistent.

6 Conclusion

The main contribution of this paper is a family of resource-bounded paraconsistent inference relations. In contrast to many approaches to inconsistency handling, the computational issue is very central in our framework. Especially, the computational resources can be tuned to give rise to realistic, computationally viable, inference relations. Nevertheless, our framework is general enough to encompass $S-3$ logic, the standard coherence-based approach to inconsistency handling, and some other systems for paraconsistent reasoning as specific cases.

Several additional families of inference relations could be defined in our framework. Analyzing their computational and logical properties is a topic for further research.

Acknowledgements

Many thanks to the reviewers for their helpful comments, and to the Région Nord / Pas-de-Calais and the IUT de Lens for their support. Nadège Porquet is granted by the Région Nord / Pas-de-Calais.

References

[Benferhat *et al.*, 1993] S. Benferhat, C. Cayrol, D. Dubois, J. Lang and H. Prade. Inconsistency management and prioritized syntax-based entailment. *Proc. IJCAI'93*, 640-645, 1993.

[Benferhat *et al.*, 1995] S. Benferhat, D. Dubois, and H. Prade. How to infer from inconsistent beliefs without revising? *Proc. IJCAI'95*, 1449-1455, 1995.

[Benferhat, 2000] S. Benferhat. Computing specificity in default reasoning. In *Handbook of DRUMS*, vol. 5, Kluwer Academic, 147-177, 2000.

[Besnard and Schaub, 1998] Ph. Besnard and T. Schaub. Signed systems for paraconsistent reasoning. *Journal of Automated Reasoning*, 20:191-213, 1998.

[Cadoli and Schaerf, 1996] M. Cadoli and M. Schaerf. On the complexity of entailment in propositional multivalued logics. *Annals of Mathematics and Artificial Intelligence*, 18:29-50, 1996.

[Cayrol *et al.*, 1998] C. Cayrol, M.-C. Lagasquie-Schiex and Th. Schiex. Nonmonotonic reasoning: from complexity to algorithms. *Annals of Mathematics and Artificial Intelligence*, 22(3-4):207-236, 1998.

[Chopra and Parikh, 1999] S. Chopra and R. Parikh. An inconsistency tolerant model for belief representation and belief revision. *Proc. IJCAI'99*, 192-197, 1999.

[Crawford and Etherington, 1998] J.M. Crawford and D.W. Etherington. A non-deterministic semantics for tractable inference. *Proc. AAAI'98*, 1998.

[Crawford and Kuipers, 1989] J.M. Crawford and B. Kuipers. Towards a theory of access-limited logic for knowledge representation. *Proc. KR'89*, 67-78, 1989.

[da Costa, 1974] N. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:497-510, 1974.

[Dalal, 1996] M. Dalal. Anytime families of tractable propositional reasoners. *Proc. AI&Math'96*, 42-45, 1996.

[Epstein, 1990] R.L. Epstein. The semantic foundation of logic. Vol. I: propositional logics. *Kluwer Academic Publisher*, 1990.

[Levesque, 1984] H.J. Levesque. A logic of implicit and explicit belief. *Proc. AAAI'84*, 198-202, 1984.

[Lin, 1987] F. Lin. Reasoning in the presence of inconsistency. *Proc. AAAI'87*, 139-143, 1987.

[Nebel, 1998] B. Nebel. How hard is it to revise a belief base? In *Handbook of DRUMS*, vol. 3, Kluwer Academic, 77-145, 1998.

[Papadimitriou, 1994] Ch. H. Papadimitriou. *Computational Complexity*, Addison-Wesley, 1994.

[Pinkas and Loui, 1992] G. Pinkas and R.P. Loui. Reasoning from inconsistency: a taxonomy of principles for resolving conflict. *Proc. KR'92*, 709-719, 1992.

[Priest, 1989] G. Priest. Reasoning about truth. *Artificial Intelligence* 39:231-244, 1989.

[Priest, 1991] G. Priest. Minimally inconsistent LP. *Studia Logica* 50:321-331, 1991.

[Schaerf and Cadoli, 1995] M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence* 74(2):249-310, 1995.

[Selman and Kautz, 1996] B. Selman and H.A. Kautz. Knowledge compilation and theory approximation. *Journal of the Association for Computing Machinery* 43(2):193-224, 1996.

Weakening Conflicting Information for Iterated Revision and Knowledge Integration

Salem Benferhat and Souhila Kaci I.R.I.T.–C.N.R.S. Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex 4, France
{benferhat,kaci}@irit.fr

Daniel Le Berre and Mary-Anne Williams Business & Technology Research Laboratory
The University of Newcastle
Newcastle, NSW 2308, Australia
{daniel,maryanne}@cafe.newcastle.edu.au

Abstract

The ability to handle exceptions, to perform iterated belief revision and to integrate information from multiple sources are essential skills for an intelligent agent. These important skills are related in the sense that they all rely on resolving inconsistent information. We develop a novel and useful strategy for conflict resolution, and compare and contrast it with existing strategies. Ideally the process of conflict resolution should conform with the principle of Minimal Change and should result in the *minimal loss* of information. Our approach to minimizing the loss of information is to weaken information involved in conflicts rather than completely removing it. We implemented and tested the relative performance of our new strategy in three different ways. We show that it retains more information than the existing Maxi-Adjustment strategy at no extra computational cost. Surprisingly, we are able to demonstrate that it provides a computationally effective compilation of the lexicographical strategy, a strategy which is known to have desirable theoretical properties.

1 Introduction

Information modeling and management is a fundamental activity of intelligent systems. Intelligent systems require robust and sophisticated information management capabilities such as exception handling, iterated revision and the integration of information. In this paper we develop a novel and useful strategy for conflict resolution which can be applied to exception handling, iterated revision, and information integration. Throughout we assume that the available information is given as ordered knowledge bases, i.e. a ranking of information as logical sentences. Solving conflicts in our context means computing a consistent knowledge base. One well known system that can deal with conflicts in knowledge bases is the so-called *Adjustment* procedure [Williams, 1994]. In essence, Adjustment propagates as many highly ranked formulas as possible, and ignores information at and below the highest rank where an inconsistency is found. The main advantage of this system is its computational efficiency. For example, it only needs at most $\text{Log}_2 n$ calls to a SAT solver

to build the consistent knowledge base where n is the number of ranks in the knowledge base. The obvious disadvantage of Adjustment, however, is that it can remove more formulae than is necessary to restore the consistency of the knowledge base if the independence of information is not made explicit. In order to overcome this shortcoming another strategy called *Maxi-Adjustment* was introduced [Williams, 1996] and implemented [Williams and Sims, 2000]. Maxi-Adjustment has proved to be a useful strategy for real world applications e.g. software engineering [Williams, 1998], information filtering [Lau *et al.*, 2000] and intelligent payment systems [Wong and Lau, 2000]. The main idea of Maxi-Adjustment is to solve conflicts at each rank of priority in the knowledge base. This is done, incrementally, starting from the information with highest rank. When inconsistency is encountered in the knowledge base, then all formulas in the rank responsible for the conflicts are removed. The other formulas are kept, and the process continues to the next rank.

Clearly Maxi-Adjustment keeps more information than Adjustment, since it does not stop at the first rank where inconsistency is met. Even though Maxi-Adjustment propagates more information than Adjustment, one can still argue that Maxi-Adjustment removes too much information because it adopts a sceptical approach to the way it removes the conflict sets at each rank.

The purpose of this paper is to describe a significant improvement to Maxi-Adjustment. We call this system *Disjunctive Maxi-Adjustment*, and denote it by DMA. The idea is similar to Maxi-Adjustment, except that information is weakened instead of being removed when conflicts are detected. So instead of removing all formulas involved in conflicts, as it is done in Maxi-Adjustment, DMA takes their disjunctions pairwise. If the result is consistent, then we move to the next rank. If the result is still inconsistent, then we replace the formulas in conflicts by all possible disjunctions involving 3 formulas in the conflict sets and again if the result is consistent we move to the next layer, and if it is inconsistent we consider disjunctions of size 4, 5, etc. The only case where all formulas responsible for conflicts are removed is when the disjunction of all these formulas is inconsistent with the higher priority information.

This paper focuses on the DMA strategy from the theoretical and experimental perspectives. In particular,

- We show that DMA is equivalent to the well known

lexicographical strategy [Benferhat *et al.*, 1993; Lehmann, 1995]. More precisely, we show that for an inconsistent base K if $\delta_{DMA}(K)$ is the classical base obtained using DMA, and $\delta_{Lex}(K)$ is the set of all lexicographically maximal consistent subbases of K , then:

$$\forall \psi, \delta_{DMA}(K) \vdash \psi \text{ iff } \forall A \in \delta_{Lex}(K), A \vdash \psi.$$

In other words, we obtain the surprising and computationally useful result that DMA provides a “compilation” of lexicographical systems.

- It is well known that computing conflicts is a hard task, and we are able to show that DMA works even if the conflicts are not explicitly computed. For this, we propose an alternative, but equivalent, approach to DMA called whole-DMA where disjunctions are built on the whole stratum when we meet inconsistency instead of only on the conflicts.
- We also propose another equivalent alternative to DMA called iterative-DMA where instead of considering disjunctions of size (3,4, etc) on the initial set of conflicts, we only compute disjunctions of size 2 but on new sets of conflicts.
- Lastly, we compare these different implementations of DMA experimentally, and contrast their applicability.

2 Ordered information in Spohn’s OCF framework

We consider a finite propositional language denoted by \mathcal{L} . Let Ω be the set of interpretations. \vdash denotes the classical consequence relation, Greek letters ϕ, ψ, \dots represent formulas. We use Spohn’s ordinal conditional function [Spohn, 1988] framework, which is also known as the Kappa function framework.

At the semantic level, the basic notion of Spohn’s ordinal conditional function framework is a distribution called an OCF, denoted by κ , which is a mapping from Ω to \mathcal{N} , such that $\exists \omega, \kappa(\omega) = 0$. \mathcal{N} is the set of natural numbers. $\kappa(\omega)$ can be viewed as the degree of impossibility of ω .

By convention, $\kappa(\omega) = 0$ means that nothing prevents ω from being the real world, and $\kappa(\omega) = +\infty$ means that ω is certainly not the real world¹. The lower $\kappa(\omega)$ is, the more expected it is, i.e. if $\kappa(\omega) < \kappa(\omega')$ then ω is said to be more plausible than ω' .

In practice, OCF distributions over all possible worlds are not available, however a ranked knowledge base provides a compact representation of an OCF distribution [Williams, 1994]. Since we will be working with ranked knowledge bases throughout, we define a knowledge base to be ranked. In particular, a knowledge base is a set of weighted formulas of the form $K = \{(\phi_i, k_i) : i = 1, \dots, n\}$ where ϕ_i is a classical formula and k_i is a positive number representing the level of priority of ϕ_i . The higher k_i , the more important the formula ϕ_i .

Given K , we can generate a unique OCF distribution, denoted by κ_K , such that all the interpretations satisfying all

¹Note that the notion of impossible worlds ($+\infty$) does not exist in original works of Spohn.

the formulae in K will have the lowest value, namely 0, and the other interpretations will be ranked with respect to the highest formulae that they falsify. Namely:

Definition 1 $\forall \omega \in \Omega$,

$$\kappa_K(\omega) = \begin{cases} 0 & \forall (\phi_i, k_i) \in K, \omega \models \phi_i \\ \max\{k_i : (\phi_i, k_i) \in K \text{ and } \omega \not\models \phi_i\} & \text{otherwise.} \end{cases}$$

Then, given κ_K associated with a knowledge base K , the models of K are the interpretations ω s.t. $\kappa_K(\omega) = 0$.

3 Adjustment and Maxi-Adjustment

3.1 Stratified vs ranked knowledge base

We have seen that ranked information is represented by means of knowledge bases of the form $K = \{(\phi_i, k_i) : i = 1, \dots, n\}$. We sometimes also represent this base K in a stratified form as follows: $K = \{S_1, \dots, S_n\}$ where S_i ($i = 1, \dots, n$) contains classical formulas of K having the same rank and which are more reliable than formulas of S_j for $j > i$. So the lower the stratum, the higher the rank.

In this representation, subbases are also stratified. That is, if A is a subbase of $K = \{S_1, \dots, S_n\}$, then $A = \{A_1, \dots, A_n\}$ such that $A_j \subseteq S_j$, $j = 1, \dots, n$. (A_j may be empty).

Conversely, we can represent a stratified base $K = \{S_1, \dots, S_n\}$ using a weighted knowledge base by associating formulas of each strata S_i to the same rank k_i . These ranks should be such that $k_1 > \dots > k_n$.

Let us now introduce the notion of conflicts and kernel which will prove useful in the subsequent discussion:

Definition 2 Let $K = \{S_1, \dots, S_n\}$ be a stratified base. A conflict in K , denoted by C , is a subbase of K such that:

- $C \vdash \perp$ (inconsistency),
- $\forall \phi, \phi \in C, C - \{\phi\} \not\vdash \perp$ (minimality).

Definition 3 Let \mathcal{C} be the set of all possible conflicts in K . We define the kernel of K , denoted by $\text{kernel}(K)$, as the set of formulas of K which are involved in at least one conflict in \mathcal{C} i.e., $\text{kernel}(K)$ is the union of all conflicts in K .

Formulas in K which are not involved in any conflict in K are called *free* formulas.

3.2 The problem

Our aim in this paper is to address the problem of identifying conflicts for the purposes of drawing plausible inferences from inconsistent knowledge bases, iterated revision and information integration. Our technique for resolving conflicts can be used: (i) to build a transmutation for iterated belief revision [Williams, 1994] where the new information can be incorporated into any rank, and (ii) for theory extraction [Williams and Sims, 2000] which provides a natural and puissant mechanism for merging conflicting information. Without loss of generality we focus on a particular case of revision where some new information φ is added to some ranked knowledge base K . Namely,

given a knowledge base K , and a new formula φ we compute $\delta(K \cup \{(\varphi, +\infty)\})$, the classical (not stratified) consistent subbase of $K \cup \{(\varphi, +\infty)\}$. Then, ψ is said to be a plausible consequence of $K \cup \{(\varphi, +\infty)\}$ iff $\delta(K \cup \{(\varphi, +\infty)\}) \vdash \psi$. In the rest of this paper we simply write K_φ instead of $K \cup \{(\varphi, +\infty)\}$. In a stratified form we write $\{S_0, S_1, \dots, S_n\}$ where $S_0 = \{\varphi\}$. We briefly recall two important methods to compute $\delta(K \cup \{(\varphi, +\infty)\})$: *Adjustment* and *Maxi-Adjustment*. We will illustrate them using a simple example. We point the reader to [Williams, 1994; 1996] for more details.

3.3 Adjustment

From a syntactical point of view, the idea of Adjustment is to start with formulas having the highest rank in K_φ and to add as many prioritized formulas as possible while maintaining consistency. We stop at the highest rank (or the lowest stratum) where we meet inconsistency called the inconsistency rank of K_φ , denoted by $Inc(K_\varphi)$.

Note that a more efficient binary search based algorithm which only needs $\log_2 n$ consistency checks has been developed and implemented² [Williams and Sims, 2000]. The selected base will be denoted by $\delta_A(K_\varphi)$. Note that the process of selecting the consistent base using the Adjustment for new pieces of information placed in the highest rank is identical to that used in possibilistic logic [Dubois *et al.*, 1994].

One can easily see that this is not a completely satisfactory way to deal with the inconsistency since formulas with rank lower than $Inc(K_\varphi)$ are ignored even if they are consistent with the selected base.

A formula ψ is said to be an Adjustment consequence of K_φ , denoted by $K_\varphi \vdash_A \psi$, if $\delta_A(K_\varphi) \vdash \psi$. One important property of Adjustment is that it is semantically well defined. More precisely, we have the following soundness and completeness result: $K_\varphi \vdash_A \psi$ iff $\forall \omega \in Pref(\kappa_{K_\varphi}), \omega \models \psi$, where $Pref(\kappa_{K_\varphi})$ is the set of interpretations which satisfy φ and have minimal rank in the OCF distribution κ_{K_φ} given by Definition 1.

Example 1 Let $K = \{S_1, S_2, S_3\}$ be such that $S_1 = \{(-a \vee -b \vee c, 3), (-d \vee c, 3), (-e \vee c, 3)\}$, $S_2 = \{(d, 2), (e, 2), (f, 2), (\neg f \vee \neg g \vee c, 2)\}$ and $S_3 = \{(a, 1), (b, 1), (g, 1), (h, 1)\}$. Let $\varphi = \neg c$. First, we have $\delta_A(K_{\neg c}) = \{\neg c\}$.

There is no conflict in $\delta_A(K_{\neg c}) \cup S_1$ then $\delta_A(K_{\neg c}) \leftarrow \{\neg c, -a \vee -b \vee c, -d \vee c, -e \vee c\}$. Now, S_2 contradicts $\delta_A(K_{\neg c})$ due to the conflicts $\{d, -d \vee c, -c\}$ and $\{e, -e \vee c, -c\}$. Then, we do not add the stratum S_2 and the computation of $\delta_A(K_{\neg c})$ is achieved, and we get $\delta_A(K_{\neg c}) = \{\neg c, -a \vee -b \vee c, -d \vee c, -e \vee c\}$. Note that $\delta_A(K_{\neg c}) \not\vdash h$, even if h is not involved in any conflict in $K_{\neg c}$.

3.4 Maxi-Adjustment

Maxi-Adjustment [Williams, 1996] was developed to address the problem of discarding too much information for applications like software engineering [Williams, 1998] and information filtering [Lau *et al.*, 2000].

²<http://cafe.newcastle.edu.au/systems/saten.html>

The idea in Maxi-Adjustment also involves selecting one consistent subbase from K denoted by $\delta_{MA}(K_\varphi)$. The difference is that it does not stop at the first rank where it meets inconsistency. Moreover, conflicts are solved rank by rank. We start from the first rank and take the formulas of S_1 which do not belong to any conflict in $\{\varphi\} \cup S_1$. Let S'_1 be the set of these formulas. Then, we move to the next rank and add all formulas which are not involved in any conflict in $S'_1 \cup S_2$, and so on. It is clear that Maxi-Adjustment keeps more formulas than the Adjustment.

Example 1 (using Maxi-Adjustment)

First, we have $\delta_{MA}(K_{\neg c}) = \{\neg c\}$. There is no conflict in $\delta_{MA}(K_{\neg c}) \cup S_1$ then $\delta_{MA}(K_{\neg c}) \leftarrow \{\neg c, -a \vee -b \vee c, -d \vee c, -e \vee c\}$. Now, S_2 contradicts $\delta_{MA}(K_{\neg c})$ due to the conflicts $\{d, -d \vee c, -c\}$ and $\{e, -e \vee c, -c\}$. Then, we do not add the clauses from S_2 involved in these conflicts: $\delta_{MA}(K_{\neg c}) \leftarrow \delta_{MA}(K_{\neg c}) \cup \{f, \neg f \vee \neg g \vee c\}$. Now, S_3 contradicts $\delta_{MA}(K_{\neg c})$ due to the conflicts $\{a, b, -a \vee -b \vee c, -c\}$ and $\{f, g, \neg f \vee \neg g \vee c, -c\}$. Since all the clauses, except h , from the stratum S_3 are involved in one conflict, we only add h to $\delta_{MA}(K_{\neg c})$. Finally, we get: $\delta_{MA}(K_{\neg c}) = \{\neg c, -a \vee -b \vee c, -d \vee c, -e \vee c, f, \neg f \vee \neg g \vee c, h\}$. Note that $\delta_{MA}(K_{\neg c}) \vdash h$.

4 Disjunctive Maxi-Adjustment

Although Maxi-Adjustment retains more information than Adjustment, it can still be argued that it is too cavalier in the way it solves the conflicts.

In this section, we propose a new strategy which is a significant improvement of Maxi-Adjustment. The computation of the consistent base is essentially the same as in Maxi-Adjustment, the only difference is when we meet an inconsistency at some rank, instead of removing all the formulas involved in the conflicts at this rank we weaken them, by replacing them by their pairwise disjunctions. If the result is consistent then we move to the next rank, else we replace these formulas by their possible disjunctions of size 3. If the result is consistent then we move to the next rank, else we add the disjunctions of size 4 of these formulas, and so on. We summarize this process in Algorithm 1:

Notation: $d_k(C)$ is the set of all possible disjunctions of size k between formulas of C . If $k > |C|$ then $d_k(C) = \emptyset$.

Example 1 (using DMA)

First, we have $KB = \{\neg c\}$. There is no conflict in $KB \cup S_1$ then $KB \leftarrow \{\neg c, -a \vee -b \vee c, -d \vee c, -e \vee c\}$. Now, S_2 contradicts KB due to the conflicts $\{d, -d \vee c, -c\}$ and $\{e, -e \vee c, -c\}$. We do not add the clauses from S_2 involved in these conflicts: $KB \leftarrow KB \cup \{f, \neg f \vee \neg g \vee c\}$. Now we create all the possible disjunctions of size 2 with $C = \{d, e\}$: $d_2(C) = \{d \vee e\}$. Since $KB \cup d_2(C)$ is inconsistent, and we cannot create larger disjunctions, we do not add anything from S_2 to KB .

Algorithm 1: DMA(K, φ)Data: a stratified knowledge base $K = \{S_1, \dots, S_n\}$;a new sure formula: φ ;Result: a consistent subbase $\delta_{DMA}(K_\varphi)$

```

begin
   $KB \leftarrow \{\varphi\}$ ;
  for  $i \leftarrow 1$  to  $n$  do
    if  $KB \cup S_i$  is consistent then  $KB \leftarrow KB \cup S_i$ 
    else
      Let  $C$  be the subset of  $S_i$  in kernel of  $KB \cup S_i$ ;
       $KB \leftarrow KB \cup \{\phi : \phi \in S_i \text{ and } \phi \notin C\}$ ;
       $k \leftarrow 2$ ;
      while  $k \leq |C|$  and  $KB \cup d_k(C)$  is inconsistent
      do
         $k \leftarrow k + 1$ ;
      if  $k \leq |C|$  then  $KB \leftarrow KB \cup d_k(C)$ ;
  return  $KB$ 
end

```

Please note at this rank, we do not add more information than Maxi-Adjustment.

Now, S_3 contradicts KB due to the conflicts $\{a, b, \neg a \vee \neg b \vee c, \neg c\}$ and $\{f, g, \neg f \vee \neg g \vee c, \neg c\}$. h is not involved in any conflict. Then, $KB \leftarrow KB \cup \{h\}$.

We now create all the possible pairwise disjunctions with $C = \{a, b, g\}$: $d_2(C) = \{a \vee b, a \vee g, b \vee g\}$. Since $KB \cup d_2(C)$ is inconsistent, we create $d_3(C) = \{a \vee b \vee g\}$. Since $KB \cup d_3(C)$ is consistent, we add $d_3(C)$ to KB and the algorithm stops.

Then $\delta_{DMA}(K_\varphi) = \{\neg c, \neg a \vee \neg b \vee c, \neg d \vee c, \neg e \vee c, f, \neg f \vee \neg g \vee c, h, a \vee b \vee g\}$

which is equivalent to $\{\neg c, \neg a \vee \neg b, \neg d, \neg e, f, \neg g, h, a \vee b\}$. DMA keeps more information from the last stratum than Maxi-Adjustment does.

Definition 4 A formula ψ is said to be a DMA consequence of K and φ , denoted by $K_\varphi \vdash_{DMA} \psi$, if it is inferred from $\delta_{DMA}(K_\varphi)$. Namely, $K_\varphi \vdash_{DMA} \psi$ iff $\delta_{DMA}(K_\varphi) \vdash \psi$.

5 Two other implementations of DMA

In the previous section we have shown a way to compute $\delta_{DMA}(K_\varphi)$ using the computation of the kernel. In this section, we propose two alternative ways to compute $\delta_{DMA}(K_\varphi)$. The first approach, called whole-DMA(K, φ), does not compute the kernel. The main motivation for this alternative is that computing the kernel is in general hard. For the second approach, called iterative-DMA(K, φ), when inconsistency is (again) met after weakening the kernel, then rather than weakening the original kernel by considering its disjunctions of size 3, we only weaken the newly computed kernel obtained by considering disjunctions of size 2. The motivation of this approach is to reduce the size of added (disjunctions) formulas.

5.1 Whole Disjunctive Maxi-Adjustment

We propose a slightly modified version of the DMA algorithm. The idea is that when $KB \cup S_i$ is inconsistent, instead

of considering all possible disjunctions of size j of elements of S_i which are in $\text{kernel}(KB \cup S_i)$, we consider all possible disjunctions of size j of S_i without computing a kernel. This is justified by the following proposition:

Proposition 1 Let $KB \cup S$ be inconsistent. Let C be the subset of S in $\text{kernel}(KB \cup S)$, and $F = S - C$ be the set of remaining formulas. Let $d_j(C)$ (resp. $d_j(S)$) be the set of all possible disjunctions of size j from C (resp. S). Then, $KB \cup d_j(C) \cup F \equiv KB \cup d_j(S)$.

Proof (sketch)

Let us assume that $d_{j-1}(S) \cup KB$ is inconsistent, and show that $d_j(S) \cup KB \equiv d_j(C) \cup KB \cup F$.

It is clear that $d_j(C) \subseteq d_j(S)$. Hence it is enough to show that $KB \cup d_j(S) \vdash F$.

Let A be a conflict of $KB \cup d_{j-1}(S)$, and $\{\psi_1, \dots, \psi_n\}$ be a subset of A in $d_{j-1}(S)$. Let $\varphi \in F$.

Then $\{\varphi \vee \psi_1, \dots, \varphi \vee \psi_n\} \subseteq d_j(S)$, with $\psi_i \neq \varphi$ since $\varphi \notin A$ (because φ is free).

Now since $KB \cup A$ is inconsistent then $KB \vdash \neg\psi_1 \vee \dots \vee \neg\psi_n$. Applying successive resolutions between $\{\varphi \vee \psi_1, \dots, \varphi \vee \psi_n\}$ and $\neg\psi_1 \vee \dots \vee \neg\psi_n$ leads to entail φ .

Hence there is no need to consider disjunctions containing free formulas since they will be subsumed. \square

With the help of this proposition, the “else” block in the DMA algorithm is replaced by

```

else
   $k \leftarrow 2$ 
  while  $KB \cup d_k(S_i)$  is inconsistent and  $k \leq |S_i|$  do
     $k \leftarrow k + 1$ 
  if  $k \leq |S_i|$  then  $KB \leftarrow KB \cup d_k(S_i)$ 
to obtain the whole DMA algorithm.

```

Example 1 (using whole DMA)

First, we have $KB = \{\neg c\}$.

S_1 is consistent with KB . Then, $KB \leftarrow KB \cup S_1$.

Now, S_2 contradicts KB . We compute all possible pairwise disjunctions with S_2 . $d_2(S_2) = \{d \vee e, d \vee f, d \vee \neg f \vee \neg g \vee c, e \vee f, e \vee \neg f \vee \neg g \vee c\}$.

Since, $KB \cup S_2$ is inconsistent, we compute all possible disjunctions of size 3 between formulas of S_2 . We get $d_3(S_2) = \{d \vee e \vee f, d \vee e \vee \neg f \vee \neg g \vee c\}$ which is consistent with KB . Then, $KB \leftarrow KB \cup d_3(S_2)$.

Now, S_3 is inconsistent with KB . We compute all possible pairwise disjunctions with S_3 . $d_2(S_3) = \{a \vee b, a \vee g, a \vee h, b \vee g, b \vee h, g \vee h\}$ which is still inconsistent with KB . We have $d_3(S_3) = \{a \vee b \vee g, a \vee b \vee h, b \vee g \vee h, a \vee g \vee h\}$ which is consistent with KB , then $KB \leftarrow KB \cup d_3(S_3)$.

Hence, $\delta_{WDMA}(K_{\neg c}) = \{\neg c, \neg a \vee \neg b \vee c, \neg d \vee c, \neg e \vee c, d \vee e \vee f, d \vee e \vee \neg f \vee \neg g \vee c, a \vee b \vee g, a \vee b \vee h, b \vee g \vee h, a \vee g \vee h\}$ which is equivalent to $\{\neg c, \neg a \vee \neg b, \neg d, \neg e, f, \neg g, a \vee b, h\}$. Then, it is equivalent to $\delta_{DMA}(K_{\neg c})$.

5.2 Iterative Disjunctive Maxi-Adjustment

The idea of this alternative implementation of DMA is as follows: let S_i be inconsistent with KB . Let C and F be the kernel and the remaining formulas of S_i .

Now assume that $KB \cup F \cup d_2(C)$ is still inconsistent. Then

rather than weakening C again by considering disjunctions of size 3, we only weaken those formulas in $d_2(C)$ which are still responsible for conflicts. Namely, we split $d_2(C)$ into C' and F' which respectively represent the kernel and remaining formulas of $d_2(C)$. Then instead of taking $KB \cup F \cup d_3(C)$ as in DMA, we take $KB \cup F \cup F' \cup d_2(C')$. The algorithm becomes:

Algorithm 2: IDMA(K, φ)

Data: a stratified knowledge base $K = \{S_1, \dots, S_n\}$;
a new sure formula: φ

Result: a consistent subbase $\delta_{IDMA}(K_\varphi)$

```

begin
   $KB \leftarrow \{\varphi\}, i \leftarrow 1;$ 
  while  $i \leq n$  do
    if  $KB \cup S_i$  is consistent then
       $KB \leftarrow KB \cup S_i; i \leftarrow i + 1;$ 
    else
      Let  $C \subseteq S_i$  be in  $\text{kernel}(KB \cup S_i)$ ;
       $S_i \leftarrow \{\phi : \phi \in S_i \text{ and } \phi \notin C\}$ ;
      if  $|C| = 1$  then  $i \leftarrow i + 1$  else  $S_i \leftarrow S_i \cup d_2(C)$ ;
  return  $KB$ 
end

```

Proposition 2 Let $KB \cup F \cup d_i(C)$ be inconsistent. Then,
 $KB \cup F \cup d_i(C) \equiv KB \cup F \cup F' \cup d_2(C')$,
where F' and C' are kernels from $d_i(C)$.

The proof is a corollary of Prop. 1 and the following lemma:

Lemma 1 Let A be a set of formulas. Let $B = d_i(A)$ and $C = d_{i+1}(A)$ be the set of all possible disjunctions of A of size i and $i + 1$ respectively. Then, $C = d_2(B)$.

This lemma means that taking all disjunctions of size i , then reconsidering all disjunctions of size 2 again on the result is the same as considering all disjunctions of size $i + 1$.

Example 1 (using iterative DMA)

First, we have $KB = \{\neg c\}$.

There is no conflict in $KB \cup S_1$. Then, $KB \leftarrow KB \cup S_1$. S_2 is inconsistent with KB due to the conflicts $\{\neg c, \neg d \vee c, d\}$ and $\{\neg c, \neg e \vee c, e\}$. We add $\{f, \neg f \vee \neg g \vee c\}$ to KB . The disjunction $d \vee e$ is still inconsistent with KB , then we move to S_3 .

S_3 contradicts KB due to the conflicts $\{a, b, \neg a \vee \neg b \vee c, \neg c\}$ and $\{f, g, \neg f \vee \neg g \vee c, \neg c\}$. h is not involved in any conflict. Then, $KB \leftarrow KB \cup \{h\}$.

We now create all the possible pairwise disjunctions with $C = \{a, b, g\}$: $d_2(C) = \{a \vee b, a \vee g, b \vee g\}$.

$KB \cup d_2(C)$ is inconsistent due to the conflict $\{\neg c, \neg a \vee \neg b \vee c, f, \neg f \vee \neg g \vee c, a \vee g, b \vee g\}$. $a \vee b$ in $d_2(C)$ is not involved in the conflict, then $KB \leftarrow KB \cup \{a \vee b\}$.

Now, we take the pairwise disjunctions with $C = \{a \vee g, b \vee g\}$. $d_2(C) = \{a \vee b \vee g\}$. $KB \cup d_2(C)$ is consistent. However, there is no need to add $a \vee b \vee g$ to KB since $a \vee b$ already belongs to KB . Hence, $\delta_{IDMA}(K_{\neg c}) = \{\neg c, \neg a \vee \neg b \vee c, \neg d \vee c, \neg e \vee c, f, \neg f \vee \neg g \vee c, a \vee b, h\}$ which is equivalent to $\{\neg c, \neg a \vee \neg b, \neg d, \neg e, f, \neg g, a \vee b, h\}$. Hence, it is equivalent to $\delta_{DMA}(K_{\neg c})$.

6 DMA: Compilation of lexicographical inferences

The aim of this section is to show that *DMA* is a compilation of the lexicographical system, hence it satisfies the AGM postulates [Alchourrón *et al.*, 1985]. First let us recall the lexicographical inference.

6.1 Lexicographical inference

The lexicographical system [Benferhat *et al.*, 1993; Lehmann, 1995] is a coherence-based approach where an inconsistent knowledge base is replaced by a set of maximally preferred consistent subbases. The preference relation between subbases is defined as follows:

Definition 5 Let $A = \{A_1, \dots, A_n\}$ and $B = \{B_1, \dots, B_n\}$ be two consistent subbases of K . A is said to be lexicographically preferred to B , denoted by $A >_{Lex} B$, iff
 $\exists k$ s.t. $|A_k| > |B_k|$ and $\forall j < k, |A_j| = |B_j|$.

Let $\delta_{Lex}(K_\varphi)$ denotes the set of all lexicographically preferred subbases of K_φ , those which are maximal w.r.t. $>_{Lex}$. Then, the lexicographical inference is defined by:

Definition 6 A formula ψ is said to be a lexicographical consequence of K_φ , denoted by $K_\varphi \vdash_{Lex} \psi$, if it is a classical consequence of all the elements of $\delta_{Lex}(K_\varphi)$, namely
 $\forall A \in \delta_{Lex}(K_\varphi), A \vdash \psi$.

Example 1 (continued)

We have $\delta_{Lex}(K_{\neg c}) = \{A, B\}$ where $A = \{\neg c, \neg a \vee \neg b \vee c, \neg d \vee c, \neg e \vee c, f, \neg f \vee \neg g \vee c, a, h\}$ and $B = \{\neg c, \neg a \vee \neg b \vee c, \neg d \vee c, \neg e \vee c, f, \neg f \vee \neg g \vee c, b, h\}$.

For example, we have

$K_{\neg c} \vdash_{Lex} a \vee b$ since $A \vdash a \vee b$ and $B \vdash a \vee b$.

6.2 Basic steps of the compilation

The aim of this section is to show that *DMA* is equivalent to the lexicographical system. *DMA* offers a clear advantage over the lexicographical system because it obviates the need to explicitly compute $\delta_{Lex}(K_\varphi)$ which may be exponential in size. Formally, we will show the following equivalence:

$$K_\varphi \vdash_{Lex} \psi \Leftrightarrow K_\varphi \vdash_{DMA} \psi \quad (1)$$

Note that $\delta_{DMA}(K_\varphi)$ is a classical consistent base.

Example 1 (continued)

Let us first show that applying the lexicographical system on $K_{\neg c}$ gives the same results as applying *DMA* on $K_{\neg c}$.

Indeed, $K_{\neg c} \vdash_{Lex} \psi$ iff $A \vdash \psi$ and $B \vdash \psi$
iff $A \vee B \vdash \psi$ iff $\{\neg c, \neg a \vee \neg b, \neg d, \neg e, f, \neg g, a \vee b, h\} \vdash \psi$
(after removing subsumed formulas in $A \vee B$)
iff $\delta_{DMA}(K_{\neg c}) \vdash \psi$ iff $K_{\neg c} \vdash_{DMA} \psi$.

To show (1) we follow the following steps:

Step 1: we construct a new base K' from K s.t.

$$K_\varphi \vdash_{Lex} \psi \Leftrightarrow K'_\varphi \vdash_A \psi \quad (2)$$

Namely, applying lexicographical system on K_φ is equivalent to applying Adjustment to K'_φ .

Step 2: in the second step we show that

$$K'_\varphi \vdash_A \psi \Leftrightarrow K_\varphi \vdash_{DMA} \psi \quad (3)$$

Namely, applying Adjustment to K'_φ is equivalent to applying DMA to K_φ .

Step 1: Constructing K'

In order to show the proof of (2), we need to rewrite the lexicographical system at the semantic level, which is immediate:

Definition 7 Let $K = \{S_1, \dots, S_n\}$. Let ω and ω' be two interpretations, and $A_\omega, A_{\omega'}$ be the subbases composed of all formulas of K satisfied by ω and ω' respectively.

Then, ω is said to be lexicographically preferred to ω' w.r.t. K , denoted by $\omega >_{Lex, K} \omega'$, iff $A_\omega >_{Lex} A_{\omega'}$ (using Definition 5).

Proposition 3 Let $\delta_{Lex}(K)$ be the set of lexicographical preferred consistent subbases of K .

Let A_ω be the set of formulas in K satisfied by ω . Then,

- i. If ω is minimal w.r.t. $>_{Lex, K}$ then $A_\omega \in \delta_{Lex}(K)$
- ii. $\forall A \in \delta_{Lex}(K), \exists \omega \models A$ s.t. ω is minimal w.r.t. $>_{Lex, K}$.

Using Prop. 3, at the semantic level, (2) is equivalent to:

$$\kappa_{K'_\varphi}(\omega) < \kappa_{K'_\varphi}(\omega') \text{ iff } \omega >_{Lex, K_\varphi} \omega' \quad (4)$$

where $\kappa_{K'_\varphi}$ is the OCF associated to K'_φ obtained from Definition 1.

Let us now show how to construct K' from K such that it satisfies (4). For this, we use two intuitive ideas.

The first idea is that Adjustment is insensitive to the number of equally reliable formulas falsified while lexicographical system is not (i.e. cardinality of conflict sets). Assume that we have a base $K = \{(\phi, i), (\psi, i)\}$ which contains two formulas with a same rank. Then, the rank (using Def. 1) associated with an interpretation ω falsifying one formula has a same rank as an interpretation falsifying two formulas. However, if we use the lexicographical system, an interpretation falsifying one formula is preferred to an interpretation falsifying two formulas. Now one can check that if we construct a knowledge base $K' = \{(\phi, i), (\psi, i), (\phi \vee \psi, 2i)\}$ from K by adding the disjunction $\phi \vee \psi$ with a higher rank, then equation (4) is satisfied. So the first idea is to add disjunctions with the rank equal to the sum of ranks of formulas composing the disjunctions.

The second idea is related to the notion of compensation. To illustrate this idea, let us now consider $K = \{S_1, S_2\}$ such that $S_1 = \{\phi_1\}$ and $S_2 = \{\phi_2, \phi_3, \phi_4\}$. The intuition behind this example is to show that ranks associated with the formulas should satisfy some constraints in order to recover the lexicographical inference. Indeed, let us for instance associate the rank 2 with ϕ_1 , and the rank 1 with ϕ_2, ϕ_3, ϕ_4 . Let ω and ω' be two interpretations such that $A_\omega = \{\{\phi_1\}, \{\}\}$ and $A_{\omega'} = \{\{\}, \{\phi_2, \phi_3, \phi_4\}\}$. A_ω means that ω satisfies all formulas of S_1 but falsifies all formulas of S_2 . $A_{\omega'}$ means that ω' satisfies all the formulas of S_2 but falsifies all the formulas of S_1 . Following the suggestion of the first idea, let us add all possible disjunctions. We obtain:

$$K' = \{(\phi_1 \vee \phi_2 \vee \phi_3 \vee \phi_4, 5); (\phi_1 \vee \phi_3 \vee \phi_4, 4); (\phi_1 \vee \phi_2 \vee \phi_4, 4); (\phi_1 \vee \phi_2 \vee \phi_3, 4); \{(\phi_1 \vee \phi_2, 3); (\phi_1 \vee \phi_3, 3); (\phi_1 \vee \phi_4, 3); (\phi_2 \vee \phi_3 \vee \phi_4, 3)\}; \{(\phi_1, 2); (\phi_2 \vee \phi_3, 2); (\phi_2 \vee \phi_4, 2); (\phi_3 \vee \phi_4, 2)\}; \{(\phi_2, 1); (\phi_3, 1); (\phi_4, 1)\}\}.$$

We can easily check that $\kappa_{K'}(\omega) = 3$ and $\kappa_{K'}(\omega') = 2$ while $A_\omega >_{Lex, K} A_{\omega'}$. This is due to the fact that the disjunction $\phi_2 \vee \phi_3 \vee \phi_4$ has a rank higher than ϕ_1 . Hence, there is a compensation effect. So, in order to recover the lexicographical order, ϕ_1 must have a rank strictly greater than the rank of $\phi_2 \vee \phi_3 \vee \phi_4$. A way to do this is to significantly differentiate the different ranks associated with strata. For this, we associate to each formula $(\phi_{ij}, k_i) \in S_i$ the rank N^{k_i} where N is very large. N should be s.t. $\forall i, N^{k_i} > \sum_{j>i} N^{k_j}$. Such an N always exists. It means that the rank given to a stratum must be greater than the sum of all the ranks of the less reliable strata.

Following these two ideas, K' is formally constructed as follows:

Let $K = \{S_1, \dots, S_n\}$, and φ a new sure information:

1. We define a new base B :

$$B = \{(\phi_{ij}, N^{k_i}) : i = 1, n \text{ and } \phi_{ij} \in S_i\}.$$
2. $K' = \{(D_j(B), a_j)\}$ where $D_j(B)$ is the set of all possible disjunctions of size j between formulas of B , and a_j is the sum of ranks of formulas in $D_j(B)$.

Then we have:

Proposition 4 $K'_\varphi \vdash_A \psi$ iff $K_\varphi \vdash_{Lex} \psi$.

Step 2: Adjustment on $K'_\varphi \equiv$ DMA on K_φ

The following proposition shows that the base K' constructed in Step 1 allows us to recover the lexicographical system.

Proposition 5 Let $K = \{S_1, \dots, S_n\}$ be a stratified base, and φ be a sure formula. Let K' be a base constructed in Step 1. Then,

$$K'_\varphi \vdash_A \psi \Leftrightarrow K_\varphi \vdash_{DMA} \psi.$$

Due to the lack of space, we skip the proof of Prop. 5 and illustrate its main ideas by an example. The idea is to simplify the computation of $\delta_A(K'_\varphi)$ until recovering $\delta_{DMA}(K_\varphi)$.

Example 2 Let $K = \{S_1, S_2\}$ where $S_1 = \{\neg a \vee \neg b \vee c\}$ and $S_2 = \{a, b, g\}$. Let $\varphi = \neg c$.

First it can be checked that

$$\delta_{DMA}(K_{\neg c}) = \{\neg c, \neg a \vee \neg b \vee c, a \vee b, g\}.$$

Let N be a large number. Using Step 1, we have:

$$B = \{(\neg a \vee \neg b \vee c, N^2), (a, N), (b, N), (g, N)\}.$$

The base K' obtained from Step 1 (after removing tautologies): $K' = \{(\neg a \vee \neg b \vee c \vee g, N^2 + N), (\neg a \vee \neg b \vee c, N^2), (a \vee b \vee g, 3N), (a \vee b, 2N), (a \vee g, 2N), (b \vee g, 2N), (a, N), (b, N), (g, N)\}$.

Since we apply Adjustment on $K'_{\neg c}$, the first idea is to ignore formulas in $K'_{\neg c}$ under the inconsistency level (see Section 3.3). We can check that $Inc(K'_{\neg c}) = N$. Then, $\delta_A(K'_{\neg c})$ is the classical base (obtained by ignoring the ranks) associated with $\{(\neg c, +\infty), (\neg a \vee \neg b \vee c \vee g, N^2 + N), (\neg a \vee \neg b \vee c, N^2), (a \vee b \vee g, 3N), (a \vee b, 2N), (a \vee g, 2N), (b \vee g, 2N)\}$. The second idea is that subsumed disjunctions are not added. In this example, since $\neg a \vee \neg b \vee c$ and $a \vee b, a \vee g, b \vee g$ will

belong to $\delta_A(K_{-c}')$ then there is no need to keep the disjunctions $\neg a \vee \neg b \vee c \vee g$ and $a \vee b \vee g$.

Lastly, the other disjunctions can be refined. Since $C = \{\neg c, \neg a \vee \neg b \vee c, a, b\}$ is inconsistent, then all disjunctions constructed from g and this conflict C are reduced to g .

Therefore, we have $\delta_A(K_{-c}') \equiv \{\neg c, \neg a \vee \neg b \vee c, a \vee b, g\}$ which is equivalent to $\delta_{DMA}(K_{-c})$.

7 Experimental results

We now present some experimental results which illustrate the different behaviour of each strategy. We used a propositional logic implementation of the strategies³. We chose 8 inconsistent bases at random from the DIMACS challenge (aim-50-no) containing 50 variables each and 80 clauses for the first 4, 100 clauses for the others. Then we stratified the bases with 20 clauses per strata, keeping the clauses in their original order. It appeared that each time the conflicts were discovered and weakened in the second strata, no more appeared in the remaining strata. The following table gives the number of clauses in the second strata after applying a given strategy. WDMA (resp. IDMA) stands for whole-DMA (resp. iterative DMA).

#clauses	t1	t2	t3	t4	t5	t6	t7	t8
Adj.	0	0	0	0	0	0	0	0
MA	17	7	8	18	13	7	10	17
DMA	17	54	49	18	21	60	35	18
WDMA	168	149	153	161	161	155	152	160
IDMA	17	54	49	18	21	60	35	18

There are no differences between DMA and IDMA because on these examples consistency was either restored using $d_2(C)$ (t2,t3,t5,t6,t7) or all the clauses involved in a conflict have to be removed. Whole-DMA clearly hides the information contained in the knowledge base by generating a large number of clauses but timewise its fast. Let us now take a look at the time spent computing each strategy.

time (s)	t1	t2	t3	t4	t5	t6	t7	t8
Adj.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MA	137	0.6	2.0	332	6.1	0.3	1.2	304
DMA	136	0.6	2.1	329	6.2	0.3	1.2	302
WDMA	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
IDMA	139	0.6	2.1	329	6.0	0.3	1.2	306

These results can be interpreted as follows: computing the set of clauses involved in conflicts (kernel) is costly, so all methods relying on this information will require small KB's to revise. This can be achieved for instance using modular KB's, a common practice in knowledge engineering.

Interestingly, since the three DMA approaches we introduced are logically equivalent, we can propose one way to efficiently compute the DMA policy: whole DMA, only based on satisfiability testing. This method can be used for instance if the knowledge base is hidden to the final user, and that only the queries are important. On the other hand, if the knowledge base itself is important for the user, such that the revised base must be as "close" as possible to the original one, an IDMA approach should be used (only necessary information will be

³ADS: <http://cafe.newcastle.edu.au/daniel/ADS/>

weakened), but a computational cost must be paid. DMA is a tradeoff between these two policies.

8 Conclusion

We introduced a new family of computationally effective strategies for conflict resolution which can be used for exception handling, iterated belief revision and merging information from multiple sources. The most important feature of our strategy is that it relies on weakening conflicting information rather than removing conflicts completely, and hence it retains at least as much, and in most cases more, information than all other known strategies. Furthermore, it achieves this higher retention of information at no extra computational cost. We compared and contrasted three implementations of our new strategy with existing ones from a theoretical standpoint and by measuring their relative performance. We were also able to show the surprising result that the DMA policy provides a compilation of the lexicographical system which is known to have desirable theoretical properties. DMA offers the clear advantage of obviating the need to explicitly compute the set of all preferred subbases which can be hard. Another pleasing result is that the DMA strategy can be implemented as whole-DMA where the need to explicitly compute the culprits responsible for the conflicts is not required.

References

- [Alchourrón *et al.*, 1985] C. E. Alchourrón, P. Gärdenfors and D. Makinson *On the logic of theory change: Partial meet contraction and revision functions*. Journal of symbolic logic, 50:510-530, 1985.
- [Benferhat *et al.*, 1993] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, H. Prade. *Inconsistency management and prioritized syntax-based entailment*. Proc. IJCAI, France, 640-645.
- [Dubois *et al.*, 1994] D. Dubois, J. Lang, H. Prade. *Possibilistic logic*. Handbook of Logic in AI and Log. Prog., (3), 439-513.
- [Lau *et al.*, 2000] R. Lau, A.H.M. ter Hofstede, P.D. Bruza and K.F. Wong. *Belief Revision and Possibilistic Logic for Adaptive Information Filtering Agents*. Proc. IEEE (ICTAI), Canada, 19-26.
- [Lehmann, 1995] D. Lehmann. Another perspective on default reasoning. Annals of Mathematics and Artif. Intel., 15, 61-82.
- [Spohn, 1988] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In: Causation in Decision, Belief Change and Statistic,(2), Reidel, Dordrecht, 105-134.
- [Williams, 1994] M. A. Williams. Transmutations of knowledge systems. Proc. KR'94, 619-629.
- [Williams, 1996] M. A. Williams. A practical approach to belief revision: reason-based change. Proc. KR'96, 412-421.
- [Williams, 1998] M. A. Williams. Applications of Belief Revision. in Transactions and Change in Logic Databases, LNAI 1472, 285-314, Springer Verlag.
- [Williams and Sims, 2000] M. A. Williams and A. Sims *SATEN: An object-oriented web-based revision and extraction engine*. Proc. of the Int. Workshop on Nonmonotonic Reasoning (NMR' 2000). Online Computer Science Abstract <http://arxiv.org/abs/cs.AI/0003059/>
- [Wong and Lau, 2000] O. Wong and R. Lau *Possibilistic Reasoning for Intelligent Payment Agents*. Proc. of the 2nd Workshop on AI in Electronic Commerce (AIEC' 2000), 1-13.

KNOWLEDGE REPRESENTATION AND REASONING

ACTION AND CAUSALITY

Updates, actions, and planning

Andreas Herzig

IRIT/UPS

F-31062 Toulouse Cedex
France

herzig@irit.fr

Jérôme Lang

IRIT/UPS

F-31062 Toulouse Cedex
France

lang@irit.fr

Pierre Marquis

CRIL/Université d'Artois

F-62307 Lens Cedex
France

marquis@cril.univ-artois.fr

Thomas Polacsek

IRIT/UPS

F-31062 Toulouse Cedex
France

polacsek@irit.fr

Abstract

A general framework for update-based planning is presented. We first give a new family of dependence-based update operators that are well-suited to the representation of simple actions and we identify the complexity of query entailment from an updated belief base. Then we introduce conditional, nondeterministic and concurrent updates so as to encode the corresponding types of action effects. Plan verification and existence are expressed in this update-based framework.

1 Introduction

Three subareas of AI seem significantly connected and still, in the literature, work in each of these areas has been so far disconnected, up to a few exceptions. These areas are (a) belief update, (b) reasoning about action and change, and (c) planning in nondeterministic environments.

Belief update (a) mainly focuses on determining how a belief state should evolve after adding a new piece of information reflecting an explicit evolution of the world; how beliefs persist is studied in depth (this often – yet not always – relies on an assumption of minimal change), as well as causality and dependence between pieces of information; in particular, disjunctive information (reflecting uncertainty about the world after update) is taken into account, and many theoretical results exist (characterization of operators thanks to representation theorems, and computational complexity results).

Reasoning about action (b) focuses on the nature of pieces of information to take account for (preconditions for executability, direct changes, indirect changes or ramifications, static laws of the domain) and on the nature of the actions themselves (determinism, conditional effects, normal and/or exceptional effects, concurrent execution...) using sophisticated languages, generally more expressive than those of belief update but slightly less worked from a theoretical point of view (especially to what concerns representation theorems and complexity results – yet some results exist).

Finally, while (a) and (b) care on determining the consequences on the agent's current belief state of, respectively, a new piece of information and the execution of an action, most of the previous works about planning (c) are centered on plan generation; algorithmic developments have been pushed for-

ward especially when actions are very simple – like in STRIPS – but less so when they are more complex¹.

Our purpose is to show how existing works on belief update can be extended so as to represent complex actions representations and planning in nondeterministic environments.

After some formal preliminaries (Section 2), we introduce in Section 3 a new family of update operators based on literal dependence. Such operators generalize existing dependence-based operators but grasp in a better way the incorporation of the effects of an action in a belief base. Interestingly, this generalization has no influence on computational properties since the complexity of query entailment from an updated belief base is just as hard as classical entailment in the general case (unlike other operators like Winslett's PMA – unless the polynomial hierarchy collapses). Then we show in Section 4 how the effects of more sophisticated (ontic) actions can be represented in our framework by generalizing our family of update operators to deal with conditionals, nondeterminism (distinct from disjunction!) and concurrency; this establishes a closer parallel between updates and effects of complex actions such as considered in action languages. In Section 5, we show how to use our (extended) update operators so as to compute the effect of a (possibly conditional) plan in a nondeterministic domain. We successively consider the case where the environment is fully observable and unobservable. Complexity results for plan verification and plan existence are given for both cases. Finally, connections to related work are discussed in Section 6, before the concluding section.

2 Formal preliminaries

We consider a propositional language $PROP_{PS}$ built up from a finite set PS of propositional variables and the boolean constants \top and \perp . Propositional formulas are denoted by A, B etc. and interpretations over PS (or *worlds*) are denoted by ω etc. We represent them as tuples or conjunctions of literals over PS . In the following, we will identify every propositional formula A with its set of models $Mod(A)$.

Let LIT be the set of literals of the language, i.e., $LIT = \bigcup_{x \in PS} \{x, \neg x\}$. We recall that a formula A is in Nega-

¹Only a few recent works such as [McCain and Turner, 1998], [Rintanen, 1999] and [Ferraris and Giunchiglia, 2000] considered planners with nondeterministic actions represented in sophisticated action languages.

tive Normal Form iff it makes use of the connectives \wedge , \vee , \neg , only and the scope of negation connectives appearing in A includes propositional symbols only. For instance, $A = \neg((\neg a \wedge b) \vee (a \wedge c))$ is equivalent to the NNF formula $(a \vee \neg b) \wedge (\neg a \vee \neg c)$ ². $A[x \leftarrow C]$ denotes the formula obtained by uniformly replacing every occurrence of propositional variable x by the formula C . $A[l \leftarrow C]$ denotes the formula obtained by replacing in a uniform way every positive (resp. negative) occurrence of literal l in the NNF of A by C (resp. $\neg C$). We denote by $Lit(A) \subseteq LIT$ the set of literals appearing in the NNF of A . In the previous example, $Lit(A) = \{a, \neg a, \neg b, \neg c\}$. For any $L \subseteq LIT$, we note $NEG(L) = \{\neg l \mid l \in L\}$, supposing $\neg\neg x$ identified with x . Lastly, if ω is a world and L a consistent set of literals from LIT , then $Force(\omega, L)$ is the world that gives the same truth value as ω to all variables except the variables of literals of L and $Force(\omega, L) \models \bigwedge_{l \in L} l$. For instance, if $\omega = (a, \neg b, c, d)$ and $L = \{b, \neg d\}$ then $Force(\omega, L) = (a, b, c, \neg d)$.

A *belief update operator* \diamond maps the propositional belief base (a formula) B representing the initial beliefs of a given agent and an input formula A reflecting some explicit evolution of the world [Katsuno and Mendelzon, 1991], to a new set of beliefs $B \diamond A$ held by the agent after this evolution has taken place. Katsuno and Mendelzon [Katsuno and Mendelzon, 1991] proposed a general semantics for update. The most prominent feature of KM updates (distinguishing updates from revision) is that update must be performed model-wise, i.e., $Mod(B \diamond A) = \bigcup_{\omega \models B} \omega \diamond A$.

Finally, we assume the reader familiar with some basic notions of computational complexity (see [Papadimitriou, 1994] otherwise).

3 A new family of dependence-based updates

Many proposals for update operators have been made. Recently, several authors showed that there are good reasons for building a belief operator from a dependence relation [Doherty *et al.*, 1998] [Herzig and Rifi, 1999]. The dependence-based update of a belief base B by an input formula A consists in first forgetting in B all information “relevant” to A (leaving unchanged the truth value of variables not relevant to the update), and then expanding the result with A . What remains to be defined is the notion of “being relevant to”.

3.1 Formula-variable dependence

A *formula-variable dependence function* is modelled a mapping Dep from $PROP_{PS}$ to 2^{PS} . Many choices for Dep are possible (see [Herzig and Rifi, 1999] for details). Whatever the choice, the *dependence-based update* of a world ω by a formula A w.r.t. Dep , denoted by $\omega \diamond_{Dep} A$, is the set of all worlds ω' such that $\omega' \models A$, and for every propositional variable x from PS such that $x \notin Dep(A)$, ω and ω' assign the same truth value to x .

Interestingly, the complexity of query entailment from an updated belief base is “only” **coNP**-complete when dependence-based updates are considered [Liberatore, 2000]

²Note that when A is built up from the connectives \wedge , \vee , \neg , \Rightarrow , it can be turned into an equivalent NNF formula in linear time by “pushing down” every occurrence of \neg and removing double negations. Abusing words, we call the resulting formula *the NNF* of A .

while entailment when most usual other operators - like Winslett’s PMA - are used is at the second level of the polynomial hierarchy.

3.2 Formula-literal dependence

Forgetting everything about the *variables* involved in the update often leads to *forgetting too much*. Consider a robot being told to go to a room with two doors 1 and 2, and to ensure that at least one of them is red (possibly by painting one of the doors), which can be represented as an update by $red1 \vee red2$. Suppose now that both doors are initially red: $B = red1 \wedge red2$. Then, using any dependence function Dep s.t. $\{red1, red2\} \subseteq Dep(red1 \vee red2)$, we get $B \diamond_{Dep} (red1 \vee red2) = red1 \vee red2$. Thus, we forgot that both doors were already red. Clearly, this is not what is expected: intuitively, we should not forget that doors 1 and 2 are red, because updating by $red1 \vee red2$ has no *negative* influence on $red1$ nor on $red2$. Hence, only *negative* occurrences of $red1$ and $red2$ should be forgotten before expanding by the input formula, not positive ones.

This may not be a problem in some contexts, but clearly, when reasoning about actions, these update operators make us forget too much and therefore do not handle the frame problem correctly. In order to cope with this limitation, we introduce a new family of dependence-based update operators. Such operators generalize existing ones because they are based on formula-literal dependence [Lang *et al.*, 2000], a more general notion than formula-variable dependence.

From now on, a formula-literal dependence function is a mapping $Dep : PROP_{PS} \rightarrow 2^{LIT}$. Many full-sense Dep can be considered. Like formula-variable dependence functions, $Dep(A)$ can be basically defined as $Lit(A)$ or $DepLit(A)$, where the latter is defined as follows:

Definition 1 (FL independence) [Lang *et al.*, 2000] *Let A be a formula from $PROP_{PS}$ and $l (= x \text{ or } \neg x)$ be a literal from LIT . A is literal-independent of x iff $A[x \leftarrow \top] \equiv A$, and A is literal-independent of $\neg x$ iff $A[x \leftarrow \perp] \equiv A$. Finally, A depends on l iff it is not literal-independent of l . We denote by $DepLit(A)$ the set of literals A depends on.*

Thus, $A = (a \vee b) \wedge (\neg a \vee c) \wedge (a \vee b \vee \neg c)$ is literal-dependent on a , $\neg a$, b and c ; but not on $\neg c$, because it is equivalent to $(a \vee b) \wedge (\neg a \vee c)$, and $\neg c$ does not appear in the NNF of the latter.

More sophisticated dependence functions can also be designed by considering explicit dependence (binary) relations δ on LIT . One of them is $Dep(A) = \bigcup_{l \in Lit(A)} \delta(l)$ (and similarly with $DepLit(A)$). We can also take into account persistent literals, i.e., literals remaining true whatever happens, such as \neg alive (contrarily to alive). For example, $Dep(A) = DepLit(A) \cap NEG(Pers)$ where $Pers$ is a given set of persistent literals is a possible dependence function. Clearly, persistent literals cannot be taken into account by formula-variable dependence. The other way round, every formula-variable dependence function can be simulated by a formula-literal dependence function.

3.3 Update based on formula-literal dependence

Definition 2 *The update $\omega \diamond_{Dep} A$ of the world ω by the formula A w.r.t Dep is the set of all worlds ω' s.t. $\omega' \models A$ and there exists $L \subseteq Dep(A)$ s.t. $\omega = Force(\omega', NEG(L))$.*

The next result gives us a way to compute $B \diamond_{Dep} A = \bigcup_{\omega \in Mod(B)} \omega \diamond_{Dep} A$ in practice. We first need to define *literal forgetting* [Lang et al., 2000].

Definition 3 (literal forgetting) For a formula B and $L \subseteq LIT$, $ForgetLit(B, L)$ is the formula inductively defined by
(1) $ForgetLit(B, \emptyset) = B$;
(2) $ForgetLit(B, \{l\}) = B[l \leftarrow \top] \vee (\neg l \wedge B)$;
(3) $ForgetLit(B, \{l\} \cup L) = ForgetLit(ForgetLit(B, L), \{l\})$

Proposition 1

$B \diamond_{Dep} A \equiv A \wedge ForgetLit(B, NEG(Dep(A)))$.

Example 1 $B = (\neg a \vee \neg d) \wedge (a \vee b) \wedge (a \vee c \vee d)$, $A = d$ and let us take $Dep = DepLit$. We have $ForgetLit(B, Dep(A)) = ForgetLit(B, \{d\}) = (\neg a \vee \neg d) \wedge (a \vee b)$; hence $B \diamond_{Dep} A \equiv d \wedge \neg a \wedge b$.

Interestingly, for all the update operators based on literal dependence listed above, we have the following complexity result (this slightly extends Theorem 15 from [Liberatore, 2000]):

Proposition 2 Given three formulas A , B and C , $B \diamond_{Dep} A \models^? C$ is **coNP-complete**.

If $DepLit(A) \subseteq Dep(A)$, then A and $ForgetLit(B, NEG(Dep(A)))$ are strongly separable, in the sense that, if $l \in DepLit(A)$, then $\neg l \notin DepLit(ForgetLit(B, NEG(Dep(A))))$ holds. This is an important property from a computational point of view when CNF queries are considered since:

Proposition 3 If C and D are two strongly separable formulas, then they are separable in the sense of Levesque [Levesque, 1998], i.e., for every clause E , we have $C \wedge D \models E$ iff $C \models E$ or $D \models E$.

Accordingly, one global entailment test can be replaced by two local (simpler) entailment tests, and exponential savings can be achieved this way in practice. Now, clausal entailment from a *ForgetLit* formula reduces to clausal entailment from a classical formula:

Proposition 4 Let E be a non-tautological clause. Then $Forget(B, L) \models E$ iff $B \models \bigvee_{l \in Lit(E) \setminus L} l$.

Altogether, the two previous propositions give us some tractable restrictions for clausal query entailment from an updated belief base. Let us say that a formula is tractable when it belongs to a class of formulas for which clausal entailment is known as tractable (e.g., the Horn CNF class). When \diamond_{Dep} satisfies $DepLit(A) \subseteq Dep(A)$, A is a tractable formula for which literal-(in)dependence can be tested in polynomial time and B is a tractable formula, then determining whether $B \diamond_{Dep} A \models C$ can be achieved in polynomial time for every CNF formula C . This is the case for instance when $Dep = DepLit$, B is a Horn CNF formula and A is a set of binary clauses³.

³Here, clausal entailment from the updated base $B \diamond_{Dep} A$ is even easier than clausal entailment from the corresponding expanded base $B \wedge A$ since determining whether $A \wedge B \models C$ is **coNP-complete** in this situation – tractable classes are known not to mix well.

4 Updates and actions

The aim of this section is to show that belief update is not far from being able to determine the effects of an arbitrary action. In order to do so, we introduce the notion of *extended input*.

Definition 4 (extended inputs) Extended inputs are defined inductively as follows:

- every formula A of $PROPPS$ is an extended input;
- if Φ and Ψ are extended inputs then $\Phi \cup \Psi$ is an extended input (nondeterminism);
- if Φ and Ψ are extended inputs then $\Phi \parallel \Psi$ is an extended input (concurrency);
- if Φ and Ψ are extended inputs and C is a formula of $PROPPS$ then $\text{if } C \text{ then } \Phi \text{ else } \Psi$ is an extended input (conditional).

Definition 5 (updates by extended inputs) Let \diamond be an arbitrary update operator, ω a world and Φ an extended input. We define the update of ω by Φ inductively as follows:

- if $A \in PROPPS$, then $\omega \diamond A$ is given by the definition of \diamond ;
- $\omega \diamond (\Phi \cup \Psi) = (\omega \diamond \Phi) \cup (\omega \diamond \Psi)$;
- $\omega \diamond \text{if } C \text{ then } \Phi \text{ else } \Psi = \begin{cases} \omega \diamond \Phi & \text{if } \omega \models C \\ \omega \diamond \Psi & \text{if } \omega \models \neg C \end{cases}$
- $\omega \diamond (\Phi \parallel \Psi) = \omega \diamond (\Gamma(\Phi \parallel \Psi))$
where $\Gamma(\Phi \parallel \Psi)$ is defined inductively by:
$$\Gamma(\Phi \parallel \Psi) = \begin{cases} \Phi \wedge \Psi & \text{if } \Phi, \Psi \in PROPPS \\ \Gamma(\Phi \parallel \Psi_1) \cup \Gamma(\Phi \parallel \Psi_2) & \text{if } \Psi = \Psi_1 \cup \Psi_2 \\ \text{if } C \text{ then } \Gamma(\Phi \parallel \Psi_1) \text{ else } \Gamma(\Phi \parallel \Psi_2) & \text{if } \Psi = (\text{if } C \text{ then } \Psi_1 \text{ else } \Psi_2) \\ \Gamma(\Psi \parallel \Phi) & \text{otherwise} \end{cases}$$

As before, $B \diamond \Phi = \bigcup_{\omega \in Mod(B)} (\omega \diamond \Phi)$.

$\omega \diamond (\Phi \cup \Psi)$ is a *nondeterministic update*, namely, ω is nondeterministically updated by Φ or by Ψ . This has been first introduced by Brewka and Hertzberg [Brewka and Hertzberg, 1993], who have observed that $\omega \diamond (A \cup B)$ is in general different from $\omega \diamond (A \vee B)$. In the case where $\Phi = A$ and $\Psi = B$ are simple propositional formulas, updating ω by $A \cup B$ intuitively means that a nondeterministic action has been performed, whose possible effect is either A or B , and that (quoting from [Brewka and Hertzberg, 1993]) the planner has no way to know whether one of the postcondition is already satisfied and nondeterministically choose to update by A or by B . A first example is the `TOSS` action, representable by the extended input `head \cup \neg head`. Another example (adapted from [Brewka and Hertzberg, 1993]): recall the red doors example (Section 2). If the robot is told to paint door 1 or door 2 in red but has no way to know whether one of the doors is already red (because it has no sensors), then the action should be modelled by `red1 \cup red2`. On the other hand, the action consisting of an update by `red1 \vee red2` corresponds to a context where the robot can check whether one of the doors is initially red (and leaves the world unchanged if this is the case). The introduction of these two kinds of nondeterministic actions needs both standard disjunction and \cup . Note that we have $B \diamond (A \cup C) \equiv (B \diamond A) \vee (B \diamond C)$.

$\omega \diamond \text{if } C \text{ then } \Phi \text{ else } \Psi$ is a *conditional update*: the initial model is updated by Φ or by Ψ depending on whether ω

satisfies the branching condition C or not. In the case where $\Phi = A$ and $\Psi = D$ are simple propositional formulas, we have $B \diamond \text{if } C \text{ then } A \text{ else } D \equiv (B \wedge C) \diamond A \vee (B \wedge \neg C) \diamond D$. Conditional updates are needed for action with conditional effects, such as the well-used turkey shooting action, which corresponds to an update by `if loaded then (\neg alive \wedge \neg loaded) else \top` . Conditional actions are not well handled with pure unconditional updates.

$\omega \diamond (\Phi \parallel \Psi)$ is a *concurrent update*. It basically works by gathering conjunctively all possible combinations of alternative results, and then performing the corresponding updates separately; in particular, $B \diamond ((A_1 \cup A_2) \parallel (B_1 \cup B_2)) \equiv B \diamond (A_1 \wedge B_1) \vee B \diamond (A_1 \wedge B_2) \vee B \diamond (A_2 \wedge B_1) \vee B \diamond (A_2 \wedge B_2)$. This can be easily generalized to the case where more than two extended inputs must be considered as concurrent update formulas. Concurrent updates aim at representing compactly independent effects of actions. Consider the action of loading the gun which has two independent effects, namely: (1) the gun is loaded and (2) the turkey goes hiding if it is not deaf. It corresponds to the concurrent update by $\alpha = (\text{loaded} \parallel \text{if } \neg \text{deaf} \text{ then hidden else } \top)$. Concurrency can as well be used for concurrent (or parallel) actions⁴.

Consider parallel actions having some inconsistent effects. As the effect of the update by their combination is an empty set of models, this has no impact on the final result. This means that our notion of action compatibility is *very loose*. For instance, if $\omega = (a, b, c)$ and if we have two actions α and β corresponding to the extended inputs $\alpha = (a \wedge b) \cup (a \wedge c)$ and $\beta = \neg a \cup \neg b$ then $\omega \diamond (\alpha \parallel \beta) = (\omega \diamond \perp) \cup (\omega \diamond \perp) \cup (\omega \diamond \perp) \cup (\omega \diamond (a \wedge \neg b \wedge c)) = \{(a, \neg b, c)\}$. More precisely, two nondeterministic actions $\alpha = A_1 \cup \dots \cup A_p$ and $\beta = B_1 \cup \dots \cup B_q$ are compatible (i.e., concurrently executable) iff there is an effect A_i of α and an effect B_j of β such that $A_i \wedge B_j$ is consistent⁵.

The previous definition is independent of the choice of a particular update operator. In the case of a dependence-based operator, the complexity results established in Section 3 enable us to say that the prediction problem (i.e., the problem of determining if the execution of an action in a given belief state leads to satisfying a given formula) is **coNP**-complete⁶.

Note that usual ramification methods, such as the use of integrity constraints or causal rules (see [Doherty *et al.*, 1998] for their handling in an update framework) can be added on top of our framework. We do not discuss them since it is not

⁴Note that our notion of concurrency differs from concurrency in dynamic logic which is defined by $(\omega \diamond \Phi) \cap (\omega \diamond \Psi)$. Like in language \mathcal{C} [Giunchiglia and Lifschitz, 1998], concurrency means that the effects of the concurrent actions must be gathered before being used as inputs for belief update. Consider for instance the two actions “open the door” and “open the window”, corresponding to the updates `door` and `window`, applied concurrently to the initial state $\omega = (\neg \text{door}, \neg \text{window})$. For any dependence-based update operator \diamond for which `door` and `window` are independent we get $\omega \diamond \text{door} = (\text{door}, \neg \text{window})$, $\omega \diamond \text{window} = (\neg \text{door}, \text{window})$, and thus $(\omega \diamond \text{door}) \cap (\omega \diamond \text{window}) = \emptyset$, while with our definition we get the intended result $\omega \diamond (\text{door} \parallel \text{window}) = \omega \diamond (\text{door} \wedge \text{window}) = (\text{door}, \text{window})$.

⁵Other notions of compatibility can be represented in our framework. They are not developed further due to space limitations.

⁶If the update operator was instead Π_2^P -complete, as many operators are, then extended update would be Π_2^P -complete too.

a contribution of our paper.

5 Update-based planning

5.1 Definitions

In the following, we consider standard *ontic* actions, which are meant to act on the world, and do not have any other effects on the knowledge state of the agent than the realization of its ontic effects. In our framework, a standard ontic action α is described by an extended input, denoted by α as well (thus, we identify the action and its effects, which will not lead to ambiguities), and determining its effects on a world ω simply consists in updating ω by α .

What is missing now in order to define a plan is the notion of *update sequences*.

Definition 6 (plans) *Given a finite set of standard ontic actions ACT_O , (branching) plans are defined inductively by:*

- λ (empty plan) is a plan;
- for any extended input $\alpha \in ACT_O$, α is a plan;
- if π_1 and π_2 are plans, then $\pi_1; \pi_2$ is a plan;
- if π_1 and π_2 are plans and C is a propositional formula, then `if C then π_1 else π_2` is a plan.

Thus, plans are sequential and conditional extended updates (these extended updates being themselves conditional, nondeterministic and concurrent)⁷. At the plan level, nondeterministic choice and parallel composition are not allowed.

Definition 7 (planning problems) *A planning problem is a triple $\mathcal{P} = \langle I, ACT_O, G \rangle$ where:*

- I is a propositional formula representing the knowledge on the initial state;
- $ACT_O = \{\alpha_1, \dots, \alpha_q\}$ is a set of standard ontic actions;
- G is a propositional formula representing the goal.

Note that these definitions do not mention anything regarding observability. They can therefore be used as generic definitions. What changes when various assumptions about observability are made is the type of plans allowed. When full observability is assumed, any branching is allowed, while with unobservability, branching is never allowed. With partial observability we may branch only on conditions whose truth value can be determined by the agent.

Definition 8 (trajectories) *A trajectory τ is a nonempty sequence $\tau = \langle \omega_0, \dots, \omega_T \rangle$ of models ($T \geq 0$). If $\tau = \langle \omega_0, \dots, \omega_T \rangle$, then $\text{length}(\tau) = T$. If π is a plan and I an initial knowledge base, τ is a possible trajectory for (I, π) w.r.t. ACT_O iff (1) $\omega_0 \models I$ and (2) one of the following conditions holds:*

1. $\pi = \lambda$ and $T = 0$;
2. $\pi \in ACT_O$ and $T = 1$ and $\omega_1 \in \text{Mod}(\omega_0 \diamond \pi)$;

⁷Note that `if .. then .. else` is used both at the action and at the plan levels. Formally, both structures are identical but both kinds of conditionals do not share the same purpose; especially, observability considerations may restrict the range of allowed branching conditions at the plan level, but not at the action level.

3. $\pi = \pi_1; \pi_2$ and τ can be decomposed in two subtrajectories $\tau_1 = \langle \omega_0, \dots, \omega_i \rangle$, $\tau_2 = \langle \omega_i, \dots, \omega_T \rangle$, s.t. τ_1 is a possible trajectory for (I, π_1) , and τ_2 is a possible trajectory for (ω_i, π_2) ;
4. $\pi = \text{if } C \text{ then } \pi_1 \text{ else } \pi_2$ and
 - either $\omega_0 \models C$ and τ is a possible trajectory for (I, π_1)
 - or $\omega_0 \models \neg C$ and τ is a possible trajectory for (I, π_2) .

Definition 9 (succeeding plans) A plan π is a succeeding plan for a planning problem $\mathcal{P} = \langle I, ACT_O, G \rangle$ iff every possible trajectory τ for (I, π) is s.t. $\omega_{length(\tau)} \models G$.

5.2 Fully observable environments

Under full observability, any plan is allowed. Let us consider the complexity of plan verification and existence.

Proposition 5 (complexity of plan verification/existence) Suppose that the update operator used \diamond is one of the dependence-based operator given in the previous sections.

1. determining whether π is a succeeding plan for $\mathcal{P} = \langle I, ACT_O, G \rangle$ is **coNP**-complete;
2. determining whether there exists a succeeding plan π for $\mathcal{P} = \langle I, ACT_O, G \rangle$ is **EXPTIME**-complete, and determining whether there exists a succeeding plan π of polynomially bounded length for \mathcal{P} is **PSPACE**-complete.

Point 1. is a key result. It is a strong generalization of Theorem 20 from [Liberatore, 2000]. Point 2. is easier to establish: the upper bounds are a byproduct of Point 1., and the lower bounds are derived from Littman's results on probabilistic planning [Littman, 1997]. These results (especially Point 1.) show that update-based planning in fully observable environments is not harder than STRIPS-based nondeterministic planning, which (somewhat) means that the facilities brought with the use of update are "for free".

5.3 Unobservable environments

Definition 10 (unbranching plans) An unbranching plan is a (possibly empty) sequence $\alpha_0; \dots; \alpha_T$ of standard ontic actions, or equivalently a plan in which no $\text{if } \dots \text{ then } \dots \text{ else appears at the outer level}$ ⁸.

Trajectories and succeeding plans are defined the same way as above. The following property, reformulating succeeding plans in terms of iterated updates, is almost straightforward:

Proposition 6 Let $\pi = \alpha_1; \dots; \alpha_T$ be an unbranching plan, and $\mathcal{P} = \langle I, ACT_O, G \rangle$ a planning problem. Then π is a succeeding plan for \mathcal{P} iff $((I \diamond \alpha_1) \diamond \alpha_2) \diamond \dots \diamond \alpha_T \models G$.

Proposition 7 (complexity of plan verification/existence) Suppose the update operator used \diamond is one of the dependence-based operator given in the previous sections.

1. determining whether an unbranching plan π is a succeeding plan for $\mathcal{P} = \langle I, ACT_O, G \rangle$ is **coNP**-complete.

⁸However, $\text{if } \dots \text{ then } \dots \text{ else}$ may appear inside the extended inputs, for representing conditional effects.

2. determining whether there exists a succeeding unbranching plan π for $\mathcal{P} = \langle I, ACT_O, G \rangle$ is **EXPSpace**-complete, and determining whether there exists a succeeding unbranching plan π for \mathcal{P} of polynomially bounded length is Σ_2^P -complete.

The **EXPSpace**-hardness result is a direct consequence of a result from [Haslum and Jonsson, 1999]. The latter point (Σ_2^P -completeness) is similar to results in [Baral *et al.*, 1999] and in [da Costa *et al.*, 1997], which again suggests that dependence-based updates can be added "for free".

6 Related work

[Brewka and Hertzberg, 1993] were the first to use belief update operators for representing actions with conditional and nondeterministic effects. In Sections 3 and 4, we truly follow in their steps. They use Winslett's PMA update operator, which has been shown in many places to possess severe drawbacks (both from the points of view of expressivity and complexity). By making use of further advances in belief update, especially in what concerns dependence-based update, we were able to go much further than they did towards the practical computation of plans (and as well by allowing for concurrent actions, conditional plans and epistemic actions). See also the transition-based update-like operators of Cordier and Siegel [Cordier and Siegel, 1995].

[Fargier *et al.*, 2000] use dependence-based update for one-stage decision making processes (that can be viewed as degenerated planning problems). Their use of a formula-variable dependence is, as we show in Section 3, not the best choice for reasoning about action, and they do not use nondeterministic updates, nor sequences of updates.

[del Val and Shoham, 1994] point out some correspondances between the Katsuno-Mendelzon postulates and generic properties of action (by rewriting KM postulates in the situation calculus and showing that each KM postulate can be expressed as a specific property of actions), which enable them to give some intuitive reasons to accept or reject some of the KM postulates. Rather than practically proposing action languages based on belief update operators as we did, their work remains at the meta-level and made us think that such an update-based framework for planning was feasible.

[Shapiro *et al.*, 2000] represent iterated updates as well as epistemic actions in the situation calculus. They distinguish between ontic actions (leading to updates) and epistemic (leading to revisions). They represent the set of possible worlds (or situations) explicitly, while, using STRIPS-like representation and a propositional formalism, we represent it in a compact way, which makes our framework very different from theirs.

A line of work which is parallel to ours is the planning as satisfiability frameworks with actions languages, especially [Giunchiglia, 2000] based on the language \mathcal{C} and [McCain and Turner, 1998] based on the causal theory developed by the authors. The main differences between these languages and update is in the handling of disjunction: the literal completions used in [Giunchiglia, 2000] and [McCain and Turner, 1998] require that dynamic laws expressing the effects of actions have a single literal in their head. The exact links

between these expressive languages and belief update is an interesting topic for further research. [Geffner, 1996] also makes use of causal theories for actions and planning, and uses plausibility functions to distinguish various levels of exceptionality in the effects of actions.

7 Conclusion

In this paper, we have presented a new update-based framework for planning. Our contribution is manifold. We have introduced a new family of dependence-based update operators, based on formula-literal dependence, which includes existing dependence-based operators as specific cases, suits better the needs of reasoning about action, and whose computational complexity is not being above that of classical entailment. We have also shown how belief update has to be augmented with conditionals, nondeterminism and concurrency in order to be fully adapted to the representation of actions. Third, we have shown how to define (conditional or unconditional) plans in our framework. Finally, we have shown that the complexity of plan verification and existence in our framework is not higher than with a STRIPS-like representation.

This work calls for several perspectives. One of them concerns the introduction of partial observability in our framework. The simplest way to do it consists in considering that a fixed set of formulas is directly observable; allowed branching conditions are in this case formulas whose truth value can be determined from the truth values of these formulas. A more expressive solution consists in allowing for epistemic actions in the framework. Unlike ontic actions, epistemic actions leave the world unchanged and therefore do not imply an update, but instead, a knowledge expansion. In order to distinguish between facts and knowledge, an epistemic modality \mathcal{K} can be introduced. Plans can be defined in a similar way as before except for branching conditions, for which we allow only for epistemically interpretable branching conditions as in [Herzig *et al.*, 2000].

Acknowledgements

The third author has been partly supported by the IUT de Lens and the Région Nord/Pas-de-Calais.

References

[Baral *et al.*, 1999] C. R. Baral, V. Kreinovich, and R. Trejo. Computational complexity of planning and approximate planning in presence of incompleteness. In *Proc. of IJCAI'99*, pages 948–955, 1999.

[Brewka and Hertzberg, 1993] G. Brewka and J. Hertzberg. How to do things with worlds: on formalizing actions and plans. *J. of Logic and Computation*, 3(5):517–532, 1993.

[Cordier and Siegel, 1995] M.O. Cordier and P. Siegel. Prioritized transitions for updates. In *Proc. of ECSQARU'95*, pages 142–150. LNAI 946, Springer-Verlag, 1995.

[da Costa *et al.*, 1997] C. da Costa, F. Garcia, J. Lang, and R. Martin-Clouaire. Possibilistic planning: Representation and complexity. In *Proc. of ECP'97*, pages 143–155, 1997.

[del Val and Shoham, 1994] A. del Val and Y. Shoham. Deriving properties of belief update from theories of action. *J. of Logic, Language, and Information*, 3:81–119, 1994.

[Doherty *et al.*, 1998] P. Doherty, W. Łukaszewicz, and E. Madalińska-Bugaj. The PMA and relativizing change for action update. In *Proc. of KR'98*, pages 258–269, 1998.

[Fargier *et al.*, 2000] H. Fargier, J. Lang, and P. Marquis. Propositional logic and one-stage decision making. In *Proc. of KR'2000*, pages 445–456, 2000.

[Ferraris and Giunchiglia, 2000] P. Ferraris and E. Giunchiglia. Planning as satisfiability in nondeterministic domains. In *Proc. of AAI'00*, pages 748–753, 2000.

[Geffner, 1996] H. Geffner. A qualitative model for temporal reasoning with incomplete information. In *Proc. of AAI'96*, pages 1176–1181, 1996.

[Giunchiglia and Lifschitz, 1998] E. Giunchiglia and V. Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. of AAI'98*, pp. 623–630, 1998.

[Giunchiglia, 2000] E. Giunchiglia. Planning as satisfiability with expressive action languages: Concurrency, constraints, and nondeterminism. In *Proc. of KR'2000*, pages 657–666, 2000.

[Haslum and Jonsson, 1999] P. Haslum and P. Jonsson. Some results on the complexity of planning with incomplete information. In *Proc. of ECP'99*, 308–318, 1999.

[Herzig and Rifi, 1999] A. Herzig and O. Rifi. Propositional belief update and minimal change. *Artificial Intelligence*, 115:107–138, 1999.

[Herzig *et al.*, 2000] A. Herzig, J. Lang, D. Longin, and T. Polacsek. A logic for planning under partial observability. In *Proc. of AAI'2000*, pages 768–773, 2000.

[Katsuno and Mendelzon, 1991] H. Katsuno and A. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[Lang *et al.*, 2000] J. Lang, P. Liberatore, and P. Marquis. Propositional independence, part I: formula-variable dependence and forgetting. Technical report, IRIT, 2000.

[Levesque, 1998] H. Levesque. A completeness result for reasoning with incomplete first-order knowledge bases. In *Proc. of KR'98*, pages 14–23, 1998.

[Liberatore, 2000] P. Liberatore. The complexity of belief update. *Artificial Intelligence*, 119:141–190, 2000.

[Littman, 1997] M. L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proc. of AAI'97*, pages 748–754, 1997.

[McCain and Turner, 1998] N. McCain and H. Turner. Satisfiability planning with causal theories. In *Proc. of KR'98*, pages 212–223, 1998.

[Papadimitriou, 1994] Ch. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[Rintanen, 1999] J. Rintanen. Constructing conditional plans by a theorem prover. *Journal of Artificial Intelligence Research*, 10:323–353, 1999.

[Shapiro *et al.*, 2000] S. Shapiro, M. Pagnucco, Y. Lespérance, and H. J. Levesque. Iterated belief change in the situation calculus. In *Proc. of KR'00*, pages 527–538, 2000.

Causality and Minimal Change Demystified

Maurice Pagnucco

Computational Reasoning Group
Department of Computing
Division of ICS
Macquarie University
NSW, 2109, Australia
morri@ics.mq.edu.au

Pavlos Peppas

Dept. of Business Administration
University of Patras
Patras 26500, Greece
ppeppas@otenet.gr

Abstract

The Principle of *Minimal Change* is prevalent in various guises throughout the development of areas such as reasoning about action, belief change and nonmonotonic reasoning. Recent literature has witnessed the proposal of several theories of action that adopt an explicit representation of *causality*. It is claimed that an explicit notion of causality is able to deal with the frame problem in a manner not possible with traditional approaches based on minimal change.

However, such claims remain untested by all but representative examples. It is our purpose here to objectively test these claims in an abstract sense; to determine whether an explicit representation of causality is capable of providing something that the Principle of Minimal Change is unable to capture. Working towards this end, we provide a precise characterisation of the limit of applicability of minimal change.

1 Introduction

The problem of reasoning about action and change has been one of the major preoccupations for artificial intelligence researchers since the inception of the field. One of the early tenets applied when reasoning about such phenomena was that *as little as possible should change in the world when performing an action*; what we might call the *Principle of Minimal Change*.¹ This principle is manifest in many guises: preferential-style systems [Shoham, 1988], persistence approaches [Krautz, 1986], circumscription [McCarthy, 1980], etc. Over the years, aspects of this principle have been called into question leading to a variety of suggested fixes: fixed versus variable predicates in circumscription, occluded fluents [Sandewall, 1989], frame fluents [Lifschitz, 1990], to name but a few. Moreover, in the more recent literature explicit representations of causality have found favour [Lin, 1995; McCain and Turner, 1995; 1997; Thielscher, 1997]. However, what is not clear—beyond some simple representative

¹Although, one might be tempted to say that the Principle of Minimal Change is more general in scope.

examples—is the purchase afforded by explicitly representing causality over the more traditional minimal change approaches. *It is this imbalance that this paper seeks to redress in a clear and objective manner.* In fact, the results we present here have further reaching consequences, giving a rather lucid characterisation of the extent of applicability of minimal change. By this we mean that, given a framework for reasoning about action and change, it will be clear whether such a framework can be modelled by minimal change once certain properties of the framework can be established.

We achieve our aims through a correspondence between two formal systems which we call *dynamic systems* and *preferential models* respectively. Intuitively, the dynamic system is an abstract modelling of the dynamic domain under consideration (the behaviour of which we wish to reason about). Essentially, this abstract model captures the domain at hand by a result function $\mathcal{R}(w, \alpha)$ which returns the states (of the domain) that could possibly result from the application of an action with direct effects α (i.e., postconditions) at the initial state w . A preferential model on the other hand is a formal structure that encodes the Principle of Minimal Change in an abstract and quite general manner. With the aid of preferential models we are able to provide a precise characterisation of the class of dynamic systems that are amenable to theories of action based on minimal change; we call such dynamic systems *minimisable*. Having a precise characterisation of minimisable dynamic systems we can then examine whether theories of action adopting an explicit representation of causality, which we shall call *causal theories of action* are capable of forms of reasoning that cannot be captured by the Principle of Minimal Change; more precisely, we can examine whether causal theories of action are applicable *outside* the scope of minimisable dynamic systems. According to the results reported herein, the logic of action proposed by Thielscher [1997] is indeed applicable to non-minimisable dynamic systems, whereas, perhaps surprisingly, McCain and Turner's causal theory of action [McCain and Turner, 1995] has a range of applicability that is subsumed by the class of minimisable dynamic systems.

In the following section we introduce both dynamic systems and preferential models. Moreover, we state clearly the notion of *minimal change* that we shall adopt here. In Section 3 we examine the formal properties that the result function of a dynamic system must obey in order for it to be *min-*

imisable. Section 4 presents an analysis of some of the theories of action found in the literature. We end with a discussion and conclusions, including pointers to future work.

2 Dynamic Systems and Preferential Systems

As mentioned above, the main results in this paper will be achieved by demonstrating a correspondence between two formal systems. The first, called a *dynamic system*, is meant to serve as a general abstraction of domains (such as the *blocks world*, or the domain described by the *Yale Shooting Problem*, etc.), for which theories of action are designed to reason about. Our main interest shall be in the properties of the dynamic system's result function. In particular, we shall formulate necessary and sufficient conditions under which the system's result function can be characterised in terms of an appropriately defined *minimisation policy*. Minimisation policies are in turn encoded by our second formal system called a *preferential model*. Dynamic systems and preferential models are formally defined below.

2.1 Dynamic Systems

Throughout this article we shall be working with a *finitary* propositional language \mathcal{L} the details of which shall be left open.² We shall often refer to \mathcal{L} as the *object language*. We shall call the propositional variables of \mathcal{L} *fluents*. The set of all fluents will be denoted by $\mathcal{F}_{\mathcal{L}}$. A *literal* is either a fluent or the negation of a fluent. We shall denote the set of all literals by $\mathcal{Z}_{\mathcal{L}}$. A *state* of \mathcal{L} (also referred to as an *object state*) is a maximally consistent set of literals. The set of all states of \mathcal{L} is denoted by $\mathcal{M}_{\mathcal{L}}$. For a set of sentences G of \mathcal{L} , by $[G]$ we denote the set of all states of \mathcal{L} that satisfy G , i.e. $[G] = \{r \in \mathcal{M}_{\mathcal{L}} : r \vdash G\}$. Finally, for a sentence φ of \mathcal{L} we shall use $[\varphi]$ as an abbreviation for $[\{\varphi\}]$.

Definition 2.1 A dynamic system is a triple $W = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ where,

- \mathcal{S} is a nonempty subset of $\mathcal{M}_{\mathcal{L}}$ the elements of which we shall call *valid states*.
- \mathcal{A} is a nonempty set of sentences of \mathcal{L} . The intended meaning of the sentences in \mathcal{A} is that they correspond to the *postconditions* (or direct effects) of actions. For simplicity, we shall identify actions with their postconditions and refer to the sentences in \mathcal{A} as *actions*.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$, is called the *result function*.

Intuitively, the result function $\mathcal{R}(w, \alpha)$ returns the set of object states considered to be possible resultant states after applying the action with postcondition α at the object state w . If for a certain w, α , it happens that $\mathcal{R}(w, \alpha) = \emptyset$, this is taken to mean that α is not applicable at w .

2.2 Extensions of the Object Language

Despite the many different ways in which the principle of minimal change has been encoded [McCarthy, 1980; Winslett, 1988; Katsuno and Mendelzon, 1992; Doherty, 1994;

²By a language, we intend all well formed formulae of that language.

Sandewall, 1996], a feature that is common to all these approaches is the existence of an ordering \leq on states used to determine which inferences are drawn about the effects of actions. In some of these approaches [Winslett, 1988; Katsuno and Mendelzon, 1992; Sandewall, 1996] the ordering \leq is defined over the set $\mathcal{M}_{\mathcal{L}}$ of object states. For example, according to the Possible Models Approach (PMA), the ordering \leq_w associated with an (initial) state w is defined as follows: for any $r, r' \in \mathcal{M}_{\mathcal{L}}$, $r \leq_w r'$ if and only if $Diff(w, r) \subseteq Diff(w, r')$.³ There are however many theories of action for which the ordering \leq is defined, not over the set of object states, but rather over an extended set of *meta-states*. Consider, for example, a theory of action based on circumscription [McCarthy, 1980]. Circumscription's minimisation policy induces an ordering \leq that is defined over a set of meta-states $\mathcal{M}_{\mathcal{L}'}$, generated from the set of object states $\mathcal{M}_{\mathcal{L}}$ with the addition of the *abnormality predicate* Ab . More precisely, if the object language has n fluents, and m actions, there will be 2^n object states in $\mathcal{M}_{\mathcal{L}}$; with the addition of the abnormality predicate Ab , each object state w "splits" into $2^{n \times m}$ meta-states, all of which agree with w on the truth value of the n (object) fluents, and differ only on the value of the abnormality predicate Ab for each pair of (object) fluent and action. Thus there will be a total of $2^{n+(n \times m)}$ meta-states over which the ordering \leq is defined.

As we prove later in this paper, moving the minimisation policy from object states to meta-states results in significant gains in the range of applicability of minimal change approaches. Given the major role of meta-states in our study, in the rest of this section we introduce some further notation and formally define the concepts related to meta-states.

A propositional language \mathcal{L}' is called an *extension* of \mathcal{L} if and only if firstly, \mathcal{L}' is *finitary* and secondly, the propositional variables of \mathcal{L} are included in \mathcal{L}' . If \mathcal{L}' is an extension of \mathcal{L} we shall refer to the additional propositional variables of \mathcal{L}' (i.e., those that do not appear in \mathcal{L}) as *control variables* or *control fluents*.⁴ We shall say that \mathcal{L}' is a k -extension of \mathcal{L} , for a natural number $k \in \mathbb{N}$, if and only if \mathcal{L}' is an extension of \mathcal{L} and it contains precisely k control fluents. Clearly, any 0-extension of \mathcal{L} is identical with \mathcal{L} .

For an extension \mathcal{L}' of \mathcal{L} , any maximally consistent set of literals of \mathcal{L}' is called a *meta-state*. For a set of sentences G of \mathcal{L}' , we define the *restriction of G to \mathcal{L}* , denoted G/\mathcal{L} , to be the set $G \cap \mathcal{L}$. Finally, we define the restriction to \mathcal{L} of a collection V of sets of sentences of \mathcal{L}' , denoted V/\mathcal{L} , to be the set consisting of the restriction to \mathcal{L} of the elements of V ; in symbols, $V/\mathcal{L}' = \{G/\mathcal{L} : G \in V\}$.

2.3 Preferential Models

Having formally defined meta-states it remains to introduce a general model that encodes the concept of minimisation over meta-states.

Definition 2.2 A preferential structure for \mathcal{L} is a triple $\mathcal{U} = \langle \mathcal{L}', \mathcal{S}', \theta \rangle$ where:

³For any two states w, z , $Diff(w, z)$ denotes the symmetric difference of w and z .

⁴Like the abnormality predicate, control fluents are meant to be variables guiding the minimisation policy.

- \mathcal{L}' is an extension of \mathcal{L} .
- \mathcal{S}' is a nonempty collection of maximally consistent sets of literals of \mathcal{L}' ; we shall call the elements of \mathcal{S}' valid meta-states.
- θ is a function mapping each object state $w \in \mathcal{M}_{\mathcal{L}}$ to a (partial) preorder over $\mathcal{M}_{\mathcal{L}'}$ (the set of all maximally consistent sets of literals of \mathcal{L}'); we shall denote the preorder assigned to w , by \leq_w .⁵

As mentioned earlier, a preferential structure $\mathcal{U} = \langle \mathcal{L}', \mathcal{S}', \theta \rangle$ is meant to be the basis for encoding formally (and in a quite abstract manner) the concept of minimal change. More precisely, let $w \in \mathcal{M}_{\mathcal{L}}$ be any object state. The preorder \leq_w associated with w represents the *comparative similarity* of meta-states to w . Using \leq_w (and the principle of minimal change), one can then determine the states $\mathcal{R}(w, \alpha)$ that can possibly result from the application of an action α to w by means of the condition (M) given below:

$$(M) \quad \mathcal{R}(w, \alpha) = (\min([\alpha]_{\mathcal{L}'}, <_w) \cap \mathcal{S}') / \mathcal{L}.$$

In the above condition, $[\alpha]_{\mathcal{L}'}$ denotes the set of meta-states consistent with the sentence α and $\min([\alpha]_{\mathcal{L}'}, <_w)$ is the set of such meta-states that are minimal (“most preferred”) with respect to $<_w$.

The intuition behind condition (M) should be clear. Essentially, we select those meta-states consistent with formula α (representing the postcondition of an action) that are minimal under the ordering $<_w$, filter out the valid ones and then restrict these to the language of the dynamic system under consideration.

We shall say that a preferential structure $\mathcal{U} = \langle \mathcal{L}', \mathcal{S}', \theta \rangle$ is a *preferential model* for the dynamic system $W = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ if its result function can be reproduced from \mathcal{U} by means of condition (M); more precisely, if and only if for all $w \in \mathcal{S}$ and $\alpha \in \mathcal{A}$, condition (M) is satisfied. If a dynamic system W has a preferential model \mathcal{M} , we shall say that W is *minimisable*; moreover, if there are precisely k control fluents in \mathcal{L}' , we shall say that W is *k-minimisable* or that it has a preferential model with *degree k*. Clearly, if a dynamic system W has a preferential model with degree k for some $k \in \mathbb{N}$ it also has a preferential model with degree m for any $m \geq k$.

3 Minimisable Dynamic Systems

Our aim in this section is to provide a characterisation of the class of minimisable dynamic systems, in terms of conditions imposed on the result function \mathcal{R} .

Let $W = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ be a dynamic system, w a state in \mathcal{S} , and α, α' any two actions in \mathcal{A} . Consider the conditions (P1) – (P3) below:

- (P1) If $w \in \mathcal{S}$ and $r \in \mathcal{R}(w, \alpha)$, then $r \vdash \alpha$
- (P2) If $w \in \mathcal{S}$ and $\vdash \alpha \leftrightarrow \alpha'$, then $\mathcal{R}(w, \alpha) = \mathcal{R}(w, \alpha')$
- (P3) If $\alpha \vdash \alpha'$, $w \in \mathcal{S}$, $r \in \mathcal{R}(w, \alpha')$ and $r \in [\alpha]$, then $r \in \mathcal{R}(w, \alpha)$

⁵We shall also use $<_w$ to refer to the strict (non-reflexive) part of \leq_w .

These conditions can be interpreted quite simply. Condition (P1) says that the postconditions of an action (i.e., α) should be true at all possible resultant states. Condition (P2) is an *irrelevance of syntax* condition stating that actions having logically equivalent postconditions should predict the same resultant states. Condition (P3) states that if a state r is chosen as a possible outcome of an action α' , then r should also be chosen as a possible outcome of any action α which is stronger than α' and consistent with r . (NB: $\alpha \vdash \alpha'$ implies $[\alpha] \subseteq [\alpha']$). This last condition is similar to the choice theoretic condition known as (α) in the literature [Sen, 1977]. These three, simply stated conditions suffice to exactly characterise the class of minimisable dynamic systems.

Theorem 3.1 *A dynamic system is minimisable if and only if it satisfies the conditions (P1) – (P3).*

Theorem 3.1 is the central result of this article. What is perhaps surprising about this theorem is that it manages to provide a characterisation of minimality defined over *meta-states* via conditions on the result function, which operates on *object states*.

The proof of Theorem 3.1 is omitted due to space limitations. The most interesting part of the proof is a construction that, given a dynamic system W with n fluents satisfying (P1) – (P3), generates a preferential model for W with degree 2^n . An immediate corollary of this is that, if a dynamic system is at all minimisable, then it is minimisable with degree 2^n . We shall call the *smallest* number k for which a minimisable dynamic system W has a preferential model with degree k , the *minimality rank* of W . As already mentioned, the corollary below follows directly from the proof of Theorem 3.1.

Corollary 3.1 *The minimality rank of a minimisable dynamic system with n fluents is no greater than 2^n .*

Having provided a general characterisation of minimisable dynamic systems by means of (P1) – (P3), we shall now turn to special cases. More precisely, we shall impose certain constraints on preferential models and examine their implications on minimisable dynamic systems via condition (M).

The first such constraint is *totality* on the preorders of a preferential model. More precisely, we shall say that a preferential model $\mathcal{U} = \langle \mathcal{L}', \mathcal{S}', \theta \rangle$ is *linear* if and only if all preorders in θ are *total* (sometimes referred to as *connected*). We shall say that a dynamic system $W = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ is *strictly minimisable* if and only if W has a linear preferential model. In the presence of (P1) – (P3), condition (P4) below characterises precisely the class of strictly minimisable dynamic systems. The term $\mathcal{R}(w, \mathcal{A}')$ in (P4), where \mathcal{A}' is a subset of \mathcal{A} , is used as an abbreviation for $\bigcup_{\alpha' \in \mathcal{A}'} \mathcal{R}(w, \alpha')$.

- (P4) For any nonempty $\mathcal{A}' \subseteq \mathcal{A}$ such that $\mathcal{R}(w, \alpha') \neq \emptyset$ for all $\alpha' \in \mathcal{A}'$, there exists a nonempty subset \mathcal{B} of $\mathcal{R}(w, \mathcal{A}')$, such that $\mathcal{R}(w, \alpha) = [\alpha] \cap \mathcal{B}$, for all $\alpha \in \mathcal{A}$ such that $[\alpha] \subseteq [\mathcal{A}']$ and $[\alpha] \cap \mathcal{B} \neq \emptyset$.

Condition (P4) essentially says that, under certain conditions, a collection of states V contains a subset \mathcal{B} of “best” elements. Consequently, whenever an action α is such that all α -states are contained in V , and moreover, among the α -states there are some of the “best” elements of V , then any state resulting from α is among those “best” α -states (i.e., $\mathcal{R}(w, \alpha) =$

$[\alpha] \cap \beta$). Notice that (P4) collapses to the (much more familiar) condition (P4') below, whenever $\mathcal{R}(w, \alpha)$ is defined for all pairs of states w , and sentences α . In the principal case however where \mathcal{R} is defined over a proper subset of $\mathcal{M}_{\mathcal{L}} \times \mathcal{L}$, (P4') is strictly weaker than (P4).

(P4') If $\mathcal{R}(w, \alpha), \mathcal{R}(w, \alpha') \neq \emptyset, \mathcal{R}(w, \alpha) \subseteq [\alpha']$, and $\mathcal{R}(w, \alpha') \subseteq [\alpha]$, then $\mathcal{R}(w, \alpha) = \mathcal{R}(w, \alpha')$.

This is essentially the (U6) postulate of Katsuno and Mendelzon [1992]. It is also found as property (3.13) in Gärdenfors [1988, p. 57].

Theorem 3.2 *A dynamic system is strictly minimisable if and only if it satisfies the conditions (P1) – (P4).*

A direct consequence of the (only-if part of the) above proof is the following corollary.

Corollary 3.2 *The minimality rank of a strictly minimisable dynamic system is no greater than 1.*

Corollary 3.2 shows that by imposing totality on the preorders of the preferential model, we get very close to zero-minimisable dynamic systems. Very close indeed, but not quite there. In this paper we do not provide a complete characterisation of zero-minimisable dynamic systems as it is not central to our aims here; we do however provide some preliminary results in this direction. More precisely, consider the conditions (P5) – (P7) below (we implicitly assume that $w \in \mathcal{S}$ in each case):

(P5) If $([\alpha] - \mathcal{S}) \cup \mathcal{R}(w, \alpha) \subseteq [\alpha']$, then $[\alpha] \cap \mathcal{R}(w, \alpha') \subseteq \mathcal{R}(w, \alpha)$.

(P6) If $[\alpha] \subseteq \mathcal{S}$, then $\mathcal{R}(w, \alpha) \neq \emptyset$.

(P7) $\mathcal{R}(w, \alpha) \cap \mathcal{R}(w, \alpha') \subseteq \mathcal{R}(w, \alpha \vee \alpha')$

Condition (P5) says that if all non-valid α -states together with those α -states “chosen” by the result function are compatible with another action’s postconditions (α'), then any α -states chosen when performing α should also be chosen when performing α' . Notice that this condition implies condition (P3) above. (P6) states that the result function must return at least one possible resultant state if all states satisfying the postconditions of the action are valid. (P7) says that if a state r is chosen as a possible next state when either α or α' is performed, then r should also be chosen when the action (with postcondition) $\alpha \vee \alpha'$ is performed.

Theorem 3.3 *Every zero-minimisable dynamic system satisfies the conditions (P1) – (P3), (P5) – (P7).*

The converse of Theorem 3.3 is not true in general. However, for a restricted class of dynamic systems, which we call *dense*, the conditions (P1) – (P3) and (P5) – (P7) suffice to characterise zero-minimisability. More precisely, we shall say that a dynamic system is *dense* if and only if every sentence of the object language \mathcal{L} corresponds to an action (i.e., $\mathcal{A} = \mathcal{L}$).

Theorem 3.4 *If a dense dynamic system satisfies (P1) – (P3), (P5) – (P7), then it is zero-minimisable.*

We conclude this section with a brief comment on previous frameworks that encode the concept of minimal change. Perhaps the framework most closely related to our own comes from the area of *theory change* and it is the one developed by Katsuno and Mendelzon [1992] for modelling *belief update*. We shall leave a detailed comparison between the conditions presented herein and the postulates proposed by Katsuno and Mendelzon (known as the *KM postulates*) for future work. Here we simply note that the main difference between the two is that the KM postulates are designed to apply *only* to dense, zero-minimisable, dynamic systems.

4 Causality and Minimal Change

Recall that one of the main motivations for this work was the desire to formally evaluate claims about the strength of causal theories of action over ones based on the notion of minimal change. In this section we use the foregoing results to analyse two of the most prominent causal theories of action, the first developed by McCain and Turner [1995], and the second by Thielscher [1997].

4.1 McCain and Turner

McCain and Turner [1995] have developed a theory of action that represents causality explicitly. In their framework they introduce a causal connective \Rightarrow where $\phi \Rightarrow \psi$ can be read as “ ϕ causes ψ ” and referred to as a *causal rule*. Here ϕ and ψ are propositional sentences that do not contain \Rightarrow (i.e., \Rightarrow cannot be nested). They then introduce the notion of *causal closure* $C_{\mathcal{D}}(\Gamma)$ for a set of sentences Γ with respect to a set of causal rules \mathcal{D} as the smallest set closed under classical deduction that includes Γ and applies causal rules in the direction of the “arrow” (i.e., no contrapositive—the interested reader is referred to the citation above for the full details). The notation $\vdash_{\mathcal{D}}$ refers to the corresponding (causal) consequence relation: $\Gamma \vdash_{\mathcal{D}} \gamma$ if and only if $\gamma \in C_{\mathcal{D}}(\Gamma)$. The result function can be defined using the following fixed-point equation to be found in McCain and Turner [1995].

$$(MT) \quad r \in \mathcal{R}_{\mathcal{D}}^{MT}(w, \alpha) \text{ iff } r = \{\rho \in \mathcal{Z}_{\mathcal{L}} : (w \cap r) \cup \{\alpha\} \vdash_{\mathcal{D}} \rho\}$$

We can now establish the following results.

Theorem 4.1 *For any set of causal rules \mathcal{D} , the result function $\mathcal{R}_{\mathcal{D}}^{MT}(w, \alpha)$ defined by means of (MT), satisfies the conditions (P1) – (P3).*

From Theorem 4.1 it follows that the theory of action developed by McCain and Turner is applicable *only* to minimisable dynamic systems. This is a most curious result for it shows that the conclusions drawn with the aid of causality (as encoded by McCain and Turner) can be reproduced by an appropriately defined minimisation policy. It is also especially interesting in light of recent results reported by Peppas *et al.* [1999] who show that for McCain and Turner’s approach it is in general not possible to construct an ordering over object states such that the minimal object states satisfying the postconditions of an action are those predicted by the McCain and Turner fixed-point equation. This tells us that this approach is not zero-minimisable. We have, however,

just shown that this approach is minimisable in a more general sense (i.e., if one considers meta-states) so this system has a minimality rank greater than zero.

4.2 Thielscher

We shall not describe Thielscher's [1997] approach in depth here. However, the underlying principle is to consider trajectories of state-effect pairs each of which is the result of applying a causal law at a previous state-effect pair (starting with the initial state and action postcondition). The resultant states are those at the end of a trajectory where causal laws no longer apply. We note that Thielscher's system does not satisfy postulates (P1) – (P3) and hence is not minimisable. In particular, Thielscher's result function, which we denote by \mathcal{R}^T violates condition (P1); the postconditions of the action do not necessarily have to hold after applying the action. What would be of some interest however, is to generate from \mathcal{R}^T a new result function \mathcal{R}' that chooses among the resultant states selected by \mathcal{R}^T the ones that satisfy the postconditions of the occurring action; in symbols, $\mathcal{R}' = [\alpha] \cap \mathcal{R}^T$. One can then examine whether the new function \mathcal{R}' satisfies the conditions (P2) and (P3). We leave this for future investigation.

5 Discussion

Let us take a step back and examine what we have accomplished thus far. The main result reported herein is a characterisation of the class of dynamic systems amenable to minimisation, in terms of conditions on the result function. Existing causal theories of actions can then be assessed against these conditions to determine the added value (if any) of explicit representations of causality. This is clearly a significant step towards “demystifying” the (comparative) strengths and weaknesses of the notions of causality and minimal change in reasoning about action. Admittedly though in this paper we have not given the whole story (especially as far as “demystifying” causality is concerned). What is missing is a generic model of the use of the concept of causality in reasoning about action (in the same way that preferential structures are such a model for the concept of minimality), based on which a general, formal comparison between causality and minimality can be made. Until such a generic model is available, the best that can be done is evaluations of *specific* causal theories of action (such as the ones by McCain and Turner [1995] or Thielscher [1997]) against the conditions (P1) – (P3). There is of course still a lot of value in such assessments; showing for example that all existing causal approaches satisfy these conditions would be strong evidence in support of the claim that minimality subsumes causality (at least as far as the range of applicability is concerned). Showing, on the other hand, that some causal approach violates one of the conditions (P1) – (P3), would prove that causality is essential in reasoning about action since it covers domains that are “unreachable” by minimal change approaches. Similar (although weaker) conclusions can be drawn from the satisfaction or violation of (P4) and (P5) – (P7). We have already witnessed that Thielscher's [1997] approach does not satisfy condition (P1). Thus some causal approaches do indeed go beyond what is possible with minimal change. More work needs to be done

here to properly classify causal approaches both in regard to each other (with respect to the different causal notions they employ) and also with regard to the Principle of Minimal Change.

It should be noted, that when assessing a theory of action, apart from its range of applicability, a second criterion that is equally important is the *conciseness* of its representations (a solution to the *frame problem* ought to be both *correct* and *concise*). In this article, conciseness has been left out of the picture altogether. What we have mainly done herein was to axiomatically characterise certain classes of dynamic systems whose result function \mathcal{R} can be reproduced in terms of preorders \leq_w on states. No consideration was given as to whether \leq_w can be represented concisely or not. As far as our results are concerned, describing \leq_w could be as “expensive” as listing the frame axioms corresponding to \mathcal{R} , which of course defeats the whole purpose of using minimality. A much more useful result (for practical purposes) would be one that characterises the class of what we might call *concisely* minimisable dynamic systems; that is, dynamic systems whose result function \mathcal{R} can be reproduced by preorders \leq_w , which in turn can be represented concisely.⁶

Similar considerations apply to causality. Characterising the class of dynamic systems for which causality (in one form or another) can duplicate the result function, although an interesting theoretical result, would not fully address the issue of the applicability of causality in reasoning about action. Further work would be required to identify the domains that are amenable to concise causal descriptions.

Notice, also, that in cases where the range of applicability does not differentiate between causal and minimality-based approaches, conciseness considerations may well favour one over the other. More precisely, if the class of domains at focus is within the range of applicability of both causal and minimal change approaches, the determining factor in choosing between the two could be the “information cost” associated with the usage of each approach.

Notice that, while the concept of minimality has typically been used in the literature to deal with the frame and ramification problems, the nature of condition (M) is such that it requires minimality to deal with the *qualification problem* as well. Indeed, via condition (M), the preorders of a preferential model are used, not only to determine the set of states $\mathcal{R}(w, \alpha)$ that result from the application of an action α at an initial state w (frame and ramification problems), but also to determine whether α is at all *applicable* at w (qualification problem). The latter is decided based on whether $\mathcal{R}(w, \alpha)$ is the empty set or not (cf. McCain and Turner [1995] who claim this is a *derived qualification*). One could argue that such an additional burden is perhaps too much from minimality to carry (or that it is even counter-intuitive), and maybe, if liberated from it, minimality could be used in many more situations. More formally, consider the condition (M') below:

⁶It should be noted that all existing approaches that are based on the concept of minimal change are indeed of that nature; that is, their preorders on states have a concise description, typically in the form of a second-order axiom (sometimes coupled with limited domain-specific information).

(M') If $\mathcal{R}(w, \alpha) \neq \emptyset$, then $\mathcal{R}(w, \alpha) = (\min([\alpha]_{\mathcal{L}'}, \leq_w) \cap \mathcal{S}') / \mathcal{L}$.

Clearly (M') is weaker than (M). In fact, it is exactly the weakening of (M) that is needed to disengage minimality from the qualification problem. It would be a worthwhile exercise to reproduce the results of this article, having replaced (M) with (M'). The class of minimisable systems could be larger, and moreover, we conjecture that under (M'), strictly minimisable systems are a proper subclass of zero-minimisable dynamic systems.

We conclude this section with a remark on minimality ranks. Preliminary considerations suggest that there is a close relation between the minimality rank of a dynamic system on the one hand, and its ontological properties on the other. For example, domains where actions have no ramifications, tend to have lower minimality ranks than domains where ramifications do appear. If this connection can be generalised and formally proved, the minimality rank of a domain could serve as a precise measure of its complexity. More work however needs to be done in this direction.

6 Conclusions

We have developed a formal framework within which we were able to formulate necessary and sufficient conditions under which a dynamic system is minimisable; that is, its result function can be reproduced by an appropriately defined minimisation policy. What is particularly pleasing about these conditions is that they are few in number and relatively easy and intuitive to understand. Our original motivation in this study was to answer the question as to whether recently proposed theories of action involving explicit representations of causality are capable of forms of reasoning not possible via minimal change. We have seen that this is not the case with some approaches (McCain and Turner [1995]) but in others (Thielscher [1997]) the causal reasoning cannot be characterised by the Principle of Minimal Change. Perhaps of wider significance is the fact that the results reported here clearly indicate the range of applicability of the Principle of Minimal Change; one simply needs to verify three properties (viz. (P1) – (P3)).

This work opens up many interesting avenues for future work, various of which were mentioned in the discussion. One of the more pressing is to consider a variety of theories of action, particularly causal ones, and verify which properties they satisfy. This task is well under way and reserved for a much lengthier work. Also of much interest would be to further categorise levels of minimisability and what distinguishes them together with the various theories of action at those levels of minimisability.

Acknowledgments

The authors would like to thank three anonymous referees for their inciteful and generous comments. They would also like to thank Norman Foo, Abhaya Nayak, Mikhail Prokopenko, members of the Cognitive Robotics Group at the University of Toronto, and attendees of the Seminar in Applications of Logic at the City University of New York for enlightening discussions.

References

- [Doherty, 1994] P. Doherty. Reasoning about Action and Change using Occlusion. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pp. 401–405, 1994.
- [Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux*. The MIT Press, Cambridge, MA, 1988.
- [Katsuno and Mendelzon, 1992] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In P. Gärdenfors, ed., *Belief Revision*, pp. 183–203, 1992.
- [Krautz, 1986] A. Krautz. The logic of persistence. in *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 401-405, 1986.
- [Lifschitz, 1990] V. Lifschitz. Frames in the space of situations, *Artificial Intelligence*, **46**:365–376, 1990.
- [Lin, 1995] F. Lin. Embracing causality in specifying the indeterminate effects of actions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 2001–2007, 1995.
- [McCain and Turner, 1995] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1978–1984. Montreal, 1995.
- [McCain and Turner, 1997] N. McCain and H. Turner. Causal theories of action and change. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 460-465, 1997.
- [McCarthy, 1980] J. McCarthy. Circumscription—A form of nonmonotonic reasoning. *Artificial Intelligence*, **13**:27–39, 1980.
- [Peppas et al., 1999] P. Peppas, M. Pagnucco, M. Prokopenko, N. Foo and A. Nayak. Preferential semantics for causal system. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 119–123, 1999.
- [Sandewall, 1989] E. Sandewall. Filter preferential entailment for the logic of action in almost continuous worlds. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.
- [Sandewall, 1994] E. Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems. Vol. 1*. Oxford University Press, 1994.
- [Sandewall, 1996] E. Sandewall. Assessments of ramification methods that use static domain constraints. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning*. 1996.
- [Sen, 1977] A. Sen, Social choice theory: A re-examination, *Econometrica*, **45**:53–89, 1977.
- [Shoham, 1988] Y. Shoham. *Reasoning About Change*. MIT Press, Cambridge, Massachusetts, 1988.
- [Thielscher, 1997] M. Thielscher. Ramification and causality. *Artificial Intelligence*, **89**:317–364, 1997.
- [Winslett, 1988] M. Winslett. Reasoning about actions using a possible models approach. In *Proceedings of the Seventh National Artificial Intelligence Conference*, 1988.

EPDL: A Logic for Causal Reasoning

Dongmo Zhang and Norman Foo

Knowledge Systems Group
School of Computer Science and Engineering
The University of New South Wales, Australia
{dongmo, norman}@cse.unsw.edu.au

Abstract

This paper presents an extended system *EPDL* of propositional dynamic logic by allowing a proposition as a modality for representing and specifying direct and indirect effects of actions in a unified logical structure. A set of causal logics based on the framework are proposed to model causal propagations through logical relevancy and iterated effects of causation. It is shown that these logics capture the basic properties of causal reasoning.

Keywords: causality, reasoning about actions, common-sense reasoning

1 Introduction

Causality plays a basic role in human reasoning. The philosophical study of causality dates back to the origin of the discipline. AI research in causality finds its impetus in considerations from commonsense reasoning, especially the ramification problem. It has been found that propagations of effects cannot be appropriately specified by domain constraints (a set of pure logical expressions) ([Lin 1995;McCain and Tuner 1995;Thielscher 1997]). Not only the causation between actions and fluents but also the causation between fluents has to be explicitly represented and specified. It is obvious that these two kinds of causation bear some essential similarities and so might be formalized with a similar logical structure. This is of importance not just in ontology. From the methodological point of view, we might take advantage of such a similarity to propose a provably correct theory of causality. We have both a good understanding of, and formalisms for the direct effects of actions. However, causal propagation is often confused with logical relevancy and persistence of fluents.

In this paper, we introduce a formalism for representing and specifying both the direct and indirect effects of actions in a unified logical structure. Our framework is based on dynamic logic. We do so not only because the dynamic logic is one of the typical formalisms for reasoning about action (c.f. the preface of [Harel 1979]) but also because the modal expression of causal relation is one of the main approaches to causation in philosophy ([Brand 1976]) and in AI ([McCain and Turner 1995;Giordano *et al.* 2000]).

In dynamic logic, causation between an action and its effects is expressed by the modal formula $[\alpha]A$, where α is a

(primitive or compound) action and A is a property. For instance, $[Shoot]\neg alive$ is read as “the action *Shoot* causes a turkey to be dead”. In many cases, the effects of an action can be further propagated through causal relations between fluents. For instance, if a turkey is shot down, its death will cause the turkey to be unable to walk: $\neg alive$ causes $\neg walking$. However, this cannot be formally expressed in dynamic logic. Obviously it cannot simply be written as either $\neg alive \rightarrow \neg walking$ or $[\neg alive]\neg walking$. The first expression is obviously unsuitable because of the directionality of causation. The second one is not allowed in the syntax of dynamic logic. Therefore, our idea for expressing indirect effects of actions in dynamic logic is to extend the traditional dynamic logic language by allowing a proposition as a modality, say $[\varphi]$ for proposition φ . If we allow a proposition as a modality¹, we will have a way to unify the treatment of the causation between action and proposition and between propositions, likewise the treatment of direct and indirect effects of actions.

In this paper, we first present such an extended system of the propositional dynamic logic (*PDL*). The system serves as a formal platform for causal reasoning on which a set of causal logics are constructed to show how to build a causal logic satisfying our intuitions of causal reasoning with different domains. An inference mechanism on action descriptions is introduced to facilitate reasoning with action laws and causal laws. Finally, an application of our approach to the ramification problem is briefly presented and the computational complexity of some causal logics is discussed.

2 Extended Propositional Dynamic Logic

In this section, we extend the *PDL* by introducing propositional modalities, relative axioms and the intended semantics. The extended system is called *EPDL*.

2.1 Language of *EPDL*

The alphabet of the language \mathcal{L}_{EPDL} of the extended propositional dynamic logic consists of a countable set **Flu** of fluent symbols and a countable set **Act_P** of primitive action symbols.

¹A similar approach has been used by [Ryan and Schobbens 1997] to model knowledge updates and counterfactuals. Besides the motivation, however, the semantics and deduction system are quite different from ours.

Propositions ($\varphi \in \mathbf{Pro}$), formulas ($A \in \mathbf{Fma}$) and actions ($\alpha \in \mathbf{Act}$) are defined by the following BNF rules:

$$\begin{aligned} \varphi &::= f \mid \neg\varphi \mid \varphi_1 \rightarrow \varphi_2 \\ A &::= f \mid \neg A \mid A_1 \rightarrow A_2 \mid [\alpha]A \mid [\varphi]A \\ \alpha &::= a \mid \alpha_1; \alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \alpha^* \mid A? \\ &\text{where } f \in \mathbf{Flu} \text{ and } a \in \mathbf{Act}_P. \end{aligned}$$

The intended meaning for $[\alpha]A$ is “ α (always) causes A if α is executable”. The dual operator $\langle \alpha \rangle A$, defined as usual, reads as “ α is executable and possibly (or may) cause(s) A ”. For instance, $\langle \text{Spin} \rangle \neg \text{loaded}$ says that “spinning a gun may cause it to be unloaded”. $\langle \alpha \rangle \top$ means “ α is executable”.

The formula $[\varphi]A$, where φ is a proposition, represents *propositional causation*, reading as “ φ causes A ”. For example, $[\text{short-circuit}] \text{damaged}$ says that “a short-circuit causes the circuit to be damaged”.

Notice the difference between $[\varphi]$ and $[\varphi?]$. $\varphi?$ is an action, called *test action*, which can be compounded with other actions but φ can not be compounded with actions.

2.2 Semantics

The semantics for *EPDL* is the standard semantics for *PDL* plus the interpretation of propositional causation.

As usual, a model for *EPDL* is a structure $M = (W, \mathcal{R}, V)$, where $\mathcal{R} = \{R_\alpha : \alpha \in \mathbf{Act}\} \cup \{R_\varphi : \varphi \in \mathbf{Pro}\}$. The meanings of W , $\{R_\alpha : \alpha \in \mathbf{Act}\}$ and V are the same as the ones for a *PDL* model. For any proposition φ , R_φ , similar to R_α , is a binary relation on W .

The satisfaction relation $M \models_s A$ can be defined as usual. For instance:

$M \models_w [\gamma]A$ iff for any $(w, w') \in R_\gamma$, $M \models_{w'} A$, where $\gamma \in \mathbf{Act} \cup \mathbf{Pro}$.

As in any modal logic, $A \in \mathbf{Fma}$ is valid in M , written as $M \models A$, if $M \models_w A$ for all $w \in W$. $\models A$ means that A is valid in all models.

The conditions of *standard models* for program connectives are as usual. For instance, the condition for test action is: “ $(w, w) \in R_{A?}$ iff $M \models_w A$ ”.

The conditions of standard models for propositional causation are as follows:

- C_{CW} : If $M \models_w \varphi$, then $(w, w) \in R_\varphi$.
- C_{CE} : If $\models \varphi_1 \leftrightarrow \varphi_2$, then $R_{\varphi_1} = R_{\varphi_2}$.

The condition C_{CW} is one half of the test action semantics. It implies $R_{\varphi?} \subseteq R_\varphi$. So $\models [\varphi]A \rightarrow [\varphi?]A$. Note that $\models [\varphi?]A \leftrightarrow (\varphi \rightarrow A)$. Thus $\models [\varphi]A \rightarrow (\varphi \rightarrow A)$. This reflects the essential difference between the causal relation and the material implication. The difference between them is obvious. Semantically, to verify $\varphi \rightarrow A$, we only need to check the truth-values of φ and A in the current world whereas to verify $[\varphi]A$, not only the current world but also other related worlds have to be considered. In other words, a causal relation is action-relevant in the way that it is sensitive to both the history and the future evolution of the system under consideration.

The condition C_{CE} shows that propositional causations we consider are syntax-independent. Although it is not necessarily true in practical reasoning, it is certainly a way to make a causal theory simple.

2.3 Deductive system

The axiomatic system for *EPDL* consists of the following axiom schemes and inference rules:

(1). Axiom schemes:

- all tautologies of propositional calculus.
- all axioms for compound programs².
- EK : $\vdash [\gamma](A \rightarrow B) \rightarrow ([\gamma]A \rightarrow [\gamma]B)$ (Extended K)
- CW : $\vdash [\varphi]A \rightarrow [\varphi?]A$ (Causal Weakening)

(2). Inference rules:

• MP : From $\vdash A$ and $\vdash A \rightarrow B$, infer $\vdash B$. (Modus Ponens)

• EN : From $\vdash A$ infer $\vdash [\gamma]A$. (Extended Necessitation)

• CE : From $\vdash \varphi \leftrightarrow \psi$ infer $\vdash [\varphi]A \leftrightarrow [\psi]A$. (Cause Equivalence)

where $\varphi, \varphi_1, \varphi_2 \in \mathbf{Pro}$, $A \in \mathbf{Fma}$, $\alpha \in \mathbf{Act}$ and $\gamma \in \mathbf{Pro} \cup \mathbf{Act}$.

EK and EN are the extensions of the Axiom K and Necessitation N of *PDL*, respectively. They convey the message that propositional causation $[\varphi]$ will be treated as the standard dynamic modality. The EK specifies propagation of causations under logical implication. The inference rule EN says that tautologies can always be caused.

The axiom CW and the inference rule CE are new, specifying propositional causation. CW corresponds to the semantic condition C_{CW} , which reflects the standard way to find and judge a causal relation. The rule CE ($RCEC$ in conditional logic ([Nute 1984])) exactly corresponds to the semantic condition C_{CE} .

Note that if we add an extra axiom “ $[\varphi?]A \rightarrow [\varphi]A$ ” into the deductive system, *EPDL* will collapse to *PDL* (precisely, *PDL* plus a duplicate of each test action, denoted by PDL^+). So the consistency of this axiomatic system is obvious.

Following the standard technique of using canonical models and filtration, we can prove that the deductive system for *EPDL* is sound and complete.

Theorem 1 $\vdash A$ if and only if $\models A$.

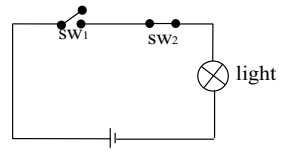
EPDL also preserves the finite model property, so it is decidable.

2.4 Reasoning with action descriptions

In reasoning about a dynamic system, we generally specify the system by a set of domain-dependent axioms to describe preconditions, qualifications and effects of actions, causal relations and other domain constraints. Such a set is called an *action description* or *domain description*, which can be any finite set of *EPDL* formulas.

Example 1 Consider the circuit of the following figure. Let $\mathbf{Flu} = \{sw_1, sw_2, light\}$ and $\mathbf{Act}_P = \{Toggle_1, Toggle_2\}$. Then the circuit can be specified by the following action description:

$$\Sigma = \left\{ \begin{array}{l} \neg sw_i \rightarrow [Toggle_i]sw_i \\ sw_i \rightarrow [Toggle_i]\neg sw_i \\ [sw_1 \wedge sw_2]light \\ [\neg sw_1 \vee \neg sw_2]\neg light \\ \langle Toggle_i \rangle \top \\ i = 1, 2 \end{array} \right\}$$



²See [Goldblatt 1987] or any other dynamic logic book.

The first two formulas describe the direct effects of action $Toggle_1$ and $Toggle_2$. The middle two specify the causal relations among the fluents in the circuit. “ $\langle Toggle_i \rangle \top$ ” means the action $Toggle_i$ is always executable. \square

We can easily see that a formula in an action description is different from an ordinary formula. For instance, the sentence “ $sw_1 \rightarrow [Toggle_1] \neg sw_1$ ” in an action description means that “whenever switch 1 is on, toggling it will cause it to be off”. In situation calculus language, it can be expressed as $\forall s (sw_1(s) \rightarrow \neg sw_1(result(Toggle_1, s)))$. A similar expression in dynamic logic is $[any](sw_1 \rightarrow [Toggle_1] \neg sw_1)$, where **any** is a special action, meaning “any action”. With the help of **[any]**, a sentence in an action description can be easily differentiated from an ordinary formula. In [Goranko and Passy 1992], it was shown that **[any]** can be an S_5 -modality. Therefore, if we want to introduce **any** formally as [Prendinger and Schurz 1996] and [Castilho et al. 1999] did, we must extend our system further by adding all the axioms of S_5 and an extra axiom “ $[any]A \rightarrow [a]A$ ”, stating that **any** is a universal action.³ Instead of doing this, however, we prefer a simpler way to deal with action description by treating it as a set of extra axioms (domain axioms).

Definition 1 Let Σ be an action description. A formula A is Σ -provable, written as $\vdash^\Sigma A$, if it belongs to the smallest set of formulas which contains all the theorems of $EPDL$, all the elements of Σ , and is closed under MP and EN .⁴

Example 2 Consider the action description in Example 1. We can easily prove the following consequences:

1. $\vdash^\Sigma (sw_1 \wedge sw_2) \leftrightarrow light$
2. $\vdash^\Sigma sw_1 \rightarrow [Toggle_1] \neg light$

The first expression shows that the causal laws “ $[sw_1 \wedge sw_2] light$ ” and “ $[\neg sw_1 \vee \neg sw_2] \neg light$ ” imply the domain constraint “ $(sw_1 \wedge sw_2) \leftrightarrow light$ ”. The second inference relation reflects the propagation of effects of action $Toggle_1$ through the causal law $[\neg sw_1 \vee \neg sw_2] \neg light$. \square

Note that the action description Σ is not complete. All the frame axioms were left out. For such an incomplete action description, a solution to the frame problem is necessary. Otherwise, we can not even derive the following very intuitive consequence:

$$\vdash^\Sigma \neg sw_1 \wedge \neg sw_2 \rightarrow [Toggle_1; Toggle_2] light$$

We will not include a solution to the frame problem in this paper. A solution to the frame problem based on $EPDL$ has been presented in a separated paper⁵ There are also several other PDL-based solutions to the problem in the literature

³This process is not necessary if the underlying language is finite. Additionally, [Castilho et al. 1999] treated **[any]** as S_4 -modality.

⁴Note that we do not require that Σ -provability is closed under CE . In fact, it is provable in the following sense:

$$\text{If } \vdash \varphi \leftrightarrow \psi, \text{ then } \vdash^\Sigma [\psi]A \leftrightarrow [\varphi]A.$$

⁵The basic idea of that paper is postponing the listing of frame axioms till a query arises in order to reduce the requirement of frame axioms. We proved that the required frame axioms for answering a query only depend on the objects involved in the query and possibly some objects in action description. The manuscript of the paper is available at <ftp://ftp.cse.unsw.edu.au/pub/users/ksg/Working/action3.zip>.

which can be easily embedded in $EPDL$ ([Prendinger and Schurz 1996; Castilho et al. 1999]).

A standard model M is called a Σ -model if $M \models B$ for any $B \in \Sigma$. A is Σ -valid, written by $\models^\Sigma A$, if it is valid in every Σ -model. Then we have

Theorem 2 (Soundness and completeness of Σ -provability)

$$\vdash^\Sigma A \text{ if and only if } \models^\Sigma A.$$

3 Logics for Causal Propagation

$EPDL$ provides a unified syntax and inference mechanism for reasoning about both direct effects and indirect effects of actions. It is a minor extension of the classical propositional dynamic logic.

We did not endow $EPDL$ with unlimited ability for causal reasoning. For instance, in Example 1, it is obvious that one of the switches being off will cause the light to be off. However, we cannot prove in $EPDL$ that $\vdash^\Sigma [\neg sw_1] \neg light$ or $\vdash^\Sigma [\neg sw_2] \neg light$. The reason for our decision is that causality is a relation sensitive to context, time, domains and so on. It seems impossible to have a universal causal logic to satisfy all kinds of applications. Instead, we introduce some optional inference laws for causality in order to form a flexible platform for causal reasoning.

Let us consider the following plausible laws of causality:

$$\begin{aligned} AND : & \vdash [\varphi]A \rightarrow [\varphi \wedge \psi]A \\ OR : & \vdash [\varphi]A \wedge [\psi]A \rightarrow [\varphi \vee \psi]A \\ Chn : & \vdash [\varphi \wedge \psi]A \rightarrow [\varphi][\psi]A \text{ (Chaining)} \\ PsC : & \vdash A \rightarrow [\varphi]A \text{ (Pseudo-Causation)} \end{aligned}$$

These laws are not hard to understand even though they are not necessarily widely acceptable. For instance, the first one says that if φ causes A , then it still causes A with any extra fact. The second one says that if individually φ and ψ cause A , having one will cause A .

Chn says that if φ and ψ are the causes of A , then these causes can be chained into a nested causation. For instance, in the circuit of Example 1, we could say that the closing of switch 1 causes that the closing of switch 2 causes the light to be on.

PsC says that everything can cause a true statement.

We can easily find examples in everyday reasoning to verify or refute any of these laws. For instance, on one hand by AND we can say that if *raining* causes *wet*, then *raining* and *shining* cause *wet* as well; on the other hand, AND implies the following intuitive inference rule in the presence of CE :

$$LC : \text{if } \vdash \varphi \rightarrow \psi, \text{ then } \vdash [\psi]A \rightarrow [\varphi]A.$$

This law, named as RCK in conditional logic, reflects the regularity of causal propagation through logical relevancy.

It can be proved in $EPDL$ that PsC implies the following law:

$$Clps : \vdash [\varphi][\psi]A \rightarrow [\varphi \wedge \psi]A \text{ (Collapsing)}$$

We call it the *collapsing of cause-effect chain*. Sometimes it is a good simplification for causal reasoning. However, it could be harmful if the cause-effect chain matters.

We can also find some pros and cons for the other laws. In fact, we do not intend to persuade people to believe in these laws but try to introduce a hierarchy of causal logics to meet

different requirements of causal reasoning. We refer to these logics as:

$$EPDL_1 = EPDL + AND$$

$$EPDL_2 = EPDL_1 + OR$$

$$EPDL_3 = EPDL_2 + Chn$$

$$EPDL_4 = EPDL_3 + PsC$$

Note that all the laws are independent. So we can combine them in any fashion. We suggest that in practice such laws should be recombined, or new axioms be introduced to form particular systems of causal reasoning for particular applications.⁶

The following properties are the corresponding conditions on standard models for the associated laws:

$$C_{AND} : R_{\varphi \wedge \psi} \subseteq R_{\varphi}$$

$$C_{OR} : R_{\varphi \vee \psi} \subseteq R_{\varphi} \cup R_{\psi}$$

$$C_{Chn} : R_{\varphi} \circ R_{\psi} \subseteq R_{\varphi \wedge \psi}$$

$$C_{PsC} : R_{\varphi} \subseteq R_{id}$$

where R_{id} is the identity relation.

Note that all conditions C_{CE} , C_{AND} , C_{OR} , C_{Chn} and C_{PsC} together imply:

$$1. R_{\varphi \vee \psi} = R_{\varphi} \cup R_{\psi}.$$

$$2. R_{\varphi \wedge \psi} = R_{\varphi} \cap R_{\psi} = R_{\varphi} \cap R_{\psi}$$

Therefore the conditions for $EPDL_4$ models force the accessibility relations of propositional causation to be a Boolean lattice on the subsets of the identity relation. Fortunately, the gap between causation and implication (or test action) still exists. So $EPDL_4$ does not collapse to PDL^+ .

Proposition 1 $\not\models_4 [\varphi]\varphi$. Therefore $\not\models_4 [\varphi?]A \rightarrow [\varphi]A$.

Proof. Let \mathcal{L} be a language of $EPDL$ in which $\mathbf{Flu} = \{f_1, f_2\}$ and $\mathbf{Act}_P = \{\}$. Let $M = (W, \mathcal{R}, V)$ be a standard model of $EPDL$, where $W = \{w, w'\}$, $R_{f_1} = \{(w, w), (w', w')\}$, $R_{\neg f_1} = \{(w, w)\}$, $R_{f_2} = \{(w, w)\}$, $R_{\neg f_2} = \{(w, w), (w', w')\}$, $V(f_1) = \{w'\}$, $V(f_2) = \{w\}$. Let $R_{\varphi \wedge \psi} = R_{\varphi} \cap R_{\psi}$ and $R_{\varphi \vee \psi} = R_{\varphi} \cup R_{\psi}$. Then M is an $EPDL_4$ model. We have $M \models_w [f_1?]f_1$, but $M \not\models_w [f_1]f_1$.



Accessibility relation for $[f_1?]$. Accessibility relation for $[f_1]$.

This proposition differentiates our approach to causality from the counterfactual-based approaches where *Self-Causation* is generally accepted.

The following theorem gives the soundness and completeness of each logic, where $\vdash_i (=_i)$ represents the syntactical (semantic) entailment in each logic.

Theorem 3 $\vdash_i A$ if and only if $\models_i A$ ($i = 1, \dots, 4$).

Decidability of each logic can be also proved in a similar way with $EPDL$. Σ -provability is readily extended into each logic.

⁶These logics form a chain between PDL and PDL^+ :

$$EPDL \prec EPDL_1 \prec EPDL_2 \prec EPDL_3 \prec EPDL_4 \prec PDL^+.$$

The ordering shows the degree of closeness between causation and implication in each logic.

4 Reasoning with Causal Laws

Two kinds of causal relations have been explicitly distinguished in the philosophy of causality: *causal laws* and *causal instances*. Roughly speaking, a causal relation is a causal law if it exists everywhere whereas a causal instance only exists in some particular worlds. Interestingly, most of our intuition about causal reasoning comes from reasoning with causal laws. For instance, *Transitivity* is generally accepted as a property of causal reasoning. However, if you are not an $EPDL_4$ reasoner, it does not hold for causal instances. More precisely, $[\varphi]\psi \wedge [\psi]\lambda \rightarrow [\varphi]\lambda$ is not a tautology in $EPDL_3$ or lower. The reason is: even if φ can cause ψ to be true in any accessible worlds and ψ can cause λ to be true in all its accessible worlds because ψ does not necessarily cause λ in φ 's accessible worlds. However, if we express *Transitivity* in the following form:

“If $[\varphi]\psi$, $[\psi]\chi \in \Sigma$, then $\vdash^\Sigma [\varphi]\chi$ ”,

it will be valid in any causal logic. This means that if both $[\varphi]\psi$ and $[\psi]\chi$ are causal laws, $[\varphi]\chi$ will be a derived causal law.

We call a formula $[\varphi]\psi$ a *causal law* if it is in an action description. A general question is: *Given a set Σ of causal laws, what other causal laws can be derived?*, or formally, *whether $\vdash^\Sigma [\varphi]\psi$?*

4.1 Properties of causal reasoning

Firstly, let's show some basic properties of causal reasoning with causal laws. The following theorem is a summary of properties of $EPDL$ and its extensions.

Theorem 4 Let Σ be any action description. Then

1. If $\vdash^\Sigma [\varphi]\psi$ and $\vdash \psi \rightarrow \chi$, then $\vdash^\Sigma [\varphi]\chi$ (*Right Weakening*).
2. If $\vdash^\Sigma [\varphi]\psi$ and $\vdash^\Sigma [\psi]\chi$, then $\vdash^\Sigma [\varphi]\chi$ (*Transitivity*).
3. If $\vdash^\Sigma [\varphi]\psi$ and $\vdash^\Sigma [\varphi]\chi$, then $\vdash^\Sigma [\varphi](\psi \wedge \chi)$ (*Conjunction*).
4. If $\vdash^\Sigma [\varphi]\chi$ and $\vdash \varphi \leftrightarrow \psi$, then $\vdash^\Sigma [\psi]\chi$ (*Left Logical Equivalence*).
5. If $\vdash^\Sigma [\varphi]\psi$ and $\vdash^\Sigma [\varphi][\psi]\chi$, then $\vdash^\Sigma [\varphi]\chi$ (*Chain Effects*).
6. If $\vdash^\Sigma [\chi]\psi$ and $\vdash \varphi \rightarrow \chi$, then $\vdash^\Sigma [\varphi]\psi$ (*Left Strengthening*).
7. If $\vdash^\Sigma [\varphi \vee \psi]\chi$, then $\vdash^\Sigma [\varphi]\chi \wedge [\psi]\chi$ (*Disjunctive Antecedents*).
8. If $\vdash^\Sigma [\varphi]\chi$ and $\vdash^\Sigma [\psi]\chi$, then $\vdash^\Sigma [\varphi \vee \psi]\chi$ (*Reasoning by Cases*).
9. If $\vdash^\Sigma [\varphi]\psi$ and $\vdash^\Sigma [\varphi \wedge \psi]\chi$, then $\vdash^\Sigma [\varphi]\chi$ (*Cumulative Transitivity*).
10. If $\vdash^\Sigma [\varphi \wedge \psi]\chi$, then $\vdash^\Sigma [\varphi]\chi \vee [\psi]\chi$ (*Conjunctive Antecedents*).

Note that the majority of acceptable properties of causal reasoning have been exhibited by $EPDL$. Some properties, for instance, *Conjunctive Antecedents*, are not desirable for all situations. This implies that not all the causal logics are applicable to any domain. Nevertheless, the list of properties gives us criteria for selecting the best suitable causal logic for reasoning about particular systems. We could also use them to compare the logics we presented in the paper and other causal theories⁷. For instance, [McCain and Tuner 1995]'s causal theory is different from all the causal logics we consider because it does not satisfy *Reasoning by Cases*

⁷[Schwind 1999] provided a comparative tableaux showing some of these properties satisfied by the main formalisms of causality in AI, which could be helpful for such a comparison.

(see [Giordano *et al.* 2000]) but satisfies *Cumulative Transitivity*. Those universal causal theories [Geffner 1992; Turner 1999; Giordano *et al.* 2000] (basically using $\varphi \rightarrow \mathbf{C}\psi$ to express causation between φ and ψ) satisfy almost all the properties except *Chain Effects* or *Pseudo-Causation*⁸. On the other hand, if we take McCain and Turner's *causal explanation* ([McCain and Turner 1997]) as semantics for causal relation, causal reasoning in *EPDL* and its extensions is *sound* in the following sense:

Let $\Sigma_i^* = \{[\chi]\lambda : \vdash_i^\Sigma [\chi]\lambda\} (i = 1, \dots, 4)$. It can be proved that if I is *causally explained* according to Σ , then it is *causally explained* according to Σ_i^* for any i . In other words, Σ_i^* is a conservative expansion of Σ with respect to the causal explanation semantics.

Some properties are not expected for causal reasoning. Besides the *Self-Causation* (Proposition 1), *Contraposition* is generally not acceptable for causal reasoning ([Mackie 1974]). For instance, $[\neg \text{alive}] \neg \text{walking}$ does not imply $[\text{walking}] \text{alive}$. The following example shows that even in *EPDL*₄, contraposition is not valid.

Example 3 Let $\Sigma = \{[\neg f_1]f_2\}$. Based on the proof of Proposition 1 we have $\not\vdash_4^\Sigma [\neg f_2]f_1$. \square

Nonmonotonicity sometimes is an attractive property of reasoning. Unfortunately (or fortunately), *EPDL* and its extensions are monotonic in both the following senses:

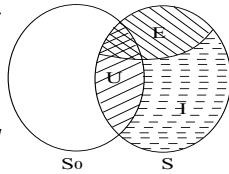
- If $\vdash^\Sigma [\varphi]\psi$, then $\vdash^{\Sigma \cup \Delta} [\varphi]\psi$.
- If $\vdash_i^\Sigma [\varphi]\psi$, then $\vdash_i^\Sigma [\varphi \wedge \chi]\psi$ for $i \geq 1$.

We do not think that nonmonotonicity is essential to causal reasoning. We can use causal logic in the meta-level to build monotonic or nonmonotonic systems for satisfying various applications. Nevertheless, it is still interesting to investigate nonmonotonic extensions of *EPDL*. For instance, the following axiom from conditional logic could be a promising substitute for *AND*⁹:

$$CV : \vdash ([\varphi]A \wedge \neg[\varphi]\neg\chi) \rightarrow [\varphi \wedge \chi]A.$$

4.2 Application to the ramification problem

Before we present some further properties of *EPDL*, let us consider an application of the logic. [Foo and Zhang 2001] presented a unified solution to the ramification problem. The main idea is the following. Suppose that we are given the initial state S_0 of a dynamic system, which is a maximal consistent set of fluent literals. A possible next state S of the system after performed an action should consist of three parts: *the direct effects E, the unchanged part U and the indirect effects I*.



According to the causation-based idea to the ramification problem, the indirect effects I should be caused by the direct

⁸*Conjunctive Antecedents* is always satisfied by such a logic but it is not acceptable for most cases of causal reasoning.

⁹If we conceive the causal relation $[\varphi]\psi$ as a nonmonotonic inference relation $\varphi \sim \psi$, then *AND* will be *Monotonicity* and *CV* will be *Rational Monotonicity* [Kraus *et al.* 1990] (One might have found some similarities between the logics *EPDL*_{*i*} and the logics in [Kraus *et al.* 1990]). Note that a significant difference between causal reasoning and nonmonotonic reasoning is that causal reasoning is not necessarily *reflexive*.

effects E and the unchanged part U .¹⁰ This idea can be formalized in our terminology as $\vdash^\Sigma [E \cup U]I$, where Σ consists of all the causal laws about the system¹¹. Therefore, a state S is a *possible next state* of the initial state S_0 if and only if $\vdash^\Sigma [E \cup U]I$, where $U = S_0 \cap S$ and $I = S \setminus (E \cup U)$.

This provides a unified approach to the ramification problem. We can choose different causal logics (not just the ones we presented here) to satisfy different applications against criteria of, for instance, inferential strength, expressive power, computational complexity and so on. This motivated us to provide a set of causal logics rather than a single one.

The results in the following section show that such a choice could be important at least for the solution to the ramification problem.

4.3 Characteristic theorem and complexity of reasoning with causal laws

A common misunderstanding in AI seems to be that an action theory in dynamic logic is prohibitively expensive in computation because the complexity of theorem proving of propositional dynamic logic is *exponential*. However, dynamic logic is highly expressive. It can express *any* complicated program. Therefore decidability may be the best result we can expect. The extended propositional dynamic logic for action and causation preserves decidability. With the same expressive power, however, the other formalisms of action may need second-order logic.

In this subsection, we argue that if we focus on causal reasoning with causal laws, the computational complexity in some causal logics can be lowered. First, we give a characterization theorem for *EPDL* and some of its extensions. This theorem shows that causal reasoning in these logics can be transformed into the classical propositional logic.

Theorem 5 Let Σ be a set of causal laws and $D(\Sigma) = \{\varphi \rightarrow \psi : [\varphi]\psi \in \Sigma\}$. Then

- $\vdash^\Sigma [\varphi]\psi$ iff $D(\Sigma) \cup \{\lambda : [\chi]\lambda \in \Sigma \ \& \ \vdash \varphi \leftrightarrow \chi\} \vdash \psi$.
- $\vdash_1^\Sigma [\varphi]\psi$ iff $D(\Sigma) \cup \{\lambda : [\chi]\lambda \in \Sigma \ \& \ \vdash \varphi \rightarrow \chi\} \vdash \psi$.
- $\vdash_2^\Sigma [\varphi]\psi$ iff there exist $\varphi_1, \dots, \varphi_n$ such that $\vdash \varphi \leftrightarrow (\varphi_1 \vee \dots \vee \varphi_n)$ and $\vdash_1^\Sigma [\varphi_k]\psi$ for each $k (1 \leq k \leq n)$.

The results in the theorem are a little bit complicated but still quite intuitive. $D(\Sigma)$ can be read as the domain constraints implied by the causal laws Σ . Take *EPDL*₁ as an example. Theorem 5 shows that given a set of causal laws Σ , a causal law $[\varphi]\psi$ is derivable if φ logically implies some causes whose effects logically imply ψ under the domain constraints. It is easy to see that the computational complexity of *causal reasoning with causal laws* in these three logics is in P^{NP} , so it is significantly lower than the complexity of *PDL*.

The characterization theorems for *EPDL*₃ and *EPDL*₄ are still open problem. The one for *EPDL*₃ could be fairly

¹⁰Note that the unchanged part should be considered as the causes of indirect effects because making something unchanged is also a kind of effect of actions.

¹¹We overloaded the set of literals to the conjunction of the set.

complicated. Relatively, the one for $EPDL_4$ might be simpler because we can transform a causal law into a normal form.

Example 4 Consider the circuit in Example 1. The causal relation in this circuit can be described by the following causal laws:

$$\Sigma = \left\{ \begin{array}{l} [sw_1 \wedge sw_2] light \\ [\neg sw_1 \vee \neg sw_2] \neg light \end{array} \right\}$$

Suppose that the initial state S_0 is $\{\neg sw_1, sw_2, \neg light\}$. Since $\not\models_1^\Sigma [sw_1 \wedge \neg light] \neg sw_2$, the only possible next state is $S = \{sw_1, sw_2, light\}$ after performed an action *Toggle*₁ if we use $EPDL_1$ as the meta-logic for reasoning. Note that $EPDL_1$ is enough for this example because we need not consider the disjunctive or nested causal laws here. \square

[Foo and Zhang 2001] provided a more general approach to deal with more complicated domains, which can also be applied in this framework.

5 Conclusion

We have presented an extended system $EPDL$ of propositional dynamic logic for reasoning about causality. The extended propositional dynamic logic provides a unified formalism for reasoning about (direct and indirect) effect of actions. The extension is so slight that only two axioms and two inference rules were added. It has been shown that the resultant system captures the basic properties of causal reasoning. Such simplicity may reflect the essential similarity between direct and indirect effect of actions.

$EPDL$ was then extended with a set of extra axioms on causal propagation. A series of causal logics were designed and the properties of these logics in causal reasoning with causal instances and causal laws are discussed, respectively. However, presenting these logics is not the main purpose of the work. We aimed to show in this paper how to construct causal logics for satisfying different applications in different domains. We believe that the representation of causation in dynamic logic affords the flexibility to accommodate different varieties of causal reasoning while retaining a core that captures their uncontroversial aspects.

There is a lot of work left for the future. Besides the nonmonotonic extensions, characterization theorems for $EPDL_3$ and $EPDL_4$, and complexity analysis of causal reasoning, a comprehensive comparison needs to be done with other formalisms of causality in AI and philosophy.

References

[Brand 1976] M. Brand ed., *The Nature of Causation*, University of Illinois Press, 1976.

[Castilho et al. 1999] M.A. Castilho, O. Gasquet, and A. Herzig, Formalizing action and change in modal logic I: the frame problem, *J. of Logic and Computations*, 5(9):701-735, 1999.

[De Giacomo and Lenzerini 1995] G. De Giacomo and M. Lenzerini, PDL-based framework for reasoning about actions, In M. Gori and G. Soda (Eds.), *Topics in Artificial Intelligence*, LNAI 992, 103-114, 1995.

[Foo and Zhang 2001] N. Foo and D. Zhang, Dealing with the ramification problem in extended propositional dynamic logic, To appear in F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyashev eds, *Advances in Modal Logic, Volume 3*, CSLI Publications, 2001.

[Geffner 1992] H. Geffner, *Default Reasoning: Causal and Conditional Theories*, The MIT Press, 1992.

[Gelfond and Lifschitz 1998] M. Gelfond and V. Lifschitz, Action language, *Electronic Transactions on AI*, 16(3), 1998.

[Giordano et al. 2000] L. Giordano, A. Martelli and C. Schwind, Ramification and causality in a modal action logic, *J. Logic Computat.* 5(10), 615-662, 2000.

[Giunchiglia et al. 1997] E. Giunchiglia, G. N. Kartha and V. Lifschitz, Representing action: indeterminacy and ramifications. *Artif. Intell.* 95(1997), 409-438, 1997.

[Goldblatt 1987] R. Goldblatt, *Logics of Time and Computation*, Stanford Univ. Press, Stanford CA, 1987.

[Goranko and Passy 1992] V. Goranko and S. Passy, Using the universal modality: gains and questions, *J. of Logic and Computation*, 2(1):5-30, 1992.

[Harel 1979] D. Harel, *First-order dynamic logic*, LNCS 68, Springer-Verlag, 1979.

[Lin 1995] F. Lin, Embracing causality in specifying the indirect effects of actions, *IJCAI-95*, 1985-1991, 1995.

[Kraus et al. 1990] S. Kraus, D. Lehmann and M. Magidor, Nonmonotonic reasoning, preferential models and cumulative logics, *Artif. Intell.*, (1-2)44, 167-207, 1990.

[Mackie 1974] J. L. Mackie, *The Cement of the Universe: A Study of Causation*, Oxford, 1974.

[McCain and Turner 1995] N. McCain and H. Turner, A causal theory of ramifications and qualifications *IJCAI-95*, 1978-1984, 1995.

[McCain and Turner 1997] N. McCain and H. Turner, Causal theories of action and change. *AAAI-97*, 460-465, 1997.

[Nute 1984] D. Nute, Conditional logic, in: D. Gabbay and F. Guenther eds, *Handbook of Philosophical Logic, Vol. II*, Kluwer Academic Publishers, 387-439, 1984.

[Prendinger and Schurz 1996] H. Prendinger and G. Schurz, Reasoning about action and change: A dynamic logic approach, *J. of Logic, Language, and Information*, 5:209-245, 1996.

[Ryan and Schobbens 1997] M. Ryan and P. -Y. Schobbens, Counterfactuals and updates as inverse modalities, *J. of Logic, Language and Information*, 6:123-146, 1997.

[Schwind 1999] C. Schwind, Causality in action theories, *Linköping Electronic Articles in Computer and Information Science*, Vol. 4 (1999):nr 4. <http://www.ep.liu.se/ea/cis/1999/004/>.

[Shoham 1988] Y. Shoham *Reasoning about Change: Time and Causation From the Standpoint of Artificial Intelligence*, The MIT Press, 1988.

[Thielscher 1997] M. Thielscher, Ramification and causality, *Artif. Intell.*, 1-2(89):317-364, 1997.

[Turner 1999] H. Turner, A Logic of universal causation, *Artif. Intell.* (1-2)113:87-123, 1999.

KNOWLEDGE REPRESENTATION AND REASONING

ACTION

A Circumscriptive Formalization of the Qualification Problem

G. Neelakantan Kartha

528 Blackfield Drive

Coppell, Tx 75019

gnkartha@yahoo.com

Abstract

The qualification problem refers to the difficulty that arises in formalizing actions, because it is difficult or impossible to specify in advance all the preconditions that should hold before an action can be executed. We study the qualification problem in the setting of the situation calculus and give a simple formalization using nested abnormality theories, a formalism based on circumscription. The formalization that we present allows us to combine a solution to the frame problem with a solution to the qualification problem.

1 Introduction

Two problems that are important in logical formalizations of commonsense reasoning are the frame problem and the qualification problem [McCarthy, 1977; 1980]. While the frame problem has received a lot of attention in the literature in recent years [Lin and Shoham, 1991; Lifschitz, 1991; Kartha and Lifschitz, 1994; Sandewall, 1994; McCain and Turner, 1995; Thielscher, 1997; Giunchiglia *et al.*, 1997; Shanahan, 1997; McIlraith, 2000], the qualification problem has not been studied as carefully (see however the discussion in the section on related work). In this paper, we clearly define what we mean by the qualification problem and introduce a formalization of the problem using nested abnormality theories [Lifschitz, 1995]. This formalization has the advantage that solutions to the frame problem can be incorporated smoothly with the solution to the qualification problem. We also point out problematic aspects of the current approaches to the qualification problem.

The qualification problem [McCarthy, 1980] refers to the difficulty that one has in formalizing an action, because it is difficult or impossible to enumerate all the preconditions that need to be satisfied before the action can be performed. One of the more well known examples that illustrate this problem is the potato in the tailpipe example due to McCarthy. The example is as follows: The action of starting a car results in the engine running. However, there are many implicit preconditions to the action of starting the car, for instance that there is no potato in the tailpipe, the battery is not dead etc. The problem is how to formalize such examples without getting bogged down in the representation of all such preconditions.

Let us examine the example in a little more detail. The action of starting the car (without any qualifications) can be formalized in the situation calculus [McCarthy and Hayes, 1969] as follows:

$$\neg \text{Holds}(\text{EngineRunning}, s) \supset \\ \text{Holds}(\text{EngineRunning}, \text{Result}(\text{StartCar}, s)). \quad (1)$$

To take into account various preconditions of the action *StartCar*, we could modify (1) by adding preconditions to its antecedent as in,

$$\neg \text{Holds}(\text{EngineRunning}, s) \wedge \neg \text{Holds}(\text{PotatoInTailPipe}, s) \\ \supset \text{Holds}(\text{EngineRunning}, \text{Result}(\text{StartCar}, s)). \quad (2)$$

However, note that we will be able to include in the language of our axiomatization only a subset of *all* the preconditions that can affect the outcome of an action, because we cannot foresee all of them. For instance, a factor that *could* affect the outcome of the action *StartCar* is the presence (or absence) of a mysterious ray. We would include facts about this ray in the language of our axiomatization only when we are given that this is a relevant fact.

Let us then start by fixing the language. Even then, it would be difficult to foresee all the circumstances (in the language) that can affect an action. Thus the qualification problem does not vanish when the language is fixed, but takes on a slightly different flavor. Now the key question becomes how one can axiomatize the domain of interest in such a way that on learning a new precondition for an action, one is able to incorporate this additional knowledge in a modular fashion in the axiomatization. Note that adding preconditions to the antecedent, as we have done in axiom (2) is not very modular.

Thus, it seems that there are really two related, but distinct aspects to the qualification problem.

1. The problem of how to smoothly incorporate various additions to the language of an axiomatization. The addition into the axiomatization of the facts pertaining to the mysterious ray and its effect on starting the car is an instance of this. Note that here the language of the axiomatization changes as a result of incorporating the change. This aspect of the qualification problem is related to McCarthy's notion of elaboration tolerance¹ [McCarthy,

¹According to McCarthy, "a formalism is elaboration tolerant to

1998]. This aspect is also related to the problem of how to move between axiomatizations (in different languages) that axiomatize a domain in different levels of detail.

2. The problem of how to represent the qualifications to an action, when the language of the axiomatization is fixed. The chief issues here are

- (a) Whether we can easily incorporate the fact that another circumstance (in the language) qualifies an action without drastically modifying the original axiomatization. For instance, we would like to achieve this by adding a new sentence into the language. Again, this is because of our desire for an elaboration tolerant solution.
- (b) To see whether the conclusions that are sanctioned formally agree with our intuitions.

Note that the two aspects share a desire for elaboration tolerance, so that a solution for one aspect is likely to contribute to a solution for the other. The current paper deals only with the second of the two aspects of the qualification problem.

The rest of the paper is organized as follows. In the next section, we explain our formalization and give an example of the kind of results that can be proved using our formalization. We then survey related work, summarize the key contributions of this work and point out directions for future work.

2 Formalization

Since our aim is to convey the key ideas, we will explain our formalization via an example, but will indicate at the appropriate places how one would generalize. The example we use is a slightly extended version of the potato in the tailpipe example from [Thielscher, 1996]. The example consists of extending the basic potato in the tailpipe scenario with another action, *PutPotato*. The result of this action is to put the potato in the tailpipe, unless this action is qualified. This action is qualified if the potato is so heavy that it cannot be lifted.

Following McCarthy [McCarthy, 1986], here are some axioms that capture the scenario:

$$\neg Holds(EngineRunning, s) \wedge \neg Ab(StartCar, s) \supset Holds(EngineRunning, Result(StartCar, s)), \quad (3)$$

$$\neg Holds(PotatoInTailPipe, s) \wedge \neg Ab(PutPotato, s) \supset Holds(PotatoInTailPipe, Result(PutPotato, s)), \quad (4)$$

$$Holds(PotatoInTailPipe, s) \supset Ab(StartCar, s), \quad (5)$$

$$Holds(HeavyPotato, s) \supset Ab(PutPotato, s), \quad (6)$$

$$\neg Holds(EngineRunning, S_0). \quad (7)$$

the extent it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances”.

Here *Ab* is an abnormality predicate that will be minimized. The advantage of this representation is that as we think of new qualifications, they can be easily incorporated into the theory by means of axioms such as (5) and (6).

However, it should be noted that we are rarely interested in solving the qualification problem in isolation. We also need to ensure that the truth values of the fluents do not change without a reason, so that we are able to obtain the intuitively expected results. To see this, let T be the theory consisting of axioms (3), (4), (5), (6) and (7). Let T' be the theory obtained by minimizing *Ab* in T . We would like T' to entail the formula $\neg Ab(PutPotato, S_0)$ and consequently $\neg Holds(HeavyPotato, S_0)$. Although, intuitively we would then conclude that the fluent *HeavyPotato* remains false after executing an action in S_0 (since there are no actions that change the truth value of that fluent), T' is not strong enough to entail formulae such as $\neg Holds(HeavyPotato, Result(StartCar, S_0))$, $\neg Holds(HeavyPotato, Result(PutPotato, S_0))$ etc. This shows that we need to combine a solution to the frame problem with a solution to the qualification problem to enable us to draw the intuitively expected conclusions.

The formalism that we use that helps us in tackling the frame problem and the qualification problem in sequence, without the solutions interfering with each other is the formalism of nested abnormality theories (NATs) [Lifschitz, 1995]. The main feature of NATs is that the effect of a circumscription can be restricted to a part of the theory called a block. Formally, a block is of the form $\{C_1, \dots, C_m : \Phi_1, \dots, \Phi_n\}$, where C_1, \dots, C_m are function and/or predicate constants and Φ_1, \dots, Φ_n are axioms, possibly containing an abnormality predicate *Ab*. The block corresponds to the circumscription of *Ab* in the theory consisting of Φ_1, \dots, Φ_n , with C_1, \dots, C_m allowed to vary. Note that each of Φ_1, \dots, Φ_n itself may be a block.

Formally, the semantics of NATs is defined by a map g that translates blocks into sentences of the underlying language. We define g as follows:

$$g\{C_1, \dots, C_m : \Phi_1, \dots, \Phi_n\} = \exists ab G(ab),$$

where

$$G(ab) = \text{CIRC}[g\Phi_1 \wedge \dots \wedge g\Phi_n; Ab; C_1, \dots, C_m].$$

Here CIRC stands for the circumscription of *Ab* in $g\Phi_1 \wedge \dots \wedge g\Phi_n$ with C_1, \dots, C_m allowed to vary (see [Lifschitz, 1993a] for the formal definitions). For a sentence Φ that does not include *Ab*, $g\Phi$ is equivalent to Φ .

Since we need to consider the frame problem, we will need to axiomatize the commonsense law of inertia, according to which the truth values of fluents persist unless they are abnormal. Using ideas from [Karthan and Lifschitz, 1995], we formalize this as

$$\neg Ab(f, a, s) \supset Holds(f, s) \equiv Holds_R(f, a, s), \quad (8)$$

and denote it by *LI*. Here $Holds_R$ is a new predicate, explicitly defined outside the block solving the frame problem as

$$Holds_R(f, a, s) \equiv Holds(f, Result(a, s)).$$

The role of this predicate for solving the frame problem can be explained roughly as follows—by “factoring away” the result function, we are able to focus attention only on the situation s and the situation that we obtain after action a is performed in s . For more details and intuitions, the reader is referred to [Karthan and Lifschitz, 1995].

In our formalization, we will formalize the effect of performing an action using the $Hold s_R$ predicate. Thus, for instance, the axioms (3) and (4) will become

$$\neg Hold s(EngineRunning, s) \wedge \neg Ab(StartCar, s) \supset Hold s_R(EngineRunning, StartCar, s), \quad (9)$$

$$\neg Hold s(PotatoInTailPipe, s) \wedge \neg Ab(PutPotato, s) \supset Hold s_R(PotatoInTailPipe, PutPotato, s). \quad (10)$$

Let T_e denote the conjunction of axioms that formalize the effect of performing an action using the predicate $Hold s_R$. Then each formula of T_e is of the syntactic form

$$pre_{F,A}^+(s) \wedge \neg Ab(A, s) \supset Hold s_R(F, A, s)$$

or

$$pre_{F,A}^-(s) \wedge \neg Ab(A, s) \supset \neg Hold s_R(F, A, s).$$

where $pre_{F,A}^\pm(s)$ denotes a finite conjunction of formulas of the form $Hold s(F, s)$ or $\neg Hold s(F, s)$. In our example, T_e thus stands for the conjunction of (9) and (10).

Similarly, let T_{qua} stand for the conjunction of axioms that formalize the qualifications. Syntactically, these axioms are of the form

$$qual_A(s) \supset Ab(A, s)$$

where $qual_A(s)$ is a finite disjunction of conjunction of formulas of the form $Hold s(F, s)$ and $\neg Hold s(F, s)$. In our example, T_{qua} is the conjunction of (5) and (6).

Let T_{obs} stand for the conjunction of axioms that formalize the observations. These axioms are a finite conjunction of formulas of the form $Hold s(F, S_0)$ and $\neg Hold s(F, S_0)$. In our case, T_{obs} is (7).

We also introduce the uniqueness of names and domain closure axioms for fluents and actions. For instance, in our example, the uniqueness of names axiom for fluents consists of the following axiom.

$$\begin{aligned} EngineRunning &\neq HeavyPotato \wedge \\ EngineRunning &\neq PotatoInTailPipe \wedge \\ HeavyPotato &\neq PotatoInTailPipe. \end{aligned}$$

The domain closure axiom for fluents is

$$\begin{aligned} f &= EngineRunning \vee \\ f &= PotatoInTailPipe \vee \\ f &= HeavyPotato. \end{aligned}$$

Let UNA and DCA denote respectively the uniqueness of names axioms and the domain closure axioms for fluents and actions.

We need two technical devices for our axiomatization. Firstly, we need to include an existence of situation axiom ([Baker, 1991]) (denoted ES) that states that corresponding to every set of fluents, there is a situation in which exactly

these fluents hold. For our example, the existence of situation axiom consists of the conjunction of

$$\begin{aligned} \exists s(\neg Hold s(EngineRunning, s) \wedge \\ \neg Hold s(PotatoInTailPipe, s) \wedge \\ \neg Hold s(HeavyPotato, s)), \end{aligned}$$

and seven other similar formulae asserting the existence of a situation corresponding to other possible truth values of the three fluents.

Secondly, we need to include a formula which states that there is a situation distinct from S_0 where exactly those fluents that hold in S_0 hold. Formally, this formula is

$$\exists s[s \neq S_0 \wedge \forall f[Hold s(f, s) \equiv Hold s(f, S_0)]].$$

This formula is denoted ESI (for existence of situation for the initial situation).

We are now ready to give our formalization.

$$\begin{aligned} &UNA, \\ &DCA, \\ &\{Hold s, Hold s_R, Result, S_0 : \\ &ES, \\ &ESI, \\ &Hold s_R(f, a, s) \equiv Hold s(f, Result(a, s)), \\ &\{Hold s_R : \\ &LI, \\ &T_e, \\ &T_{qua}, \\ &T_{obs}, \\ &\} \}. \end{aligned}$$

The inner block solves the frame problem. Note that the abnormality predicate that is being circumscribed in this block is the ternary one that occurs in LI , not the binary one occurring in T_e . Also, T_{qua} and T_{obs} do not contain any occurrence of $Hold s_R$ and $Ab(f, a, s)$, and hence do not play any role in the circumscription of the inner block². The outer block minimizes the binary Ab and solves the qualification problem. The existence of situations axioms ensure that the binary abnormalities are minimized over *all* situations. The strategy of first solving the frame problem and then the qualification problem can also be found in [Ginsberg and Smith, 1988; Lin and Reiter, 1994].

Let N be the NAT that encodes our example. By entailment relation for N , we will mean the circumscriptive entailment relation for NATs (for details, see [Lifschitz, 1995]). As a representative of the kind of results we can prove, we state the following theorem.

Theorem 1 *The following set of formulae are entailed by N .*

1. $\neg Hold s(HeavyPotato, S_0) \wedge \neg Hold s(PotatoInTailPipe, S_0)$.
2. $\forall as(Hold s(HeavyPotato, s) \equiv Hold s(HeavyPotato, Result(a, s)))$.
3. $\neg Hold s(EngineRunning, Result(PutPotato, S_0))$.
4. $\neg Hold s(EngineRunning, Result(StartCar, S_1))$ where $S_1 = Result(PutPotato, S_0)$.

²This can also be achieved by moving T_{qua} and T_{obs} from the inner to the outer block.

The first formula states that the fluents *HeavyPotato* and *PotatoInTailPipe* do not hold in the initial situation. This is intuitively expected, since we are given no information as to whether these fluents hold in the initial situation and we would like to assume that these do not hold if we can do so in order to minimize the abnormalities.

The second formula shows that the value of the fluent *HeavyPotato* does not change as a result of performing an action. Again, this is expected from a solution to the frame problem, as there are no actions that can change the truth value of the fluent *HeavyPotato*. The third formula follows from the solution of the frame problem for the fluent *EngineRunning* and (7).

The fourth formula shows that if you put potato in the tailpipe and then start the car, the engine does not start. This is because as a result of performing the first action *PutPotato*, the second action *StartCar* becomes qualified.

Without the *ES* and *ESI* axioms, we would not be able to obtain the desired conclusions from our formalization. To see this, let us consider a formalization, N_1 , of our example that differs from N only in that N_1 does not include *ES* and *ESI*. Define a structure M with the universe of situations, $|M|_s$ defined as $\{0\}$, the universe of fluents, $|M|_f$ is defined as $\{PotatoInTailPipe, EngineRunning, HeavyPotato\}$, the universe of actions is defined as $\{StartCar, PutPotato\}$, the interpretation of the predicate *Holds* defined as $Holds^M = \{(PotatoInTailPipe, 0)\}$ and the interpretation of *Result*, $Result^M$ defined as $Result^M(A, 0) = 0$, where A ranges over the actions. It is easy to check that M is a model of N_1 . Hence, we see that $N_1 \not\models \neg Holds(PotatoInTailPipe, S_0)$.

We noted that N entails the formula $\neg Holds(HeavyPotato, S_0)$. Clearly, this formula will not be entailed if T_{obs} includes the formula $Holds(HeavyPotato, S_0)$. This shows that adding an observation does not reduce the set of models of N , but changes them. Hence the solution that we present here does not have the restricted monotonicity property [Lifschitz, 1993b] with respect to observation sentences. In general, we do not expect this property to be obeyed by solutions to the qualification problem.

3 Related Work

McCarthy [McCarthy, 1986] has suggested using circumscription to solve the qualification problem. The use of *Ab* in our formalization is based on his abnormality predicate, but several ideas in our formalization such as the use of the $Holds_R$ predicate and the use of the existence of situations axiom for solving the qualification problem are new. Also, we consider how to combine a solution to the frame problem with a solution to the qualification problem.

In his brief paper on the qualification problem, Elkan [Elkan, 1995] suggests using a context mechanism for formalizing the qualification problem, but does not present any such formalization. He criticizes a particular formalization of the qualification problem based on default logic on the ground that all the potential qualifications to an action need to be made explicit in that formalization. While the criticism is on the mark for the particular formalization, it seems that a formal-

ization in default logic that uses an abnormality predicate will not suffer from this criticism.

Lifschitz [Lifschitz, 1987] formalizes the qualification problem by circumscribing a predicate *precond*. His formalism is less expressive than ours, since it does not permit one to express that an action is qualified if two fluents are simultaneously true. Also, his formalism is embedded within a causal language known to have difficulties with domain constraints³ [Baker, 1991].

Thielscher [Thielscher, 1996] claims that global minimization of abnormalities fails to solve the qualification problem. The example that he uses is the one that we have formalized in the previous section as a NAT. The question that Thielscher raises is as to what would happen if we perform the action *PutPotato* followed by the action *StartCar*. He claims that if we globally minimize abnormalities, there would be a model where the action *PutPotato* does not have its intended effect (because that action is qualified) and hence in that model the engine would start after performing the action *StartCar*.

It should be noted that Thielscher gives only an informal argument for the claim, not a formal proof. His claim seems to be based on the fact that he is considering a propositional language and does not consider abnormalities to have situational arguments. As we have shown, the unintended model does not arise in our formalization.

Thielscher represents qualifications in the form

$$a \text{ disqualified after } [a_1, \dots, a_n],$$

which indicates that after performing the sequence of actions a_1, \dots, a_n , action a becomes disqualified. To specify such axioms, it is necessary to precompute what the effect of different sequences of actions should be, which is cumbersome. In contrast, our formalization allows us to specify qualifications directly in terms of fluents. However, Thielscher considers the issue of “miraculous qualifications”, an issue that we do not consider.

McIlraith [McIlraith, 2000] addresses the frame, ramification and qualification problem within the context of the situation calculus. In contrast to the approach presented in this paper, McIlraith compiles a set of effect axioms and ramification constraints into a set of successor state axioms. It should be noted that such a compilation is possible only when the effect axioms and ramification constraints follow a syntactic restriction. (Theories with such a syntactic restriction are called solitary stratified theories in [McIlraith, 2000]). Whether the approach presented in this paper coincides with that in [McIlraith, 2000] when restricted to the case of solitary stratified theories remains open.

4 Conclusions and Future Work

We have presented a simple formalization of the qualification problem based on NATs and have shown how to combine a solution to the frame problem with this formalization. We

³By domain constraints, we mean formulae that relate fluents in the same situation. An example of a domain constraint that we may wish to include in our example is $Holds(PotatoInTailPipe, s) \supset \neg Holds(EngineRunning, s)$.

have also argued that this formalization offers several advantages in comparison with existing approaches.

Theorem 1 pertains only to the example presented in this paper. This raises the question as to how we can evaluate the general applicability of the formalization presented here. One way is by proving theorems about what the formalization does and does not entail. For instance, we can prove that the inner block (that solves the frame problem) generates conclusions identical to the ones sanctioned by a \mathcal{A} -like [Gelfond and Lifschitz, 1993] language. Evaluating that the outer block handles qualifications correctly is harder. One approach is to test that the formalization yields intuitively expected results on a test suite. A more principled approach is to define a language formalizing intuitions about qualifications and then prove soundness and completeness theorems along the lines of [Kartha, 1993]. Both these approaches are currently being pursued.

One direction for future work is to investigate how the ideas presented in this paper can be generalized to handle domain constraints. In the presence of domain constraints, the situation is more complicated for the following reasons.

- The existence of situations axiom needs to be changed, since now, all possible combination of fluents are not consistent.
- As pointed out by [Ginsberg and Smith, 1988] and [Lin and Reiter, 1994], some domain constraints can act as constraints on the performability of an action (these are called qualification constraints) whereas some other syntactically indistinguishable domain constraints give rise to indirect effects (these are called ramification constraints).

The first of the above complications can be handled, for instance, by including a block that formalizes a nonmonotonic version of the existence of situations axiom, as in [Kartha and Lifschitz, 1994].

The second difficulty is more serious. If all the constraints are ramification constraints, then solutions to the frame problem shown here can be easily extended (along the lines of [Giunchiglia *et al.*, 1997]). If we have both ramification and qualification constraints, we need to separate the set of constraints into these two sets and consider them separately.

Acknowledgment

I wish to thank the anonymous referees for several suggestions for improving the clarity of this paper.

References

- [Baker, 1991] Andrew Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.
- [Elkan, 1995] Charles Elkan. On solving the qualification problem. In *Working Notes of the Symposium on Extending Theories of Actions*, 1995.
- [Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.
- [Ginsberg and Smith, 1988] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: a possible worlds approach. *Artificial Intelligence*, 35(2):165–195, 1988.
- [Giunchiglia *et al.*, 1997] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95:409–438, 1997.
- [Kartha and Lifschitz, 1994] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects. In Jon Doyle, Erik Sandewall, and Piero Torasso, editors, *Proc. of the Fourth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 341–350, 1994.
- [Kartha and Lifschitz, 1995] G. Neelakantan Kartha and Vladimir Lifschitz. A simple formalization of actions using circumscription. In *Proc. of IJCAI-95*, pages 1970–1975, 1995.
- [Kartha, 1993] G. Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *Proc. of IJCAI-93*, pages 724–729, 1993.
- [Lifschitz, 1987] Vladimir Lifschitz. Formal theories of action (preliminary report). In *Proc. of IJCAI-87*, pages 966–972, 1987.
- [Lifschitz, 1991] Vladimir Lifschitz. Towards a metatheory of action. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proc. of the Second Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 376–386, 1991.
- [Lifschitz, 1993a] Vladimir Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *The Handbook of Logic in AI and Logic Programming*, volume 3, pages 297–352. Oxford University Press, 1993.
- [Lifschitz, 1993b] Vladimir Lifschitz. Restricted monotonicity. In *Proc. of the Eleventh National Conference of Artificial Intelligence*, pages 432–437, 1993.
- [Lifschitz, 1995] Vladimir Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74:351–365, 1995.
- [Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 4(5):655–678, 1994.
- [Lin and Shoham, 1991] Fangzhen Lin and Yoav Shoham. Provably correct theories of action. In *Proc. of the Ninth National Conference of Artificial Intelligence*, pages 349–354, 1991.
- [McCain and Turner, 1995] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proc. of IJCAI-95*, pages 1978–1984, 1995.
- [McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

- [McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proc. of IJCAI-77*, pages 1038–1044, 1977.
- [McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980.
- [McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.
- [McCarthy, 1998] John McCarthy. Elaboration tolerance. In *Fourth Symposium on Logical Formalizations of Commonsense Reasoning*, 1998.
- [McIlraith, 2000] Sheila McIlraith. Integrating actions and state constraints: A closed form solution to the ramification problem (sometimes). *Artificial Intelligence*, 116:87–121, 2000.
- [Sandewall, 1994] Erik Sandewall. *Features and Fluents. The representation of Knowledge about Dynamical Systems*. Oxford University Press, 1994.
- [Shanahan, 1997] Murray Shanahan. *Solving the Frame Problem*. MIT Press, Cambridge, MA, 1997.
- [Thielscher, 1996] Michael Thielscher. Causality and the qualification problem. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, Cambridge, MA, November 1996. Morgan Kaufmann.
- [Thielscher, 1997] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

Computing Strongest Necessary and Weakest Sufficient Conditions of First-Order Formulas

Patrick Doherty*

Dept. of Computer Science
Linköping University
S-581 83 Linköping, Sweden
email: patdo@ida.liu.se

Witold Łukaszewicz†

Andrzej Szalas‡

Dept. of Computer Science, Linköping University
and College of Economics and Computer Science
TWP, Olsztyn, Poland
email: witlu,andsz@ida.liu.se

Abstract

A technique is proposed for computing the weakest sufficient (wsc) and strongest necessary (snc) conditions for formulas in an expressive fragment of first-order logic using quantifier elimination techniques. The efficacy of the approach is demonstrated by using the techniques to compute snc's and wsc's for use in agent communication applications, theory approximation and generation of abductive hypotheses. Additionally, we generalize recent results involving the generation of successor state axioms in the propositional situation calculus via snc's to the first-order case. Subsumption results for existing approaches to this problem and a re-interpretation of the concept of *forgetting* as a process of quantifier elimination are also provided.

1 Introduction

In [Lin, 2000]¹, Lin proposes the notion of weakest sufficient and strongest necessary conditions for a proposition q under a propositional theory T , where the techniques for generating the resulting formulas may be parameterized to contain only a restricted subset of the propositional variables in T . In addition, he investigates a number of methods for automatically generating snc's and wsc's, several based on the generation of prime implicates and a preferred method for computing snc's and wsc's based on the notion of *forgetting* [Lin and Reiter, 1994]. Snc's and wsc's have many potential uses and applications ranging from generation of abductive hypotheses to approximation of theories. In fact, special cases of snc's and wsc's, strongest postconditions and weakest preconditions, have had widespread usage as a basis for programming language semantics [Dijkstra, 1976].

Weakest sufficient and strongest necessary conditions for propositional formulas are related to prime implicants and implicates, respectively. Classically, a prime implicant of a

formula α is a minimal satisfiable term logically implying α and a prime implicate is a minimal satisfiable clause which is logically implied by α . The relation between prime implicants/implicates and wsc's/snc's is used by Lin in his investigation of methods for automatically generating wsc's and snc's. Generating prime implicants and implicates for propositional formulas is intractable and in the general case, the same applies for wsc's and snc's.

Lin's results apply to the propositional case and his algorithms for automatically generating snc's and wsc's are empirically tested. For the propositional case, we propose a different method for computing snc's and wsc's that is based on second-order quantifier elimination techniques and has the following advantages:

- The general method applies to the full propositional language and snc's and wsc's are generated directly for arbitrary formulas rather than propositional atoms.
- When applying second-order quantifier elimination, one substantially simplifies the propositional case considered by Lin and often gets a more efficient computation method.
- A non-trivial fragment of the propositional language is isolated where snc's and wsc's for formulas in this fragment can be generated efficiently and are guaranteed to be so.
- The quantifier elimination algorithms which provide the basis for automatically generating snc's and wsc's for arbitrary propositional formulas are implemented.

One of the most interesting and potentially fruitful open problems regarding snc's and wsc's is generalization to the first-order case and developing associated methods for automatically computing snc's and wsc's. In [Lin, 2000], Lin states,

There are several directions for future work. One of them is to extend the results here to the first-order case. This can be a difficult task. For instance, a result in [Lin and Reiter, 1997] shows that forgetting in the first-order case cannot in general be expressible in first-order logic. As a consequence, we expect that strongest necessary conditions of a proposition under a first-order theory cannot in general be expressible in first-order logic either. It seems that the best hope for dealing with the first-order case is to first reduce it to the propositional case, and then

*Supported in part by the WITAS project grant under the Wallenberg Foundation, Sweden.

†Supported in part by the WITAS project grant under the Wallenberg Foundation, Sweden and KBN grant 8 T11C 009 19.

¹This paper received the best paper award at the KR'00 Conference, Breckenridge, Colorado, 2000.

try to learn a first-order description from a set of propositional ones.

With Lin, we agree that the task is difficult. We also agree that in the general case snc's and wsc's for a proposition or first-order formula under a first-order theory is not always expressible in first-order logic. In fact, the techniques we propose provide a great deal of insight into why this is the case and in addition define a non-trivial fragment of first-order logic where snc's and wsc's are guaranteed to be expressible in first-order logic.

Rather than using indirect techniques to reduce a first-order case to the propositional case and then try to learn a first-order description from a set of propositional ones as Lin suggests, we propose a more direct method and provide techniques for the automatic generation of snc's and wsc's for a first-order fragment. Given a theory T and a formula α in this fragment, we simply append appropriate quantifiers over relational variables in $T \wedge \alpha$ (or $T \rightarrow \alpha$) and use second-order quantifier elimination techniques to reduce the second-order formula to a logically equivalent first-order formula representing the snc or the wsc for α under T . Complexity results are provided for this fragment, but the method works for full first-order logic. In this case, depending on the nature of T and α , the technique may return a logically equivalent first-order or fixpoint formula, or terminate with failure, not always because there is not a reduction, but simply because the elimination algorithm can not find a reduction.

We compute these conditions using extensions of results described in the work of [Doherty *et al.*, 1997; 1998; Nonnengart and Szalas, 1998]. For a survey of these and other quantifier elimination techniques, see also [Nonnengart *et al.*, 1999].

1.1 Weakest Sufficient and Strongest Necessary Conditions

In the following, we will be dealing with the predicate calculus with equality, i.e. we assume that the equality, $=$, is always a logical symbol. The following definitions describe the necessary and sufficient conditions of a formula α relativized to a subset P of relation symbols under a theory T .

Definition 1.1 By a *necessary condition* of a formula α on the set of relation symbols P under theory T we shall understand any formula ϕ containing only symbols in P such that $T \models \alpha \rightarrow \phi$. It is a *strongest necessary condition*, denoted by $\text{SNC}(\alpha; T; P)$ if, additionally, for any necessary condition ψ of α on P under T , we have that $T \models \phi \rightarrow \psi$. ■

Definition 1.2 By a *sufficient condition* of a formula α on the set of relation symbols P under theory T we shall understand any formula ϕ containing only symbols in P such that $T \models \phi \rightarrow \alpha$. It is a *weakest sufficient condition*, denoted by $\text{WSC}(\alpha; T; P)$ if, additionally, for any sufficient condition ψ of α on P under T , we have that $T \models \psi \rightarrow \phi$. ■

The set P in Definitions 1.1 and 1.2 is referred to as the *target language*.

To provide some intuition as to how these definitions can be used, consider the theory

$$T = \{ \forall x. [\text{HasWheels}(x) \rightarrow \text{CanMove}(x)], \\ \forall x. [\text{Car}(x) \rightarrow \text{HasWheels}(x)] \},$$

and the formula $\alpha = \forall x. \text{CanMove}(x)$. Clearly $T \not\models \alpha$. Quite often, it is useful to hypothesize a preferred explanation ϕ for α under a theory T where $T \wedge \phi \models \alpha$, ϕ is minimal in the sense of not being overly specific and where the explanation is constrained to a particular subset P of symbols in the vocabulary. Clearly, the weakest sufficient condition ϕ for the formula α on P under T provides the basis for a minimal preferred explanation of α where $T \models \phi \rightarrow \alpha$. In the case of $P = \{\text{HasWheels}\}$, the weakest sufficient condition would be $\phi = \forall x. \text{HasWheels}(x)$, and in the case of $P = \{\text{HasWheels}, \text{Car}\}$ the wsc would be $\phi = \forall x. \text{HasWheels}(x)$. Generating abductive hypotheses is just one application of wsc's. There are many other applications which require generation of wsc's or snc's, several of which are described in section 5.

1.2 Paper Structure

In section 2, we begin with preliminaries and state the theorem which provides the basis for second-order quantifier elimination techniques. In section 3, we define snc's and wsc's for propositional formulas under propositional theories as second-order formulas with quantification over propositional symbols, show how the elimination techniques are applied and provide complexity results for the technique. In section 4, we generalize to the first-order case using primarily the same techniques, but with quantification over relational symbols. In section 5, we demonstrate both the use of snc's and wsc's in addition to the reduction techniques by providing examples from a number of potentially interesting application areas such as agent communication languages, theory approximation, generation of abductive hypotheses and generation of successor state axioms in the situation calculus. In section 6, we relate the proposed techniques to the notion of *forgetting* which serves as a basis for Lin's techniques and in so doing, prove subsumption results. In section 7, we conclude with a discussion about these results.

2 Preliminaries

The following Theorem 2.2 has been proved in [Nonnengart and Szalas, 1998]. It allows us to eliminate second-order quantifiers from formulas which are in the form appearing on the left-hand side of the equivalences (1), (2). Such formulas are called *semi-Horn formulas* w.r.t. the relational variable Φ - see also [Doherty *et al.*, 1996]. Observe, that in the context of databases one remains in the tractable framework, since fixpoint formulas over finite domains are computable in polynomial time (and space) - see e.g. [Abiteboul *et al.*, 1996; Ebbinghaus and Flum, 1995]. (Of course, the approach is applicable to areas other than databases, too).

Notation 2.1 Let e, t be any expressions and s any subexpression of e . By $e(s := t)$ we shall mean the expression obtained from e by substituting each occurrence of s by t . $\mu\Phi.A(\Phi)$ is the least fixpoint operator and $\nu\Phi.A(\Phi)$ is defined as $\neg\mu\neg\Phi.\neg A(\Phi)$.

Let $A(\bar{x})$ be a formula with free variables \bar{x} . Then by $A(\bar{x})[\bar{a}]$ we shall mean the application of $A(\bar{x})$ to arguments \bar{a} . ■

Theorem 2.2 Assume that all occurrences of the predicate variable Φ in the formula B bind only variables and that formula A is positive w.r.t. Φ .

- if B is negative w.r.t. Φ then

$$\exists \Phi \forall \bar{y} [A(\Phi) \rightarrow \Phi(\bar{y})] \wedge [B(\neg \Phi)] \equiv B[\Phi(\bar{t}) := \mu \Phi(\bar{y}).A(\Phi)[\bar{t}]] \quad (1)$$

- if B is positive w.r.t. Φ then

$$\exists \Phi \forall \bar{y} [\Phi(\bar{y}) \rightarrow A(\Phi)] \wedge [B(\Phi)] \equiv B[\Phi(\bar{t}) := \nu \Phi(\bar{y}).A(\Phi)[\bar{t}]]. \quad (2)$$

■

Example 2.3 Consider the following second-order formula:

$$\exists \Phi \forall x \forall y [(S(x, y) \vee \Phi(y, x)) \rightarrow \Phi(x, y)] \wedge [\neg \Phi(a, b) \vee \forall z (\neg \Phi(a, z))] \quad (3)$$

According to Theorem 2.2(1), formula (3) is equivalent to:

$$\neg \mu \Phi(x, y). (S(x, y) \vee \Phi(y, x))[a, b] \vee \forall z (\neg \mu \Phi(x, y). (S(x, y) \vee \Phi(y, x))[a, z]). \quad (4)$$

■

Observe that, whenever formula A in Theorem 2.2 does not contain Φ , the resulting formula is easily reducible to a first-order formula, as in this case both $\mu \Phi(\bar{y}).A$ and $\nu \Phi(\bar{y}).A$ are equivalent to A . In fact, this case is equivalent to the lemma of Ackermann (see e.g. [Doherty *et al.*, 1997]). Semi-Horn formulas of the form (1) and (2), where A does not contain Φ , are called *non-recursive semi-Horn formulas*.

3 The Propositional Case

In this section, we define *snc*'s and *wsc*'s for propositional formulas under propositional theories as second-order formulas with quantification over propositional symbols, show how the elimination techniques are applied and provide complexity results for the technique. We start with the following lemma.

Lemma 3.1 For any formula α , any set of propositional symbols P and theory Th :

1. the strongest necessary condition $SNC(\alpha; Th; P)$ is defined by $\exists \bar{q}. [Th \wedge \alpha]$,
2. the weakest sufficient condition $WSC(\alpha; Th; P)$ is defined by $\forall \bar{q}. [Th \rightarrow \alpha]$,

where \bar{q} consists of all propositions appearing in Th and α but not in P .

Proof The proof of the lemma for both the strongest necessary and weakest sufficient conditions are similar, but we provide both for clarity.

By definition, any necessary condition ϕ for α satisfies $Th \models \alpha \rightarrow \phi$, i.e. by the deduction theorem for propositional calculus, also $\models Th \rightarrow (\alpha \rightarrow \phi)$, i.e. $\models (Th \wedge \alpha) \rightarrow \phi$. Thus also $\models \exists \bar{q}. [(Th \wedge \alpha) \rightarrow \phi]$. Since ϕ is required not to contain symbols from \bar{q} , we have

$$\models [\exists \bar{q}. (Th \wedge \alpha)] \rightarrow \phi. \quad (5)$$

On the other hand, the minimal ϕ satisfying (5) is given by equivalence

$$[\exists \bar{q}. (Th \wedge \alpha)] \equiv \phi.$$

This proves Lemma 3.1.1.

By definition, any sufficient condition ϕ for α satisfies $Th \models \phi \rightarrow \alpha$, i.e. by the deduction theorem for propositional calculus, also $\models Th \rightarrow (\phi \rightarrow \alpha)$, i.e. $\models (Th \wedge \phi) \rightarrow \alpha$. Thus also $\models \forall \bar{q}. [(Th \wedge \phi) \rightarrow \alpha]$ which is equivalent to $\models \forall \bar{q}. [(Th \wedge \neg \alpha) \rightarrow \neg \phi]$.

Since ϕ is required not to contain symbols from \bar{q} , we have

$$\models [\exists \bar{q}. (Th \wedge \neg \alpha)] \rightarrow \neg \phi. \quad (6)$$

Maximizing ϕ is the same as minimizing $\neg \phi$. On the other hand, the maximal ϕ satisfying (6) is given by equivalence

$$[\exists \bar{q}. (Th \wedge \neg \alpha)] \equiv \neg \phi.$$

which is equivalent to

$$\neg [\exists \bar{q}. (Th \wedge \neg \alpha)] \equiv \phi.$$

which is equivalent to

$$[\forall \bar{q}. (Th \rightarrow \alpha)] \equiv \phi.$$

This proves Lemma 3.1.2. ■

The quantifiers over propositions can be automatically eliminated using the DLS algorithm (see [Doherty *et al.*, 1997]). For instance, all eliminations in Example 3.3 can be done using the algorithm. Theorem 2.2 reduces in the propositional case to Proposition 3.2. It is worth emphasizing here that propositional fixpoint formulas are equivalent to propositional formulas.²

Proposition 3.2 Assume that the propositional formula A is positive w.r.t. proposition p .

- if the propositional formula B is negative w.r.t. p then

$$\exists p. [A(p) \rightarrow p] \wedge [B(\neg p)] \equiv B[p := \mu p.A(p)] \quad (7)$$

- if B is positive w.r.t. p then

$$\exists p. [p \rightarrow A(p)] \wedge [B(p)] \equiv B[p := \nu p.A(p)]. \quad (8)$$

■

Observe that in the case when an input formula is a conjunction of propositional semi-Horn formulas of the form in the *lhs* of (7) or a conjunction of formulas of the form in the *lhs* of (8), the length of the resulting formula is, in the worst case, $O(n^2)$, where n is the size of the input formula. Otherwise the result might be of exponential length, as in the case of the algorithm given in [Lin, 2000].

Example 3.3 Consider the following examples of [Lin, 2000].

²In the first iteration towards the fixpoint, one replaces p in A with false. In the next disjunct, p in A is replaced by this result. The fixpoint, a propositional formula, is then always reached in at most two iterations.

1. $T_1 = \{q \rightarrow (p_1 \wedge p_2)\}$. Now, according to Lemma 3.1, $\text{SNC}(q; T_1; \{p_1, p_2\})$ is defined by formula $\exists q. [(q \rightarrow (p_1 \wedge p_2)) \wedge q]$, which, according to Proposition 3.2, is logically equivalent to $(p_1 \wedge p_2)$.

Condition $\text{SNC}(q; T_1; \{p_1\})$ is defined by formula $\exists q \exists p_2. [(q \rightarrow (p_1 \wedge p_2)) \wedge q]$, which, according to Proposition 3.2, is logically equivalent to p_1 (observe that p_2 is equivalent to the semi-Horn formula $\top \rightarrow p_2$).

2. $T_2 = \{q \rightarrow (p_1 \vee p_2)\}$. We have that $\text{SNC}(q; T_2; \{p_1, p_2\})$ is defined by the formula $\exists q. [(q \rightarrow (p_1 \vee p_2)) \wedge q]$, which, according to Proposition 3.2, is logically equivalent to $(p_1 \vee p_2)$.

Condition $\text{SNC}(q; T_2; \{p_1\})$ is defined by the formula $\exists q \exists p_2. [(q \rightarrow (p_1 \vee p_2)) \wedge q]$, which is logically equivalent to \top .

3. $T_3 = \{(p \wedge q) \rightarrow s\}$. The formula $\text{SNC}(p \wedge q; T_3; \{s\})$ is equivalent to $\exists p, q. [(p \wedge q) \rightarrow s] \wedge (p \wedge q)$, which, according to Proposition 3.2, is logically equivalent to s .

Observe that we work with formulas more directly than proposed in Lin's approach, where a new proposition has to be introduced together with an additional condition that the new proposition is equivalent to the formula in question.

■

In summary, propositional snc 's or wsc 's can be generated for any propositional formula and theory. In the case that the conjunction of both is in semi-Horn form, this can be done more efficiently. These results subsume those of Lin [Lin, 2000] in the sense that the full propositional language is covered and we work directly with propositional formulas rather than propositional atoms.

4 The First-Order Case

In this section, we generalize the results in section 3 to the first-order case using primarily the same techniques, but with quantification over relational symbols. The following lemma can be proved similarly to Lemma 3.1. The deduction theorem for first-order logic is applicable, since the theories are assumed to be closed.

Lemma 4.1 For any formula α , any set of relation symbols P and a closed³ theory Th :

1. the strongest necessary condition $\text{SNC}(\alpha; Th; P)$ is defined by $\exists \bar{\Phi}. [Th \wedge \alpha]$,
2. the weakest sufficient condition $\text{WSC}(\alpha; Th; P)$ is defined by $\forall \bar{\Phi}. [Th \rightarrow \alpha]$,

where $\bar{\Phi}$ consists of all relation symbols appearing in Th and α but not in P . ■

Observe that a second-order quantifier over the relational variable $\bar{\Phi}$ can be eliminated from any semi-Horn formula w.r.t. $\bar{\Phi}$. In such a case the resulting formula is a fixpoint formula. If the formula is non-recursive, then the resulting formula is a first-order formula. The input formula can also be a

³In fact, it suffices to assume that the set of free variables of Th is disjoint from the set of free variables of α .

conjunction of semi-Horn formulas of the form (1) or a conjunction of semi-Horn formulas of the form (2). On the other hand, one should be aware that in other cases the reduction is not guaranteed. Thus the elimination of second-order quantifiers is guaranteed for any formula of the form $\exists \bar{\Phi}. [Th \wedge \alpha]$, where $Th \wedge \alpha$ is a conjunction of semi-Horn formulas w.r.t. all relational variables in $\bar{\Phi}$.⁴ Observe also, that in the case when an input formula is a conjunction of semi-Horn formulas of the form (1) or a conjunction of formulas of the form (2), the length of the resulting formula is, in the worst case, $O(n^2)$, where n is the size of the input formula.

Example 4.2 Consider the following examples

1. $T_4 = \{\forall x. [Ab(x) \rightarrow (Bird(x) \wedge \neg Flies(x))]\}$. Consider the strongest necessary condition $\text{SNC}(Ab(z); T_4; \{Bird, Flies\})$. According to Lemma 4.1, it is equivalent to

$$\exists Ab. [\forall x. (Ab(x) \rightarrow (Bird(x) \wedge \neg Flies(x))) \wedge Ab(z)]. \quad (9)$$

By Lemma 2.2, formula (9) is equivalent to $(Bird(z) \wedge \neg Flies(z))$.

2. $T_5 = \{\forall x. [Parent(x) \rightarrow \exists z. (Father(x, z) \vee Mother(x, z))]\}$. Consider the strongest necessary condition $\text{SNC}(Parent(y); T_5; \{Mother\})$. According to Lemma 4.1, it is equivalent to

$$\exists Parent, Father. [\forall x. (Parent(x) \rightarrow \exists z. (Father(x, z) \vee Mother(x, z)) \wedge Parent(y)]. \quad (10)$$

In this case, formula (10) is not in the form required in Lemma 2.2, but the DLS algorithm eliminates the second-order quantifiers and results in the equivalent formula \top , which is the required strongest necessary condition. Consider now $\text{SNC}(Parent(y) \wedge \forall u, v. (\neg Father(u, v))); T_5; \{Mother\}$. It is equivalent to

$$\exists Parent, Father. [\forall x. (Parent(x) \rightarrow \exists z. (Father(x, z) \vee Mother(x, z)) \wedge Parent(y) \wedge \forall u, v. (\neg Father(u, v))], \quad (11)$$

i.e. after eliminating second-order quantifiers, to $\exists z. Mother(y, z)$.

■

In summary, for the non-recursive semi-Horn fragment of first-order logic, the snc or wsc for a formula α and theory T is guaranteed to be reducible to compact first-order formulas. For the recursive case, the snc 's and wsc 's are guaranteed to be reducible to fixpoint formulas. In the context of databases, this case is still tractable. The techniques may still be used for the full first-order case, but neither reduction nor complexity results are guaranteed, although the algorithm will always terminate.

⁴For universal quantification, $\forall \bar{\Phi}. A$, one simply negates the formula $(\exists \bar{\Phi}. \neg A)$, and assuming $\neg A$ can be put in to semi-Horn form, one eliminates the existential quantifiers and negates the result.

5 Applications

In this section, we demonstrate the use of the techniques by applying them to a number of potentially useful application areas.

5.1 Communicating Agents

Agents communicating, e.g. via the Internet, have to use the same language to understand each other. This is similar or related to computing interpolants.

Assume an agent A wants to ask a query Q to agent B . Suppose the query can be asked using terms \bar{R}, \bar{S} such that the terms from \bar{S} are unknown for agent B . Let $T(\bar{R}, \bar{S})$ be a theory describing some relationships between \bar{R} and \bar{S} . It is then natural for agent A to first compute the strongest necessary condition $\text{SNC}(Q; T(\bar{R}, \bar{S}); \bar{R})$ with the target language restricted to \bar{R} and then to replace the original query by the computed condition. The new query might not be as good as the previous one, but is the best that can be asked. The following example illustrates the idea.

Example 5.1 Assume an agent A wants to select from a database all persons x such that $\text{High}(x) \wedge \text{Silny}(x)$ holds. Assume further, that both agents know the terms High and Sound . Unfortunately, the database agent does not know the term Silny .⁵ Suppose, further that A lives in a world in which the condition $\forall y. [\text{Silny}(y) \rightarrow \text{Sound}(y)]$ holds. It is then natural for A to consider

$$\text{SNC}(\text{High}(x) \wedge \text{Silny}(x); \forall y. [\text{Silny}(y) \rightarrow \text{Sound}(y)]; \{\text{High}, \text{Sound}\})$$

to be the best query that can be asked. According to Lemma 4.1 this condition is equivalent to:

$$\exists \text{Silny}. [\forall y. [\text{Silny}(y) \rightarrow \text{Sound}(y)] \wedge \text{High}(x) \wedge \text{Silny}(x),$$

which, by a simple application of Theorem 2.2, is equivalent to $\text{High}(x) \wedge \text{Sound}(x)$. ■

5.2 Theory Approximation

The concept of approximating more complex theories by simpler theories has been studied in [Kautz and Selman, 1996; Cadoli, 1995], mainly in the context of approximating arbitrary propositional theories by propositional Horn clauses. The concept of approximate theories is also discussed in [McCarthy, 2000]. Now, observe that strongest necessary and weakest sufficient conditions provide us with approximations of theories expressed in a richer language by theories expressed in a simpler language.

The approach by Lin in [Lin, 2000] allows one only to approximate simple concepts on the propositional level. The generalization we introduce allows us to approximate any finite propositional and first-order theory which is semi-Horn w.r.t. the eliminated propositions or relational symbols.

In the following example, considered in [Kautz and Selman, 1996] approximating general clauses by Horn clauses results in the exponential blow up of the number of clauses.

⁵In Polish “Silny” means “Strong”, but the database agent does not know the Polish language.

We shall show, that the use of the notion of strongest necessary condition can substantially reduce the complexity of reasoning.

Example 5.2 In [Kautz and Selman, 1996] the following clauses, denoted by T , are considered:

$$(\text{CompSci} \wedge \text{Phil} \wedge \text{Psych}) \rightarrow \text{CogSci} \quad (12)$$

$$\text{ReadsMcCarthy} \rightarrow (\text{CompSci} \vee \text{CogSci}) \quad (13)$$

$$\text{ReadsDennett} \rightarrow (\text{Phil} \vee \text{CogSci}) \quad (14)$$

$$\text{ReadsKosslyn} \rightarrow (\text{Psych} \vee \text{CogSci}) \quad (15)$$

and reasoning with this theory was found to be quite complicated. On the other hand, one would like to check, for instance, whether a computer scientist who reads Dennett and Kosslyn is also a cognitive scientist. Reasoning by cases, suggested in [Kautz and Selman, 1996], shows that this is the case. One can, however, substantially reduce the theory and make the reasoning more efficient. In the first step one can notice that notions Phil and Psych are not in the considered claim, thus might appear redundant in the reasoning process. On the other hand, these notions appear in disjunctions in clauses (14) and (15). We then consider

$$\text{SNC}(\text{CompSci} \wedge \text{ReadsDennett} \wedge \text{ReadsKosslyn}; T; -\{\text{Phil}, \text{Psych}\}), \quad (16)$$

where $-\{\text{Phil}, \text{Psych}\}$ denotes all symbols in the language, other than Phil and Psych . After some simple calculations one obtains the following formula equivalent to (16):

$$(13) \wedge [\text{CompSci} \wedge \text{ReadsDennett} \wedge \text{ReadsKosslyn}] \wedge [(\text{CompSci} \wedge (\text{ReadsDennett} \wedge \neg \text{CogSci})) \wedge (\text{ReadsKosslyn} \wedge \neg \text{CogSci})] \rightarrow \text{CogSci} \quad (17)$$

which easily reduces to

$$(13) \wedge \text{CompSci} \wedge \text{ReadsDennett} \wedge \text{ReadsKosslyn} \wedge (\neg \text{CogSci} \rightarrow \text{CogSci}). \quad (18)$$

Thus the strongest necessary condition for the formula

$$\text{CompSci} \wedge \text{ReadsDennett} \wedge \text{ReadsKosslyn}$$

implies CogSci and, consequently, the formula also implies CogSci .

Assume that one wants to calculate the weakest sufficient condition of being a computer scientist in terms of $\{\text{ReadsDennett}, \text{ReadsKosslyn}, \text{ReadsMcCarthy}, \text{CogSci}\}$. We then consider

$$\text{WSC}(\text{CompSci}; T; -\{\text{Phil}, \text{Psych}, \text{CompSci}\}). \quad (19)$$

After eliminating quantifiers over $\text{Phil}, \text{Psych}, \text{CompSci}$ from the second-order formulation of the wsc, one obtains the following formula equivalent to (19):

$$\text{ReadsMcCarthy} \wedge \neg \text{CogSci}.$$

Thus the weakest condition that, together with theory T , guarantees that a person is a computer scientist is that the person reads McCarthy and is not a cognitive scientist. ■

5.3 Abduction

The weakest sufficient condition corresponds to a weakest abduction, as noticed in [Lin, 2000].

Example 5.3 Consider theory

$$T = \{\forall x.[HasWheels(x) \rightarrow CanMove(x)], \\ \forall x.[Car(x) \rightarrow HasWheels(x)]\}.$$

Assume one wants to check whether an object can move. There are three interesting cases:

1. to assume that the target language is $\{HasWheels\}$ and consider

$$WSC(CanMove(x); T; \{HasWheels\}),$$

which is equivalent to

$$\forall CanMove, Car. [\bigwedge T \rightarrow CanMove(x)]$$

2. to assume that the target language is $\{Car\}$ and consider

$$WSC(CanMove(x); T; \{Car\}),$$

which is equivalent to

$$\forall HasWheels, Car. [\bigwedge T \rightarrow CanMove(x)]$$

3. to assume that the target language is $\{HasWheels, Car\}$ and consider

$$WSC(CanMove(x); T; \{HasWheels, Car\}),$$

which is equivalent to

$$\forall CanMove. [\bigwedge T \rightarrow CanMove(x)].$$

After eliminating second-order quantifiers we obtain the following results:

1. $WSC(CanMove(x); T; \{HasWheels\}) \equiv HasWheels(x)$
2. $WSC(CanMove(x); T; \{Car\}) \equiv Car(x)$
3. $WSC(CanMove(x); T; \{HasWheels, Car\}) \equiv \forall x.[Car(x) \rightarrow HasWheels(x)] \rightarrow HasWheels(x)$.

The first two conditions are rather obvious. The third one might seem a bit strange, but observe that $\forall x.[Car(x) \rightarrow HasWheels(x)]$ is an axiom of theory T . Thus, in the third case, we have that

$$WSC(CanMove(x); T; \{HasWheels, Car\}) \equiv HasWheels(x).$$

■

5.4 Generating Successor State Axioms

Example 5.4 Consider the problem of generating successor state axioms in a robot domain. This problem, in the propositional framework, is considered in [Lin, 2000]. On the other hand, a first-order formulation is much more natural and compact. We thus apply first-order logic rather than the propositional calculus and introduce the following relations, instead of propositions as considered in [Lin, 2000]⁶:

⁶Note that even this formalization can be generalized further for more than one action and transition, but we retain the correspondence to the original example for clarity.

- $Move(o, i, j)$ - the robot is performing the action of moving the object o from location i to location j
- $At(o, i)$ - initially, the object o is in the location i
- $At1(o, j)$ - after the action $Move(o, i, j)$, the object o is in location j
- $AtR(i)$ - initially, the robot is at location i
- $AtR1(j)$ - after the action $Move(o, i, j)$, the robot is at location j
- $H(o)$ - initially, the robot is holding the object o
- $H1(o)$ - after the action $Move(o, i, j)$, the robot is holding the object o .

Assume that the background theory contains the following axioms, abbreviated by T :

$$\forall o.(At(o, 1)) \wedge \forall o.(\neg At(o, 2)),$$

$$\forall o.[H(o) \equiv H1(o)],$$

$$\forall o, i, j. [(AtR(i) \wedge At(o, i) \wedge H(o) \wedge Move(o, i, j)) \rightarrow (AtR1(j) \wedge At1(o, j))].$$

The goal is to find the weakest sufficient condition on the initial situation ensuring that the formula $At1(package, 2)$ holds. Thus we consider

$$WSC(At1(package, 2); T; \{H, At, AtR, Move\}).$$

The approach we propose is based on the observation that

$$WSC(At1(package, 2); T; \{H, At, AtR\}) \equiv \forall H1 \forall At1 \forall AtR1. (\bigwedge T \rightarrow At1(package, 2))$$

After some simple calculations which can be performed automatically using the DLS algorithm we ascertain that $WSC(At1(package, 2); T; \{H, At, AtR, Move\})$ is equivalent to:

$$[\forall o. At(o, 1) \wedge \forall o. \neg At(o, 2)] \rightarrow [H(package) \wedge$$

$$\exists i.(AtR(i) \wedge At(package, i) \wedge Move(package, i, 2))],$$

which, in the presence of axioms of theory T reduces to:

$$[H(package) \wedge$$

$$\exists i.(AtR(i) \wedge At(package, i) \wedge Move(package, i, 2))] \quad (20)$$

and, since $At(package, i)$ holds in the theory T only for i equal to 1, formula (20) reduces to:

$$H(package) \wedge AtR(1) \wedge Move(package, 1, 2).$$

Thus, the weakest condition on the initial state, making sure that after the execution of an action the package is in location 2, expresses the requirement that the robot is in location 1, holds the package and that it executes the action of moving the package from location 1 to location 2. ■

6 Forgetting and Quantifier Elimination

Forgetting is considered in [Lin and Reiter, 1994; Lin, 2000] as an important technique for database progression and computing wsc's and snc's.⁷ Given a theory T and a relation

⁷The forgetting operator for propositional logic is well-known in the literature as *eliminant* (see [Brown, 1990]).

symbol P , forgetting about P in T results in a theory with a vocabulary not containing P , but entailing the same set of sentences that are irrelevant to P . Observe that forgetting is simply a second-order quantification, as shown in [Lin and Reiter, 1994]. Namely,

$$\text{forget}(\phi; P) = \exists P.\phi(P).$$

It is no surprise then that forgetting is not always reducible to first-order logic. On the other hand, due to Theorem 2.2, second-order quantifiers can often be eliminated resulting in a fixpoint or first-order formula.

Consider the following example.

Example 6.1 Let T consist of the following two axioms:

$$\begin{aligned} \forall x, y. [Mother(x, y) \rightarrow \exists z. Father(z, y)], \\ \forall x, y. [Father(x, y) \rightarrow Parent(x, y)]. \end{aligned}$$

Forgetting about $Father$ results in the second-order formula $\exists Father. \bigwedge T$ which, according to Theorem 2.2, is equivalent to:

$$\forall x, y. [Mother(x, y) \rightarrow \exists z. Parent(z, y)].$$

■

7 Conclusions

Using Lin's work as a starting point, we have provided new definitions for weakest sufficient and strongest necessary conditions in terms of 2nd-order formulas and provided the basis for algorithms which are guaranteed to automatically generate wsc's and snc's for both the propositional case and a non-trivial fragment of the first-order case. For the general propositional case, propositional snc's and wsc's are always generated using the techniques and for the semi-Horn fragment are always generated efficiently. For the first-order case restricted to the non-recursive semi-Horn fragment, reduction of wsc's and snc's to first-order formulas is always guaranteed and can be done efficiently. For the first-order case restricted to the recursive semi-Horn fragment, reduction to fixpoint formulas is always guaranteed and can be done efficiently. For the general first-order case, the techniques can also be applied, but reductions are not always guaranteed, even though the algorithm will always terminate.

This work generalizes that of Lin which only deals with the propositional case and it provides more direct methods for generating snc's and wsc's via syntactic manipulation. We have also demonstrated the potential use of this idea and these techniques by applying them to a number of interesting applications. Finally, we have re-interpreted the notion of forgetting in terms of quantifier elimination, shown how our techniques can be applied to a first-order version of forgetting and applied the technique to generation of successor-state axioms in restricted first-order situation calculus based action theories.

As a final observation, the quantifier elimination algorithm considered here has been implemented as an extension to the original DLS algorithm described in [Doherty et al., 1997], for both the propositional and 1st-order cases.

References

- [Abiteboul et al., 1996] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Pub. Co., 1996.
- [Brown, 1990] F.M. Brown. *Boolean Reasoning*. Kluwer Academic Publishers, Dordrecht, 1990.
- [Cadoli, 1995] M. Cadoli. *Tractable Reasoning in Artificial Intelligence*, volume 941 of *LNAI*. Springer-Verlag, Berlin Heidelberg, 1995.
- [Dijkstra, 1976] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.
- [Doherty et al., 1996] P. Doherty, W. Lukaszewicz, and A. Szalas. A reduction result for circumscribed semi-Horn formulas. *Fundamenta Informaticae*, 28(3-4):261–271, 1996.
- [Doherty et al., 1997] P. Doherty, W. Lukaszewicz, and A. Szalas. Computing circumscription revisited. *Journal of Automated Reasoning*, 18(3):297–336, 1997.
- [Doherty et al., 1998] P. Doherty, W. Lukaszewicz, and A. Szalas. General domain circumscription and its effective reductions. *Fundamenta Informaticae*, 36(1):23–55, 1998.
- [Ebbinghaus and Flum, 1995] H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, Heidelberg, 1995.
- [Kautz and Selman, 1996] H. Kautz and B. Selman. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.
- [Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it! In R. Greiner and D. Subramanian, editors, *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159, Menlo Park, Ca., 1994. AAAI.
- [Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92(1-2):131–167, 1997.
- [Lin, 2000] F. Lin. On strongest necessary and weakest sufficient conditions. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000*, pages 167–175, 2000.
- [McCarthy, 2000] J. McCarthy. Approximate objects and approximate theories. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000*, pages 519–526, 2000.
- [Nonnengart and Szalas, 1998] A. Nonnengart and A. Szalas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In E. Orłowska, editor, *Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa*, volume 24 of *Studies in Fuzziness and Soft Computing*, pages 307–328. Springer Physica-Verlag, 1998.
- [Nonnengart et al., 1999] A. Nonnengart, H.J. Ohlbach, and A. Szalas. Elimination of predicate quantifiers. In H.J. Ohlbach and U. Reyle, editors, *Logic, Language and Reasoning. Essays in Honor of Dov Gabbay, Part I*, pages 159–181. Kluwer, 1999.

KNOWLEDGE REPRESENTATION AND REASONING

DESCRIPTION LOGICS

Identification Constraints and Functional Dependencies in Description Logics

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica

Università di Roma “La Sapienza”

Via Salaria 113, 00198 Roma, Italy

{calvanese,degiamoco,lenzerini}@dis.uniroma1.it

Abstract

DLR is an expressive Description Logic (DL) with n -ary relations, particularly suited for modeling database schemas. Although *DLR* has constituted one of the crucial steps for applying DL technology to data management, there is one important aspect of database schemas that DLs, including *DLR*, do not capture yet, namely the notion of identification constraints and functional dependencies. In this paper we introduce a DL which extends *DLR* and fully captures the semantics of such constraints, and we address the problem of reasoning in such a logic. We show that, verifying knowledge base satisfiability and logical implication in the presence of identification constraints and nonunary functional dependencies can be done in EXPTIME, thus with the same worst-case computational complexity as for plain *DLR*. We also show that adding just unary functional dependencies to *DLR* leads to undecidability.

1 Introduction

In the last years, Description Logics (DLs) have been successfully applied to data management [Borgida, 1995; Kirk *et al.*, 1995; Calvanese *et al.*, 1998b; 1999]. One of the basic ideas behind applying DLs to data management is that database schemas can be expressed as DL knowledge bases, so that DL reasoning techniques can be used in several ways to reason about the schema. In [Calvanese *et al.*, 1998a; 1998b], a very expressive DL with n -ary relations, called *DLR*, is introduced, and it is shown how database schemas can be captured by this logic. Also, suitable mechanisms for expressing queries over *DLR* schemas have been added, and techniques for reasoning over queries have been designed. Notably, the investigation on *DLR* has led to the design of new DL systems effectively implementing powerful reasoning techniques [Horrocks *et al.*, 1999].

Although the above mentioned work has been the crucial step for applying DL technology to data management, there is one important aspect of database schemas that DLs, including *DLR*, do not capture yet, namely the notion of identification constraints and functional dependencies. Identification constraints (also called keys) are used to state that a certain set

of properties uniquely identifies the instances of a concept. A functional dependency on a relation is used to impose that a combination of a given set of attributes functionally determines another attribute of the relation. It is easy to see that functional dependencies can be used to model keys of a relation, i.e., attributes that are sufficient to identify tuples of the relation. Both types of constraints are commonly used in database design and data management.

The question addressed in this paper is as follows: can we add identification constraints and *non-unary* functional dependencies to *DLR* and still have EXPTIME associated reasoning techniques? Somewhat surprisingly, we answer positively to the question, by illustrating an approach that allows us to incorporate both types of constraints in *DLR*. In particular, we adapt the *DLR* reasoning algorithm in such a way that reasoning on a *DLR* schema with both types of constraints and with ABoxes, can be done with the same worst-case computational complexity as for the case of plain *DLR*. Also, the proposed technique can be incorporated into present DL systems, such as the one described in [Horrocks *et al.*, 1999]. We also show that adding to *DLR* *unary* functional dependencies leads to undecidability of reasoning. Observe, however, that the presence of such functional dependencies is typically considered as an indication of bad schema design in databases.

Both identification constraints and functional dependencies have been extensively investigated in the database literature (see [Abiteboul *et al.*, 1995], Chapters 8, 9). However, database models lack the kinds of constraints expressible in expressive DLs, and therefore none of the results developed in the context of databases can be used to solve our problem.

In the last years, there have been some attempts to add identification constraints to DLs. In [Calvanese *et al.*, 1995], these constraints are modeled by means of special primitive concepts in an expressive DL, and it is shown that this mechanism allows some inference on keys to be carried on. The limitation of this approach is that several interesting semantic properties of keys are not represented in the knowledge base. In [Borgida and Weddell, 1997; Khizder *et al.*, 2001], a general mechanism is proposed for modeling path functional dependencies, and a sound and complete inference system for reasoning on such constraints is presented. Path functional dependencies are sufficiently expressive to model both identification constraints and functional dependencies. How-

ever, the DLs considered are limited in expressiveness. In particular, union, negation, number restrictions, general inclusion axioms, inverse roles and n -ary relations are not part of the considered language, and therefore useful properties of database schemas cannot be represented. The proposal presented in this paper fully captures the semantics of both identification constraints and functional dependencies in an expressive DL with all the above features.

The paper is organized as follows. In Section 2, we recall the DL \mathcal{DLR} . In Section 3, we illustrate the mechanism for specifying identification constraints and functional dependencies in \mathcal{DLR} knowledge bases. In Section 4, we discuss the modeling power of the resulting logic, called \mathcal{DLR}_{ifd} . In Section 5, we describe how we can extend the \mathcal{DLR} reasoning technique in order to take the new types of constraints into account, and in Section 6 we show that minor extensions of \mathcal{DLR}_{ifd} leads to undecidability of reasoning. Finally, Section 7 concludes the paper.

2 Description Logic \mathcal{DLR}

We focus on the Description Logic \mathcal{DLR} , which is able to capture a great variety of data models with many forms of constraints [Calvanese *et al.*, 1998a; 1999]. The basic elements of \mathcal{DLR} are *concepts* (unary relations), and *n -ary relations*.

We assume to deal with a finite set of atomic relations (having arity between 2 and n_{max}) and atomic concepts, denoted by P and A , respectively. We use R to denote arbitrary relations and C to denote arbitrary concepts, respectively built according to the following syntax:

$$\begin{aligned} R &::= \top_n \mid P \mid (i/n:C) \mid \neg R \mid R_1 \sqcap R_2 \\ C &::= \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid (\leq k [i]R) \end{aligned}$$

where n denotes the *arity* of the relations P , R , R_1 , and R_2 , i denotes a component of a relation, i.e., an integer between 1 and n , and k denotes a non-negative integer. Observe that we consider only concepts and relations that are *well-typed*, which means that: (i) only relations of the same arity n are combined to form expressions of type $R_1 \sqcap R_2$ (which inherit the arity n); (ii) $i \leq n$ whenever i denotes a component of a relation of arity n .

We introduce the following abbreviations: \perp for $\neg \top_1$; $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$; $C_1 \Rightarrow C_2$ for $\neg C_1 \sqcup C_2$; $(\geq k [i]R)$ for $\neg(\leq k-1 [i]R)$; $\exists [i]R$ for $(\geq 1 [i]R)$; $\forall [i]R$ for $\neg \exists [i] \neg R$. Moreover, we abbreviate $(i/n:C)$ with $(i:C)$ when n is clear from the context.

A \mathcal{DLR} TBox is constituted by a finite set of *inclusion assertions*, where each assertion has one of the forms:

$$C_1 \sqsubseteq C_2 \quad R_1 \sqsubseteq R_2$$

with R_1 and R_2 of the same arity.

The semantics of \mathcal{DLR} is specified as follows. An *interpretation* \mathcal{I} is constituted by an *interpretation domain* $\Delta^{\mathcal{I}}$, and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each relation R of arity n a subset $R^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$ such that the conditions in Figure 1 are satisfied. In the figure, $t[i]$ denotes the i -th component of tuple t . Observe that, the “ \neg ” constructor on relations is

$\top_n^{\mathcal{I}}$	\subseteq	$(\Delta^{\mathcal{I}})^n$
$P^{\mathcal{I}}$	\subseteq	$\top_n^{\mathcal{I}}$
$(i/n:C)^{\mathcal{I}}$	$=$	$\{t \in \top_n^{\mathcal{I}} \mid t[i] \in C^{\mathcal{I}}\}$
$(\neg R)^{\mathcal{I}}$	$=$	$\top_n^{\mathcal{I}} \setminus R^{\mathcal{I}}$
$(R_1 \sqcap R_2)^{\mathcal{I}}$	$=$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
$\top_1^{\mathcal{I}}$	$=$	$\Delta^{\mathcal{I}}$
$A^{\mathcal{I}}$	\subseteq	$\Delta^{\mathcal{I}}$
$(\neg C)^{\mathcal{I}}$	$=$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$(C_1 \sqcap C_2)^{\mathcal{I}}$	$=$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
$(\leq k [i]R)^{\mathcal{I}}$	$=$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{t \in R^{\mathcal{I}} \mid t[i] = a\} \leq k\}$

Figure 1: Semantic rules for \mathcal{DLR} (P , R , R_1 , and R_2 have arity n , and $\#\sigma$ denotes the cardinality of the set σ)

used to express difference of relations, and not the complement [Calvanese *et al.*, 1998a]. An interpretation \mathcal{I} satisfies an assertion $C_1 \sqsubseteq C_2$ (resp., $R_1 \sqsubseteq R_2$) if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ (resp., $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$).

We introduce a generalized form of \mathcal{DLR} ABox. We consider an alphabet of new symbols, called *Skolem constants* (*sk-constants*). Intuitively, an sk-constant denotes an individual in an interpretation, in such a way that different sk-constants may denote the same individual.

A *generalized \mathcal{DLR} ABox* (or simply ABox in the following) is constituted by a finite set of assertions, called ABox assertions, of the following types:

$$C(x) \quad R(x_1, \dots, x_n) \quad x \neq y \quad x = y$$

where R is a relation of arity n , and x, y, x_1, \dots, x_n are sk-constants.

The notion of interpretation is extended so as to assign to each sk-constant x an individual $x^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation \mathcal{I} satisfies

- $C(x)$ if $x^{\mathcal{I}} \in C^{\mathcal{I}}$;
- $R(x_1, \dots, x_n)$ if $(x_1^{\mathcal{I}}, \dots, x_n^{\mathcal{I}}) \in R^{\mathcal{I}}$;
- $x \neq y$ if $x^{\mathcal{I}} \neq y^{\mathcal{I}}$;
- $x = y$ if $x^{\mathcal{I}} = y^{\mathcal{I}}$.

If \mathcal{T} is a \mathcal{DLR} TBox, and \mathcal{A} is a \mathcal{DLR} ABox of the above form, then $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ is called a *\mathcal{DLR} knowledge base*. An interpretation is a *model* of \mathcal{K} if it satisfies every assertion in \mathcal{K} . A knowledge base \mathcal{K} is *satisfiable* if it has a model. An assertion α (either an inclusion, or an ABox assertion) is *logically implied* by \mathcal{K} if all models of \mathcal{K} satisfy α .

Logical implication and knowledge base satisfiability are mutually reducible to each other. For one direction, \mathcal{K} is unsatisfiable iff $\mathcal{K} \models \top_1 \sqsubseteq \perp$. For the converse direction, it is possible to show that $\mathcal{K} \models C_1 \sqsubseteq C_2$ iff $\mathcal{K} \cup \{\top_1 \sqsubseteq \exists[1](P \sqcap (2:C_1 \sqcap \neg C_2))\}$ is unsatisfiable, where P is a new binary relation. Similarly, $\mathcal{K} \models R_1 \sqsubseteq R_2$ iff $\mathcal{K} \cup \{\top_1 \sqsubseteq \exists[1](P \sqcap (2:\exists[1](C_1 \sqcap \neg C_2)))\}$ is unsatisfiable, where again P is a new binary relation. Finally, $\mathcal{K} \models C(\alpha)$ iff $\mathcal{K} \cup \{\neg C(\alpha)\}$ is unsatisfiable.

It follows from the results in [Calvanese *et al.*, 1998a], that checking a \mathcal{DLR} knowledge base for satisfiability is EXPTIME-complete.

3 Identification and Functional Dependency Assertions

We extend \mathcal{DLR} with identification constraints and functional dependencies. The resulting DL, called \mathcal{DLR}_{ifd} , allows one to express these constraints through new kinds of assertions in the TBox.

An *identification assertion* on a concept has the form:

$$(\text{id } C [i_1]R_1, \dots, [i_h]R_h)$$

where C is a concept, each R_j is a relation, and each i_j denotes one component of R_j . Intuitively, such an assertion states that two instances of C cannot agree on the participation to R_1, \dots, R_h via components i_1, \dots, i_h , respectively.

A *functional dependency assertion* on a relation has the form:

$$(\text{fd } R i_1, \dots, i_h \rightarrow j)$$

where R is a relation, $h \geq 2$, and i_1, \dots, i_h, j denote components of R . The assertion imposes that two tuples of R that agree on the components i_1, \dots, i_h , agree also on the component j .

We assign semantics to these assertions by defining when an interpretation satisfies them. Specifically:

- An interpretation \mathcal{I} *satisfies* the assertion $(\text{id } C [i_1]R_1, \dots, [i_h]R_h)$ if, for all $a, b \in C^{\mathcal{I}}$ and for all $t_1, s_1 \in R_1^{\mathcal{I}}, \dots, t_h, s_h \in R_h^{\mathcal{I}}$, we have that:

$$\left. \begin{array}{l} a = t_1[i_1] = \dots = t_h[i_h], \\ b = s_1[i_1] = \dots = s_h[i_h], \\ t_j[i_j] = s_j[i_j], \text{ for } j \in \{1, \dots, h\}, \\ \text{and for } i \neq i_j \end{array} \right\} \text{implies } a = b$$

- An interpretation \mathcal{I} *satisfies* the assertion $(\text{fd } R i_1, \dots, i_h \rightarrow j)$ if, for all $t, s \in R^{\mathcal{I}}$, we have that:

$$t[i_1] = s[i_1], \dots, t[i_h] = s[i_h] \text{ implies } t[j] = s[j]$$

A \mathcal{DLR}_{ifd} knowledge base is a set $\mathcal{K} = \mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$ of assertions, where $\mathcal{T} \cup \mathcal{A}$ is a \mathcal{DLR} knowledge base and \mathcal{F} is a set of identification and functional dependency assertions.

Note that unary functional dependencies (i.e., functional dependencies with $h = 1$) are ruled out in \mathcal{DLR}_{ifd} . We will come to this in Section 6. Note also that the right hand side of a functional dependency contains a single element. However, this is not a limitation, because any functional dependency with more than one element in the right hand side can always be split into several dependencies of the above form. Also, to verify whether a functional dependency with more than one element in the right hand side is logically implied by a \mathcal{DLR}_{ifd} knowledge base, it suffices to verify whether each of the functional dependencies in which it can be split, is logically implied by the knowledge base.

4 Modeling in \mathcal{DLR}_{ifd}

\mathcal{DLR}_{ifd} captures database schemas expressed in several data models. For example, Entity-Relationship schemas can be represented already in \mathcal{DLR} , by modeling each entity as a concept, and each relationship as a relation [Calvanese *et al.*,

1998b; 1999]. Attributes of entities are modeled by means of binary relations, and single-valued or mandatory attributes are expressible through the use of number restrictions. Attributes of relationships can be modeled in several ways, for instance through special $(n + 1)$ -ary relations, where n is the arity of the relationship. Also, integrity constraints such as is-a, cardinality, existence, and typing constraints are expressible by means of inclusion assertions. Finally, unary keys (keys constituted by a single attribute) can be modeled through number restrictions. Non-unary keys cannot be represented in \mathcal{DLR} , while they are obviously expressible in \mathcal{DLR}_{ifd} .

Example 1 Suppose that Person and University are concepts, EnrolledIn is a binary relation between Person and University, and StudentCode is an (optional) attribute (modeled as a binary relation) of Person associating to each student (a person who is enrolled in a university) a code that is unique in the context of the university in which she is enrolled. Such a situation can be represented by the following \mathcal{DLR}_{ifd} TBox:

$$\begin{aligned} \text{EnrolledIn} &\sqsubseteq (1 : \text{Person}) \sqcap (2 : \text{University}) \\ \text{StudentCode} &\sqsubseteq (1 : \text{Person}) \sqcap (2 : \text{String}) \\ \text{Person} &\sqsubseteq (\leq 1 [1] \text{StudentCode}) \\ (\text{id } \text{Person } [1] \text{StudentCode}, [1] \text{EnrolledIn}) \end{aligned}$$

Note that, the notion of student is modeled by the concept $\text{Person} \sqcap \exists[1] \text{EnrolledIn}$ and, in the conceptual modeling terminology, this concept is a weak entity, i.e., part of its identifier is external through the relationship EnrolledIn.

We additionally want to model the notion of exam in our application. An exam is a relationship involving a student, a course, a professor, and a grade. In an exam, the combination of student and course functionally determines both the professor, and the grade. This can be represented by adding the following assertions to the TBox:

$$\begin{aligned} \text{Exam} &\sqsubseteq (1 : (\text{Person} \sqcap \exists[1] \text{EnrolledIn})) \sqcap \\ &\quad (2 : \text{Course}) \sqcap (3 : \text{Person}) \sqcap (4 : \text{Grade}) \\ (\text{fd } \text{Exam } 1, 2 \rightarrow 3) \quad (\text{fd } \text{Exam } 1, 2 \rightarrow 4) \quad \blacksquare \end{aligned}$$

Observe that generally, in conceptual data models, if an attribute (or a relationship) L is part of a key for an entity E , then in the database schema it must be the case that E has a single and mandatory participation in L , i.e., each instance of E has exactly one associated value for L [Abiteboul *et al.*, 1995]. This is not required in \mathcal{DLR}_{ifd} (but can be asserted when needed), where one can define an attribute as part of a key of an entity, even if the attribute is multi-valued or optional.

We have mentioned that unary functional dependencies are not allowed in \mathcal{DLR}_{ifd} . However, this limitation does not prevent one from defining unary keys for relations. Indeed, the fact that component i is a key for the relation R can already be expressed in \mathcal{DLR} by means of the assertion:

$$\top_1 \sqsubseteq (\leq 1 [i]R)$$

The above observation also implies that functional dependencies in the context of binary relations, which are by definition unary, are expressible in \mathcal{DLR}_{ifd} . Indeed, such functional dependencies correspond to key constraints, which are

expressible as specified above. For example, the functional dependency $1 \rightarrow 2$ in the context of the binary relation R can be expressed by specifying that component 1 is a key for R . Thus, the only functional dependencies that are not admitted in \mathcal{DLR}_{ifd} are unary functional dependencies in the context of non-binary relations. This is because they lead to undecidability of reasoning, as shown in Section 6. Note also, that the presence of such functional dependencies is considered as an indication of bad design in the framework of the relational data model (see [Abiteboul *et al.*, 1995], Chapter 11). In fact, a unary functional dependency in the context of an n -ary relation (with $n > 2$) represents a hidden relationship between the arguments of the relation, which may cause several modeling problems.

The possibility of defining identification constraints substantially enriches the modeling power of DLs. In particular, it is possible to show that, even if only binary relations are allowed in a DL, then the use of identification constraints permits simulating the presence of n -ary relations in such a logic. For example, a relation with arity 3 can be modeled by means of a concept and 3 binary relations. Number restrictions are used to state that every instance of the concept participates in exactly one instance of the binary relation, and a suitable identification assertion states that the combination of the three binary relations form a key for the concept. Obviously, \mathcal{DLR}_{ifd} further increases the modeling power by allowing the explicit use of n -ary relations, and the possibility of imposing functional dependencies in the context of relations.

5 Reasoning on \mathcal{DLR}_{ifd}

First of all we observe that, when reasoning in \mathcal{DLR}_{ifd} , identification assertions of the form $(\mathbf{id} C [i]R)$, where R is a binary relation, are equivalent to \mathcal{DLR} assertions $\top \sqsubseteq (\leq 1 [j](R \sqcap (i : C)))$, where $j = 2$ if $i = 1$, and $j = 1$ if $i = 2$. Hence, in the following, without loss of generality, we will not consider such identification assertions.

Next we show that we can reduce logical implication in \mathcal{DLR}_{ifd} to knowledge base satisfiability. As already observed in Section 2, logical implication of inclusion and ABox assertions can be reduced to knowledge base satisfiability. We show that the same can be done also for identification and functional dependency assertions.

Given an identification assertion

$$\kappa = (\mathbf{id} C [i_1]R_1, \dots, [i_h]R_h)$$

we define the ABox \mathcal{A}_κ constituted by the following assertions:

- $C(x)$, $C(y)$, and $x \neq y$, where x and y are new sk-constants;
- $R_j(t_j)$ and $R_j(s_j)$, with $j \in \{1, \dots, h\}$, where t_j and s_j are tuples of new sk-constants with $t_j[i_j] = x$, $s_j[i_j] = y$, and $t_j[i] = s_j[i]$ for $i \neq i_j$.

Similarly, for a functional dependency assertion

$$\kappa = (\mathbf{fd} R i_1, \dots, i_h \rightarrow j)$$

we define \mathcal{A}_κ constituted by the following assertions:

- $R(t)$, $R(s)$, and $t[j] \neq s[j]$, where t and s are tuples of new sk-constants with $t[i_j] = s[i_j]$, for $j \in \{1, \dots, h\}$.

From the semantics of identification and functional dependency assertions it is immediate to see that \mathcal{A}_κ provides a concrete counterexample to κ . Hence it follows that, given a \mathcal{DLR}_{ifd} knowledge base \mathcal{K} , $\mathcal{K} \models \kappa$ if and only if $\mathcal{K} \cup \mathcal{A}_\kappa$ is unsatisfiable.

Theorem 2 *Logical implication in \mathcal{DLR}_{ifd} can be reduced to knowledge base satisfiability.*

We now present a reasoning procedure for knowledge base satisfiability in \mathcal{DLR}_{ifd} .

\mathcal{DLR}_{ifd} TBoxes (which in fact are \mathcal{DLR} TBoxes since they do not include identification and functional dependency assertions) have the tree-model property [Calvanese *et al.*, 1998a], which is true for most DLs. In particular, if a \mathcal{DLR} TBox admits a model, it also admits a model which has the structure of a tree, where nodes are either objects or (reified) tuples, and edges connect tuples to their components. Observe that in such models identification and functional dependency assertions (which in \mathcal{DLR}_{ifd} are non-unary) are trivially satisfied, since there cannot be two tuples agreeing on more than one component. As an immediate consequence we have that, given a \mathcal{DLR}_{ifd} TBox \mathcal{T} and a set of identification and functional dependency assertions \mathcal{F} , $\mathcal{T} \cup \mathcal{F}$ is satisfiable iff \mathcal{T} is so. This implies that, in absence of an ABox, logical implication of inclusion assertions can be verified without considering identification and functional dependency assertions at all.

When we add an ABox, then we may still restrict the attention to models that have the structure of a tree, except for a cluster of objects representing the sk-constants in the ABox (see again [Calvanese *et al.*, 1998a]¹). We call such models *clustered tree models*. On such models, identification and functional dependency assertions are always satisfied, except possibly for the cluster of sk-constants. Hence we can concentrate on verifying such assertions on the objects and tuples appearing in the ABox only.

Given a \mathcal{DLR}_{ifd} knowledge base \mathcal{K} , we define a *saturation* of \mathcal{K} as an ABox \mathcal{A}_s constructed as follows:

- for each sk-constant x occurring in \mathcal{K} , and for each identification assertion $(\mathbf{id} C [i_1]R_1, \dots, [i_h]R_h)$ in \mathcal{K} , \mathcal{A}_s contains either $C(x)$ or $\neg C(x)$;
- for each tuple t of sk-constants occurring in an assertion $R(t)$ of \mathcal{K} ,
 - for each identification assertion $(\mathbf{id} C [i_1]R_1, \dots, [i_h]R_h)$ in \mathcal{K} , for each R_j ($j \in \{1, \dots, h\}$) having the arity of R , \mathcal{A}_s contains either $R_j(t)$ or $\neg R_j(t)$,
 - for each functional dependency assertion $(\mathbf{fd} R' i_1, \dots, i_h \rightarrow j)$ in \mathcal{K} , such that R' has the arity of R , \mathcal{A}_s contains either $R'(t)$ or $\neg R'(t)$;

¹In fact, [Calvanese *et al.*, 1998a] makes use of inclusion assertions involving nominals, i.e., concepts having a single instance. It is immediate to verify that such inclusion assertions correspond to the kind of generalized ABoxes we adopt here.

- for each pair of sk-constants x and y occurring in \mathcal{K} , \mathcal{A}_s contains either $x = y$ or $x \neq y$.

Note that the size of a saturation is polynomial in the size of \mathcal{K} . Note also that there are many (actually, an exponential number) of different saturations of \mathcal{K} , one for each possible set of choices in the items above.

On a saturation one can immediately verify whether an identification or functional dependency assertion of \mathcal{K} is *violated*. Indeed, for all sk-constants and tuples of sk-constants appearing in the saturation, membership or non-membership in the relevant relations and concepts appearing in the assertions that could be violated is explicitly asserted (after substituting each sk-constant with a representative of its equivalence class according to the equalities). Hence, it suffices to verify whether the semantic condition of the assertion is violated, considering relations and concepts appearing in the assertion as primitives. Once such a check on the identification and functional dependencies is done, it remains to verify whether \mathcal{A}_s is consistent with the other assertions of \mathcal{K} .

Theorem 3 A \mathcal{DLR}_{ifd} knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$ is satisfiable if and only if there exists a saturation \mathcal{A}_s of \mathcal{K} that does not violate the identification and functional dependency assertions in \mathcal{K} and such that the \mathcal{DLR} knowledge base $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$ is satisfiable.

Proof (sketch). “ \Leftarrow ” Assume that there exists a saturation \mathcal{A}_s that does not violate the identification and functional dependency assertions in \mathcal{K} and such that $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$ is satisfiable. Then there exists a clustered tree model of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$, and such interpretation is also a model of \mathcal{K} .

“ \Rightarrow ” Assume that \mathcal{K} is satisfiable. Then from a model \mathcal{I} of \mathcal{K} one can directly construct a saturation for which \mathcal{I} satisfies all assertions. \square

The above result provides us with an upper bound for reasoning in \mathcal{DLR}_{ifd} , matching the lower bound holding already for \mathcal{DLR} .

Theorem 4 Satisfiability and logical implication in \mathcal{DLR}_{ifd} are EXPTIME-complete.

Proof (sketch). By Theorem 2, logical implication in \mathcal{DLR}_{ifd} reduces to knowledge base satisfiability in \mathcal{DLR}_{ifd} . By Theorem 3, satisfiability of a \mathcal{DLR}_{ifd} knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$ reduces to solving a (possibly exponential) number of tests, where each test involves one saturation \mathcal{A}_s and consists in directly verifying all identification and functional dependency assertions in \mathcal{A}_s (a polynomial step) and checking the satisfiability of the \mathcal{DLR} knowledge base $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$ (an exponential step). \square

6 Unary functional dependencies in \mathcal{DLR}_{ifd}

One might wonder whether the method described in the previous works also for unary functional dependencies. Actually, this is not the case since unary functional dependencies are *not trivially satisfied* in tree structured interpretations. For example, it may happen that two tuples of a relation R agree on say component 1 and not component 2, and therefore violate the functional dependency (fd $R \ 1 \rightarrow 2$).

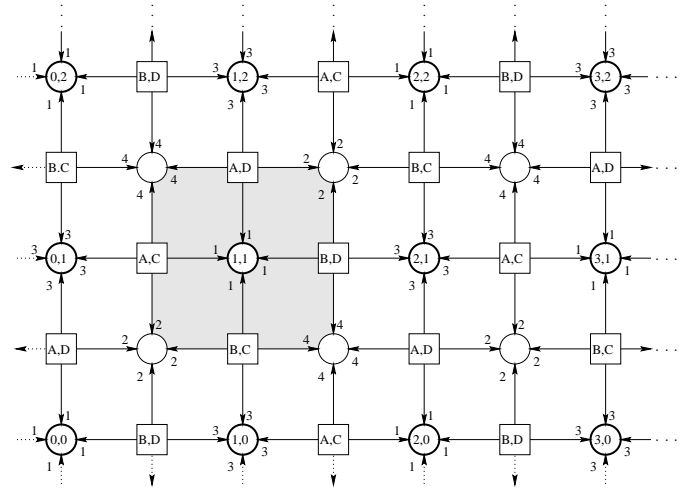


Figure 2: Grid structure enforced by \mathcal{K}_T

Indeed, we show that if we allow for unary functional dependencies, then reasoning in \mathcal{DLR}_{ifd} becomes undecidable. To do so we exhibit a reduction from the unconstrained quadrant tiling problem [van Emde Boas, 1997], which consists in deciding whether the first quadrant of the integer grid can be tiled using a finite set of square tile types in such a way that adjacent tiles respect adjacency conditions. Tiling problems are well suited to show undecidability of variants of description and dynamic logics [van Emde Boas, 1997; Baader and Sattler, 1999]. The crux of the undecidability proof consists in enforcing that the tiles lie on an integer grid. Once the grid structure is enforced, it is typically easy to impose the adjacency conditions on the tiles. In our case we exploit unary functional dependencies to construct the grid.

Formally, a *tiling system* is a triple $T = (\mathcal{D}, \mathcal{H}, \mathcal{V})$ where \mathcal{D} is a finite set of elements representing tile types and \mathcal{H} and \mathcal{V} are two binary relations over \mathcal{D} . The *unconstrained quadrant tiling problem* consists in verifying the existence of a tiling consistent with T , i.e., a mapping τ from $\mathbb{N} \times \mathbb{N}$ to \mathcal{D} such that $(\tau(i, j), \tau(i + 1, j)) \in \mathcal{H}$ and $(\tau(i, j), \tau(i, j + 1)) \in \mathcal{V}$, for $i, j \in \mathbb{N}$. Such a problem is undecidable, more precisely Π_0^0 -complete [Berger, 1966; van Emde Boas, 1997].

From a tiling system $T = (\mathcal{D}, \mathcal{H}, \mathcal{V})$ we construct a \mathcal{DLR}_{ifd} knowledge base \mathcal{K}_T as follows. The basic idea is to enforce the grid structure shown in Figure 2, where squares represent tuples of arity 4 and circles represent objects. Each object depicted using a bold circle and labeled i, j represents the node (i, j) of the grid. Numbers labeling arrows represent components of tuples. A pair of letters X, Y inside a tuple represents the fact that the tuple is an instance of the relations X and Y . In particular we use four relations of arity 4: A, B, C , and D . The gray box in the figure represents how a tile would be placed in such a grid.

We enforce the grid structure by means of the following assertions in \mathcal{K}_T :

$$\exists[i] \top_4 \sqsubseteq \exists[i](A \sqcap C) \sqcap \exists[i](A \sqcap D) \sqcap \exists[i](B \sqcap C) \sqcap \exists[i](B \sqcap D) \quad \text{for } i \in \{1, \dots, 4\}$$

(fd $A \ 2 \rightarrow 3$) (fd $B \ 2 \rightarrow 3$) (fd $C \ 2 \rightarrow 1$) (fd $D \ 1 \rightarrow 2$)
 (fd $A \ 4 \rightarrow 1$) (fd $B \ 4 \rightarrow 1$) (fd $C \ 3 \rightarrow 4$) (fd $D \ 4 \rightarrow 3$)

We enforce the adjacency conditions on the tiles of the first quadrant by using one concept for each tile type in \mathcal{D} and introducing in \mathcal{K}_T the following assertions: for each $D_i \in \mathcal{D}$

$$D_i \sqsubseteq \forall[1](B \sqcap D \Rightarrow (3 : \bigsqcup_{(D_i, D_j) \in \mathcal{H}} D_j)) \sqcap \\ \forall[3](A \sqcap C \Rightarrow (1 : \bigsqcup_{(D_i, D_j) \in \mathcal{H}} D_j)) \sqcap \\ \forall[1](A \sqcap D \Rightarrow (3 : \bigsqcup_{(D_i, D_j) \in \mathcal{V}} D_j)) \sqcap \\ \forall[3](B \sqcap C \Rightarrow (1 : \bigsqcup_{(D_i, D_j) \in \mathcal{V}} D_j))$$

Finally, to represent the origin of the tiling we use the concept $C_0 = (\exists[1] \top_4) \sqcap \bigsqcup_{D_i \in \mathcal{D}} D_i$. Then the tiling problem associated to T admits a solution if and only if $\mathcal{K}_T \not\models C_0 \sqsubseteq \perp$. Indeed, from a tiling consistent with T one obtains immediately a model of \mathcal{K}_T with an object satisfying C_0 . Conversely, from a model \mathcal{I} of \mathcal{K}_T with $C_0^{\mathcal{I}} \neq \emptyset$, we can construct a tiling consistent with T . The first set of assertions impose that a portion of \mathcal{I} has exactly the structure depicted in Figure 2 (observe that not the whole model necessarily has a grid structure, but only a portion corresponding to the first quadrant). The second set of assertions impose on such a portion only that instances of concepts representing tile types respect the adjacency conditions. As a consequence of such a reduction we obtain the following result.

Theorem 5 *Knowledge base satisfiability (and thus logical implication) in \mathcal{DLR}_{ifd} extended with unary functional dependencies is undecidable.*

The reduction above can be easily modified to show that, if we allow for *nominals* [Tobies, 2000], then even n -ary functional dependencies lead to undecidability. For example, we may use one nominal o and relations of arity 5 instead of 4. Then we can force the fifth component of all tuples to be the object o by means of the assertion $\top_5 \sqsubseteq (5 : o)$ and we can enforce the grid structure as above, by adding component 5 to the antecedent of the functional dependencies above (thus getting binary functional dependencies). Observe that, since all tuples agree on component 5, such binary functional dependencies are actually mimicking the unary dependencies in the previous reduction.

7 Conclusions

\mathcal{DLR}_{ifd} extends \mathcal{DLR} by fully capturing identification constraints on concepts and functional dependencies on relations. We have shown that reasoning in the presence of such constraints remains EXPTIME decidable. We have also shown that adding to \mathcal{DLR} just unary functional dependencies on non-binary relations, usually considered an indication of bad design in data modeling, leads to undecidability.

The approach presented in this paper can be extended in several ways. For example, our technique can be directly applied to reasoning in \mathcal{DLR}_{reg} extended with identification and functional dependency constraints. Moreover, we are working on the following extensions: (i) using chaining in specifying identification constraints, in the spirit of [Borgida and Weddell, 1997]; (ii) introducing a notion of functional dependency between properties of concepts; (iii) query containment and query answering using views in the presence of identification constraints and functional dependencies.

References

- [Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachusetts, 1995.
- [Baader and Sattler, 1999] F. Baader and U. Sattler. Expressive number restrictions in description logics. *J. of Log. and Comp.*, 9(3):319–350, 1999.
- [Berger, 1966] R. Berger. The undecidability of the domino problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.
- [Borgida and Weddell, 1997] A. Borgida and G. E. Weddell. Adding uniqueness constraints to description logics (preliminary report). In *Proc. of DOOD'97*, pages 85–102, 1997.
- [Borgida, 1995] A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, 1995.
- [Calvanese *et al.*, 1995] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of DOOD'95*, volume 1013 of *LNCS*, pages 229–246. Springer-Verlag, 1995.
- [Calvanese *et al.*, 1998a] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158, 1998.
- [Calvanese *et al.*, 1998b] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR'98*, pages 2–13, 1998.
- [Calvanese *et al.*, 1999] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.
- [Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, number 1705 in *LNAI*, pages 161–180. Springer-Verlag, 1999.
- [Khizder *et al.*, 2001] V. L. Khizder, D. Toman, and G. E. Weddell. On decidability and complexity of description logics with uniqueness constraints. In *Proc. of ICDT 2001*, 2001.
- [Kirk *et al.*, 1995] T. Kirk, A. Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, pages 85–91, 1995.
- [Tobies, 2000] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.
- [van Emde Boas, 1997] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture notes in pure and applied mathematics*, pages 331–363. Marcel Dekker Inc., 1997.

High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study

Volker Haarslev and Ralf Möller

University of Hamburg, Computer Science Department

Vogt-Kölln-Str. 30, 22527 Hamburg, Germany

<http://kogs-www.informatik.uni-hamburg.de/~haarslev|moeller/>

Abstract

We present an empirical analysis of optimization techniques devised to speed up the so-called TBox classification supported by description logic systems which have to deal with very large knowledge bases (e.g. containing more than 100,000 concept introduction axioms). These techniques are integrated into the RACE architecture which implements a TBox and ABox reasoner for the description logic \mathcal{ALCNH}_{R+} . The described techniques consist of adaptations of previously known as well as new optimization techniques for efficiently coping with these kinds of very large knowledge bases. The empirical results presented in this paper are based on experiences with an ontology for the Unified Medical Language System and demonstrate a considerable runtime improvement. They also indicate that appropriate description logic systems based on sound and complete algorithms can be particularly useful for very large knowledge bases.

1 Introduction

In application projects it is often necessary to deal with knowledge bases containing a very large number of axioms. Furthermore, many applications require only a special kind of axioms, so-called concept introduction axioms. Usually it has been argued that only systems based on incomplete calculi can deal with knowledge bases containing more than 100,000 axioms of this kind. In this contribution we present an empirical analysis of optimization techniques devised to improve the performance of description logic systems applied to this kind of knowledge bases. The analysis is based on the RACE¹ architecture [Haarslev and Möller, 2000a] which supports inference services for the description logic \mathcal{ALCNH}_{R+} [Haarslev and Möller, 2000b].²

As example knowledge bases we consider reconstructions of important parts of the UMLS (Unified Medical Language System) [McCray and Nelson, 1995] by using description logic representation techniques. The reconstruction is described in [Hahn *et al.*, 1999; Schulz and Hahn, 2000] and

¹URL: <http://kogs-www.informatik.uni-hamburg.de/~race/>

²A convenient pronunciation of \mathcal{ALCNH}_{R+} is ALC-nature.

Syntax	Semantics
Concepts	
A	$A^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}}$ (A is a concept name)
$\neg C$	$\Delta_{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists R.C$	$\{a \in \Delta_{\mathcal{I}} \mid \exists b \in \Delta_{\mathcal{I}} : (a, b) \in R^{\mathcal{I}}, b \in C^{\mathcal{I}}\}$
$\forall R.C$	$\{a \in \Delta_{\mathcal{I}} \mid \forall b \in \Delta_{\mathcal{I}} : (a, b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$
$\exists_{\geq n} S$	$\{a \in \Delta_{\mathcal{I}} \mid \ \{b \in \Delta_{\mathcal{I}} \mid (a, b) \in S^{\mathcal{I}}\}\ \geq n\}$
$\exists_{\leq m} S$	$\{a \in \Delta_{\mathcal{I}} \mid \ \{b \in \Delta_{\mathcal{I}} \mid (a, b) \in S^{\mathcal{I}}\}\ \leq m\}$
Roles	
R	$R^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$

$\|\cdot\|$ denotes the cardinality of a set, $S \in \mathcal{S}$, $n, m \in \mathbb{N}$, $n > 0$.

TBox Axioms		ABox Assertions	
Syntax	Satisfied if	Syntax	Satisfied if
$R \in \mathcal{T}$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	$(a, b) : R$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$		

Figure 1: Syntax and Semantics of \mathcal{ALCNH}_{R+} .

introduces a specific encoding scheme that uses several concept introduction axioms which represent subset as well as composition aspects of conceptual descriptions (words) mentioned in the UMLS metathesaurus.

The paper is structured as follows. We first introduce the syntax and semantics of \mathcal{ALCNH}_{R+} and characterize the form of axioms occurring in the UMLS knowledge bases. Afterwards we describe the following five complementary optimization techniques: (1) topological sorting for achieving a quasi definition order; (2) a method to cluster siblings in huge taxonomies; (3) a technique for efficiently addressing so-called domain and range restrictions; (4) exploitation of implicit disjointness declarations; (5) subset/superset caching for increasing runtime performance and reducing memory requirements. The effectiveness of these techniques is assessed using the empirical results obtained from processing the UMLS knowledge bases.

We briefly introduce the description logic (DL) \mathcal{ALCNH}_{R+} [Haarslev and Möller, 2000b] (see the tables in Figure 1) using a standard Tarski-style semantics. \mathcal{ALCNH}_{R+} extends the basic description logic \mathcal{ALC} by

role hierarchies, transitively closed roles (denoted by the set T in Figure 1), and number restrictions. Note that the combination of transitive roles and role hierarchies implies the expressivity of so-called general inclusion axioms (GCIs). The concept name \top is used as an abbreviation for $C \sqcup \neg C$.

If R, S are role names, then $R \sqsubseteq S$ is called a *role inclusion* axiom. A *role hierarchy* \mathcal{R} is defined by a finite set of role inclusion axioms. The concept language of \mathcal{ALCNH}_{R^+} syntactically restricts the combinability of number restrictions and transitive roles due to a known undecidability result in case of an unrestricted syntax [Horrocks *et al.*, 1999]. Only simple roles may occur in number restrictions. Roles are called *simple* (denotes by the set S in Figure 1) if they are neither transitive nor have a transitive role as descendant.

If C and D are concept terms, then $C \sqsubseteq D$ (*generalized concept inclusion* or *GCI*) is a terminological axiom. A finite set $\mathcal{T}_{\mathcal{R}}$ of terminological axioms is called a *terminology* or *TBox* w.r.t. to a given role hierarchy \mathcal{R} .³ A terminological axiom of the form $A \sqsubseteq C$ is called a *concept introduction axiom* and C is called the *primitive (concept) definition* of A if A is a concept name which occurs only once on the left hand side of the axioms contained in a TBox \mathcal{T} . A pair of GCIs of the form $\{A \sqsubseteq C, C \sqsubseteq A\}$ (abbreviated as $A \doteq C$) is called a *concept definition axiom* and C is called the (*concept*) *definition* of A if A is a concept name which occurs only once on the left hand side of all axioms contained in a TBox \mathcal{T} .

An *ABox* \mathcal{A} is a finite set of assertional axioms as defined in Figure 1. The ABox consistency problem is to decide whether a given ABox \mathcal{A} is consistent w.r.t. a TBox \mathcal{T} and a role hierarchy \mathcal{R} . An ABox \mathcal{A} is consistent iff there exists a model \mathcal{I} that satisfies all axioms in \mathcal{T} and all assertions in \mathcal{A} . Subsumption between concepts can be reduced to concept satisfiability since C *subsumes* D iff the concept $\neg C \sqcap D$ is unsatisfiable. Satisfiability of concepts can be reduced to ABox consistency as follows: A concept C is satisfiable iff the ABox $\{a : C\}$ is consistent.

The DL reconstruction of important parts of the UMLS introduces a specific scheme where a set of concept introduction axioms is used to represent subset as well as composition aspects of conceptual descriptions (words) mentioned in the UMLS metathesaurus. For instance, for the notion of a ‘heart’, the following concept introduction axioms for heart structures (suffix ‘s’), heart parts (suffix ‘p’) and heart entities (no suffix) are declared (see [Schulz and Hahn, 2000] for details):

```

ana_heart  $\sqsubseteq$  ana_heart_s  $\sqcap$  ana_hollow_viscus  $\sqcap$ 
  umls_body_part_organ_or_organ_component
ana_heart_s  $\sqsubseteq$  ana_hollow_viscus_s  $\sqcap$ 
  ana_cardiovascular_system_p
ana_heart_p  $\sqsubseteq$   $\neg$ ana_heart  $\sqcap$  ana_heart_s  $\sqcap$ 
   $\exists_{\geq 1}$  anatomical_part_of_ana_heart

```

Note the implicit disjointness declared between `ana_heart_p` and `ana_heart`. The following role axiom is generated as well.

```

anatomical_part_of_ana_heart  $\sqsubseteq$ 
  anatomical_part_of_ana_hollow_viscus

```

³The reference to \mathcal{R} is omitted in the following if we use \mathcal{T} .

It is beyond the scope of this paper to discuss the pros and cons of specific modeling techniques used in the UMLS reconstruction. In the next section, optimization techniques for efficiently dealing with these kinds of knowledge bases are presented.

2 Optimization Techniques

Modern DL systems such as RACE offer at least two standard inference services for concept names occurring in TBoxes: classification and coherence checking. Classification is the process of computing the most-specific subsumption relationships between all concept names mentioned in a TBox \mathcal{T} . The result is often referred to as the *taxonomy* of \mathcal{T} and gives for each concept name two sets of concept names listing its ‘parents’ (direct subsumers) and ‘children’ (direct subsumees). Coherence checking determines all concept names which are unsatisfiable.

Expressive DLs such as \mathcal{ALCNH}_{R^+} allow GCIs which can considerably slow down consistency tests. A *true* GCI is a GCI of the form $C \sqsubseteq D$ where C is not a name and $C \sqsubseteq D$ is not part of a pair representing a concept definition axiom. A standard technique (GCI absorption) [Horrocks and Tobies, 2000] which is also part of the RACE architecture performs a TBox transformation which precedes classification and coherence checking. The axioms in a TBox are transformed in a way that true GCIs can be absorbed into (primitive) concept definitions which can be efficiently dealt with by a technique called *lazy unfolding* (e.g. see [Baader *et al.*, 1994]). Lazy unfolding dynamically expands a concept name by its (primitive) definition during an ABox consistency test. The true GCIs remaining after the GCI absorption are referred to as *global axioms*.

Our findings indicate that state-of-the-art techniques currently employed for fast classification of TBoxes have to be extended in order to cope with very large knowledge bases of the above-mentioned kind. In the following we describe these extensions.

2.1 Topological Sorting for Quasi Definition Order

For TBox classification the RACE architecture employs the techniques introduced in [Baader *et al.*, 1994]. The parents and children of a certain concept name are computed in so-called ‘top search’ and ‘bottom search’ traversal phases, respectively. These phases can be illustrated with the following example. Assume a new concept name A_i has to be inserted into an existing taxonomy. The top search phase traverses the taxonomy from the top node (\top) via the set of children and checks whether A_i is subsumed by a concept node. Basically, if A_i is subsumed by a node A_j , then the children of A_j are traversed. The top search phase determines the set of parents of A_i . When the top search phase for A_i has been finished, the bottom search phase for A_i analogously traverses the taxonomy from the bottom node via the set of parents of a node. The bottom search phase determines the set of children of A_i .

Using the marking and propagation techniques described in [Baader *et al.*, 1994] the search space for traversals can usually be pruned considerably. It is always advantageous to

avoid as many traversals as possible since they require the use of “expensive” subsumption tests. This is even more important for very large TBoxes.

Let us assume, a TBox to be classified can be transformed such that no global axioms remain but cyclic (primitive) concept definitions may exist. According to [Baader *et al.*, 1994] we assume that a concept name A ‘directly uses’ a concept name B if B occurs in the (primitive) definition of A. The relation ‘uses’ is the transitive closure of ‘directly uses.’ If A uses B then B comes before A in the so-called definition order. For acyclic TBoxes (i.e. the uses relation is irreflexive) containing concept introduction axioms only, the set of concept names can be processed in definition order, i.e. a concept name is not classified until all the concept names used in its (primitive) definition are classified. In this case the set of children of a concept name to be inserted consists only of the bottom concept. Thus, if concept names are classified in definition order, the bottom search phase can safely be omitted for concept names which have only a primitive definition [Baader *et al.*, 1994].

In order to avoid the bottom search phase it is possible to impose a syntactical restriction on TBoxes for less expressive DLs, i.e. to accept only concept axioms in definition order, i.e. the (primitive) definitions do not include forward references to concept names not yet introduced. However, for an expressive DL such as \mathcal{ALCNH}_{R^+} , which offers cyclic axioms and GCIs, in general, the bottom search phase cannot be skipped.

The UMLS TBoxes contain many forward references occurring in value (e.g. $\forall R.C$) and existential restrictions (e.g. $\exists R.C$). Thus, the definition order of concept names has to be computed in a preprocessing step. As a refinement we devised a slightly less strict notion of definition order which works more efficiently. We assume a relation ‘directly refers to’ similar to ‘directly uses’ but with references occurring in the scope of quantifiers not considered. This simplification reduces the overhead caused by computing the ‘directly refers to’ relation. It is correct since subsumption between concept names with primitive definitions cannot be caused via quantifiers occurring in the concept definitions. Again ‘refers to’ is the transitive closure of ‘directly refers to’. The ‘refers to’ relation induces a partial order relation on sets of concept names. All concept names involved in a cycle become members of one set while the remaining concept names form singleton sets. A topological sorting algorithm is used to serialize the partial order such that a total order between sets of concept names is defined. This serialization is called a *quasi definition order*.

During the classification of a TBox with RACE the sets of concept names are processed in quasi definition order. For each singleton set whose member has a primitive concept definition, the bottom search can be skipped. Let $A \sqsubseteq C$ be a concept introduction axiom and A is to be inserted into the taxonomy. The ‘refers to’ relation and the quasi definition order serialization ensure that either all concept names that are potential subsumees of A are inserted after A has been inserted into the subsumption lattice or the bottom search is indeed performed. The quasi definition order is conservative w.r.t. the potential subsumers (note that \mathcal{ALCNH}_{R^+} does not

support inverse roles). Moreover, in a basic subsumption test the subsumption lattice under construction is never referred to. Thus, strict definition order classification is not necessary.

Note that in [Baader *et al.*, 1994] no experiments are discussed that involve the computation of a serialization given a TBox with axioms not already in (strict) definition order. Topological sorting is of order $n + e$ where e is the number of given ‘refers to’ relationships. Thus, we have approximately $O(n \log n)$ steps while the bottom search procedure requires $O(n^2)$ steps in the worst case.

2.2 Adaptive Clustering in TBoxes

Experiments with the UMLS TBoxes showed that the well-known techniques described in [Baader *et al.*, 1994] exhibit considerable performance deficiencies in case of (rather flat) taxonomies where some nodes have a large number of children. Therefore, in the RACE architecture a special clustering technique is employed.

If the top search phase finds a node (e.g. A) with more than θ children, the θ children are grouped into a *bucket* (e.g. A_{new}), i.e. a (virtual) concept definition axiom $A_{new} \doteq A_1 \sqcup \dots \sqcup A_\theta$ is assumed and A_{new} is inserted into the subsumption lattice with $\{A\}$ being its parents and $\{A_1 \dots A_\theta\}$ being its children. A_{new} is also referred to as a bucket concept. Note that bucket concepts (e.g. A_{new}) are considered as virtual in the sense that the external interface of RACE hides the names of virtual concepts in a transparent way.

Let us assume, a certain concept name B is to be inserted and a node A with its children $\{A_1, \dots, A_\theta\}$ is encountered during the top search phase. Instead of testing for each child A_i ($i \in \{1.. \theta\}$) whether A_i subsumes B, our findings suggest that it is more effective to initially test whether the associated virtual concept definition of A_{new} does not subsume B using the pseudo model merging technique (e.g. see [Horrocks, 1997; Haarslev *et al.*, 2001]) which provides a ‘cheap’ and sound but incomplete non-subsumption test based on pseudo models derived from concept satisfiability tests. Since in most cases no subsumption relation can be found between any A_i and B, one test possibly replaces θ “expensive” but wasted subsumption tests. On the other hand, if a subsumption relation indeed exists, then clustering introduces some overhead. However, since the UMLS TBoxes mostly contain concept introduction axioms, the pseudo model of $\neg A_{new}$ being used for model merging is very simple because the pseudo model basically consists only of a set of negated primitive concept names (see [Haarslev *et al.*, 2001] for further details about pseudo model merging in RACE). The adaptive clustering is designed in a way that it still works even if a quasi definition order cannot be guaranteed (e.g. due to the presence of defined concepts or GCIs). Therefore, a bucket becomes obsolete and has to be removed from a concept node if a member of this bucket gets a different concept node as parent.

For best performance, the number of children to be kept in a bucket should depend on the total number of known children for a concept. However, this can hardly be estimated. Therefore, the following adaptive strategy is used. If more and more concept names are “inserted”

into the subsumption lattice, the number of buckets increases as well. If a new bucket is to be created for a certain node A and there are already σ buckets clustering the children of A , then two buckets (those buckets with the smallest number of children) of A are merged. For instance, merging of the buckets $A_{new} \doteq A_1 \sqcup \dots \sqcup A_n$ and $B_{new} \doteq B_1 \sqcup \dots \sqcup B_m$ means that the bucket A_{new} is “re-defined” as $A_{new} \doteq A_1 \sqcup \dots \sqcup A_n \sqcup B_1 \sqcup \dots \sqcup B_m$ and the bucket B_{new} is reused for the new bucket to be created (see above).⁴ Whether hierarchical clustering techniques lead to performance improvements is subject to further research.

The current evaluation of clustering with buckets uses a setting with $\theta = 10$ and $\sigma = 15$.

2.3 Dealing with Domain and Range Restrictions

Some UMLS TBoxes contain axioms declaring domain and range restrictions for roles. For instance, the *domain* C of a role R can be determined by the axiom $\exists_{\geq 1} R \sqsubseteq C$ and the *range* D by the axiom $\top \sqsubseteq \forall R.D$. Domain restrictions increase the search space during a consistency test since they have to be represented as disjunctions of the form $(\neg \exists_{\geq 1} R) \sqcup C$.

It is possible to transform a domain restriction of the form $\exists_{\geq 1} R \sqsubseteq C$ into an equivalent inclusion axiom of the form $\neg C \sqsubseteq \exists_{\leq 0} R$ and to absorb the transformed axiom if no inclusion axiom of the form $C \sqsubseteq \dots$ exists. However, the transformation is not applicable to the UMLS TBoxes since they contain domain restrictions (e.g. $\exists_{\geq 1} \text{anatomical_part_of_ana_heart} \sqsubseteq \text{ana_heart_p}$) as well as inclusion axioms (e.g. $\text{ana_heart_p} \sqsubseteq \neg \text{ana_heart} \sqcap \text{ana_heart_s} \sqcap \exists_{\geq 1} \text{anatomical_part_of_ana_heart}$) violating the above-mentioned precondition. Hence, in order to apply the topological sorting optimization, it was necessary to incorporate domain restrictions into an ABox consistency test because global axioms may not exist in order to apply the topological sorting technique.

If GCIs representing domain restrictions for roles have been absorbed, they are dealt with by RACE with a generalized kind of lazy unfolding. Whenever a concept of the form $\exists R.D$ or $\exists_{\geq n} R$ is checked for unfolding, it is replaced by $C \sqcap \exists R.D$ or $C \sqcap \exists_{\geq n} R$ if a GCI of the form $\exists_{\geq 1} S \sqsubseteq C$ has been absorbed (R a sub-role of S). This technique is valid since $\exists_{\geq 1} R \sqsubseteq C$ can be represented as the global axiom $\exists_{\leq 0} R \sqcup C$ and the unfolding scheme of RACE guarantees that C is added if an R -successor will be created due to $\exists R.D$ or $\exists_{\geq n} R$. A role assertion $(a, b) : R$ is unfolded to $\{(a, b) : R, a : C\}$. If lazy unfolding is applied, domain restrictions have to be considered w.r.t. the ‘directly refers to’ relationship in a special way.

Note that, in principle, RACE also supports the absorption of GCIs of the form $\neg A \sqsubseteq C_1$ (but only if no concept introduction axiom of the form $A \sqsubseteq C_2$ and no concept definition axiom of the form $A \doteq C_2$ for A exists). Some knowledge bases can only be handled effectively if the absorption of axioms of the form $\neg A \sqsubseteq C_1$ is supported.

⁴Note that due to subsequent merging operations, n and m need not be equal to θ .

In contrast to domain restrictions, range restrictions for roles do not introduce new disjunctions. However, it is always advantageous to keep the number of global axioms to be managed as small as possible. Therefore, GCIs expressing range restrictions for roles are absorbed and concepts of the form $\exists R.C$ or $\exists_{\geq n} R$ are unfolded to $\exists R.C \sqcap \forall S.D$ or $\exists_{\geq n} R \sqcap \forall S.D$ if a GCI of the form $\top \sqsubseteq \forall S.D$ (R a sub-role of S) has been absorbed. A role assertion $(a, b) : R$ is unfolded to $\{(a, b) : R, b : D\}$.

2.4 Exploiting Disjointness Declarations

As has been discussed in [Baader *et al.*, 1994], it is important to derive told subsumers⁵ for each concept name for marking and propagation processes. Besides told subsumers, RACE exploits also the set of “told disjointness⁶”. In the ‘heart’ example presented above, *ana_heart* is computed as a told disjoint concept of *ana_heart_p* by examining the related concept introduction axioms. If it is known that a concept B is a subsumer of a concept A then A cannot be a subsumee of the told disjointness of B . This kind of information is recorded (and propagated) with appropriate non-subsumer marks (see [Baader *et al.*, 1994] for details about marking and propagation operations) in order to prune the search space for traversals causing subsumption tests. Exploiting disjointness information has not been investigated in [Baader *et al.*, 1994].

2.5 Caching Policies

RACE supports different caching policies (see also [Haarslev and Möller, 2000a] for caching in RACE). Two types of caches are provided which can be used together or alternatively. Both cache types are accessed via keys constructed from a set of concepts. The first cache (called equal cache) contains entries about the satisfiability status of concept conjunctions already encountered. This cache only returns a hit if the search key exactly matches (i.e. is equal to) the key of a known entry. For instance, the key for a concept conjunction $C_1 \sqcap \dots \sqcap C_n$ is the (ordered) set $\{C_1, \dots, C_n\}$ of concepts.

The second cache type consists of a pair of caches. The subset cache contains only entries for satisfiable concept conjunctions while the superset cache stores unsatisfiable concept conjunctions. These caches support queries concerning already encountered supersets and subsets of a given search key (see also [Hoffmann and Köhler, 1999; Giunchiglia and Tacchella, 2000]). For instance, if the subset (satisfiability) cache already contains an entry for the key $\{C_1, C_2, C_3\}$ and is queried with the key $\{C_1, C_3\}$, it returns a hit, i.e. the conjunction $C_1 \sqcap C_3$ is also satisfiable. Analogously, if the superset (unsatisfiability) cache already contains an entry for the key $\{D_2, D_3, D_5\}$ and is queried with the key $\{D_1, \dots, D_6\}$, it returns a hit, i.e. the conjunction $D_1 \sqcap \dots \sqcap D_6$ is also unsatisfiable. If the equal cache is enabled, it is the first reference, i.e. only if an equal cache lookup fails, the superset or subset caches are consulted.

⁵For instance, A_1, A_2 are told subsumers of A for a concept introduction axiom $A \sqsubseteq A_1 \sqcap A_2$ if A_1, A_2 are concept names.

⁶For instance, A_1, A_2 are told disjointness of A for a concept introduction axiom $A \sqsubseteq \neg A_1 \sqcap \neg A_2$ if A_1, A_2 are concept names.

3 Empirical Results for UMLS Classifications

The performance of the RACE system is evaluated with different versions of the UMLS knowledge bases. UMLS-1 is a preliminary version whose classification resulted in many unsatisfiable concept names. UMLS-1 contains approximately 100,000 concept names and for almost all of them there exists a concept introduction axiom of the form $A \sqsubseteq C$ where C not equal to \top . In addition, in UMLS-1 80,000 role names are declared. Role names are arranged in a hierarchy.

UMLS-2 is a new version in which the reasons for the inconsistencies have been removed. The version of UMLS-2 we used for our empirical tests uses approximately 160,000 concept names with associated primitive concept definitions and 80,000 hierarchical roles.

Originally, the UMLS knowledge bases have been developed with an incomplete description logic system which does not classify concepts with cyclic definitions (and, in turn, the concepts which use these concepts). Due to the treatment of cycles in the original approach [Schulz and Hahn, 2000], the cycle-causing concepts are placed in so-called `:implies` clauses, i.e. these concepts are not considered in concept satisfiability and subsumption tests. For the same reason, the UMLS reconstruction uses `:implies` for domain and range restrictions of roles, i.e. domain and range restrictions are also not considered in concept satisfiability and subsumption tests.

With RACE, none of these pragmatic distinctions are necessary. However, in order to mimic the original behavior and to test more than one TBox with RACE, for each of the knowledge base versions, UMLS-1 and UMLS-2, three different subversions are generated (indicated with letters a, b and c). Version ‘a’ uses axioms of the style presented above, i.e. the `:implies` parts are omitted for TBox classification (and coherence checking). In version ‘b’ the `:implies` part of the original knowledge base is indeed considered for classification by RACE. Thus, additional axioms of the following form are part of the TBox.

ana_heart $\sqsubseteq \exists \text{has_developmental_fo} . \text{ana_fetal_heart} \sqcap \exists \text{surrounded_by} . \text{ana_pericardium}$

Version ‘c’ is the hardest version. Additional axioms express domain and range restrictions for roles. For instance, the following axioms are included in the TBox for `anatomical_part_of_ana_heart`.

$\exists_{\geq 1} \text{anatomical_part_of_ana_heart} \sqsubseteq \text{ana_heart_p}$

$\top \sqsubseteq \forall \text{anatomical_part_of_ana_heart} . \text{ana_heart}$

Thus, for the performance evaluation we have tested 6 different knowledge bases. All measurements have been performed on a Sun UltraSPARC 2 with 1.4 GByte main memory and Solaris 6. RACE is implemented in ANSI Common Lisp and for the tests Franz Allegro Common Lisp 5.0.1 has been used. The results are as follows.

If the generalized unfolding technique for domain and range restrictions is disabled in RACE, even a small subset (with a size of $\sim 5\%$) of the UMLS TBoxes with GCIs for domain and range restrictions could not be classified within several hours of runtime. Furthermore, the equal cache had to be disabled for all UMLS TBoxes in order to reduce the space requirements during TBox classification.

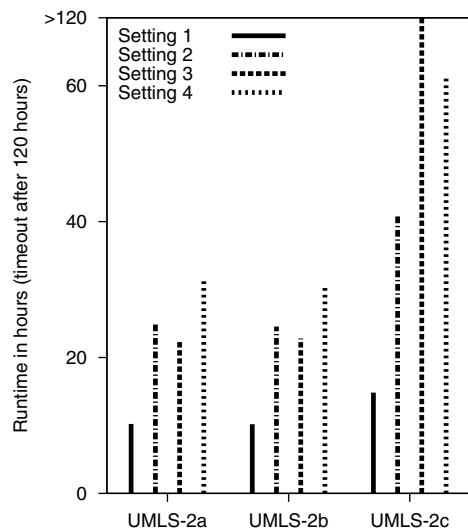


Figure 2: Evaluation of the topological sorting and clustering techniques for UMLS2a-c (see explanation in text).

Without clustering and topological sorting, UMLS-1a can be classified in approximately 11 hours (1636 concept names are recognized as unsatisfiable). With clustering and topological sorting enabled, only 5.5 hours are necessary to compute the same result for UMLS-1a. The second version, UMLS-1b, requires 3.6 hours (with optimization) and 6.1 hours (without optimization). The reason for the enhanced performance with more constraints is that in this version already 47855 concept names are unsatisfiable. With domain and range restrictions we found that even 60246 concept names are unsatisfiable. The computation times with RACE are 3.4 hours (with optimization) and 8.7 hours (without optimization). Up to 500 MBytes of memory are required to compute the classification results. For UMLS-1, checking TBox coherence (see above) requires approximately 10 minutes.

The new second version, UMLS-2, contains an additional part of the UMLS ontology and, therefore, is harder to deal with. Furthermore, there are no unsatisfiable concept names, i.e. classification is much harder because there are much more nodes in the subsumption lattice. In UMLS-1, due to the large number of unsatisfiable concepts, the subsumption lattice is rather small because many concept names “disappear” as synonyms of the bottom concept. For UMLS-2, checking TBox coherence (see above) requires between 15 and 50 minutes (2a: 16 min, 2b: 19 min, 2c: 51 min).

The results for classifying the UMLS-2 TBoxes are shown in Figure 2. A comparison of setting 1 (all optimizations enabled) and setting 2 (clustering disabled) reveals that clustering is a very effective optimization technique for the UMLS-2 TBoxes. The result for setting 3 (topological sorting disabled) and UMLS-2a/b supports the fact that topological sorting is also very effective. The runtime result for setting 3 and UMLS-2c is due to removed buckets (see Section 2.2). This is very likely to happen if topological sorting is disabled and apparently shows a dramatic overhead for UMLS-2c. If, in setting 4, both clustering and topological sorting are dis-

abled, runtimes increase only to a limited extent. Moreover, according to the evaluation results, UMLS-2b does not require more computational resources than UMLS-2a (see the discussion about `:implies` from above). Only the incorporation of domain and range restrictions cause runtimes to increase. For the UMLS-2 TBoxes up to 800 MBytes of memory are required. For other benchmark TBoxes (e.g. Galen [Horrocks, 1997] with approx. 3000 concepts) our results indicate an overhead of $\sim 5\%$ in runtime. This is caused by the presence of many defined concepts and a small average number of concept children in the taxonomy. In summary, the results for the UMLS TBoxes clearly demonstrate that clustering is only effective in conjunction with topological sorting establishing a quasi-definition order.

4 Conclusion

In this paper enhanced optimization techniques which are essential for an initiative towards sound and complete high performance knowledge base classification are presented. Thus, fast classification of very large terminologies which mostly consist of concept introduction axioms now has become possible with description logic systems based on sound and complete algorithms.

A final comment concerning the significance of the UMLS knowledge bases used for the empirical evaluation is appropriate. Even though the UMLS knowledge bases do not make use of defined concepts or arbitrary GCIs, a large number of concept introduction axioms and a possibly large role hierarchy can be called the standard case in many practical applications. Furthermore, some UMLS TBoxes (version 'c') demand the absorption of GCIs expressing domain and range restrictions for roles. Without this new technique even a very small subset of these TBoxes cannot be classified within a reasonable amount of time. If description logics are to be successful in large-scale practical applications, being able to deal with very large knowledge bases such as those based on the UMLS is mandatory.

Even with the above-mentioned optimization techniques, dealing with 160,000 concept names is by no means a trivial task. Large knowledge bases reveal even slightly less than optimal algorithms used for specific subproblems. Thus, experiments with very large knowledge bases also provide feedback concerning lurking performance bottlenecks not becoming apparent when dealing with smaller knowledge bases. To the best of our knowledge, RACE is the only DL system based on sound and complete algorithms that can deal with this kind of very large knowledge bases.

We would like to thank Stefan Schulz for making the UMLS reconstruction available.

References

- [Baader *et al.*, 1994] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making *KRIS* get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.
- [Cohn *et al.*, 2000] A.G. Cohn, F. Giunchiglia, and B. Selman, editors. *Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, Breckenridge, Colorado, USA, April 11-15, 2000, April 2000.
- [Giunchiglia and Tacchella, 2000] E. Giunchiglia and A. Tacchella. A subset-matching size-bounded cache for satisfiability of modal logics. In *Proceedings International Conference Tableaux'2000*, pages 237–251. Springer-Verlag, 2000.
- [Haarslev and Möller, 2000a] V. Haarslev and R. Möller. Consistency testing: The RACE experience. In R. Dyckhoff, editor, *Proceedings, Automated Reasoning with Analytic Tableaux and Related Methods, University of St Andrews, Scotland, 4-7 July, 2000*, pages 57–61. Springer-Verlag, Berlin, April 2000.
- [Haarslev and Möller, 2000b] V. Haarslev and R. Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In Cohn *et al.* [2000], pages 273–284.
- [Haarslev *et al.*, 2001] V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In *Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy*, LNCS. Springer-Verlag, Berlin, June 2001.
- [Hahn *et al.*, 1999] U. Hahn, S. Schulz, and M. Romacker. Part-whole reasoning: A case study in medical ontology engineering. *IEEE Intelligent Systems*, 14(5):59–67, September 1999.
- [Hoffmann and Köhler, 1999] J. Hoffmann and J. Köhler. A new method to query and index sets. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI-99*, pages 462–467. Morgan-Kaufmann Publishers, July 31 – August 6, 1999.
- [Horrocks and Tobies, 2000] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In Cohn *et al.* [2000], pages 285–296.
- [Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, pages 161–180. Springer-Verlag, Berlin, September 1999.
- [Horrocks, 1997] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
- [McCray and Nelson, 1995] A.T. McCray and S.J. Nelson. The representation of meaning in the UMLS. *Methods of Information in Medicine*, 34(1/2):193–201, 1995.
- [Schulz and Hahn, 2000] S. Schulz and U. Hahn. Knowledge engineering by large-scale knowledge reuse – Experience from the medical domain. In Cohn *et al.* [2000], pages 601–610.

KNOWLEDGE REPRESENTATION AND REASONING

COMPLEXITY ANALYSIS

Complexity of Nested Circumscription and Abnormality Theories

Marco Cadoli

Dipartimento di Informatica e Sistemistica
Università di Roma “La Sapienza”
Via Salaria 113, I-00198 Roma, Italy
cadoli@dis.uniroma1.it

Thomas Eiter Georg Gottlob

Institut für Informationssysteme
Technische Universität Wien
Favoritenstrasse 9, A-1040 Wien, Austria
eiter@kr.tuwien.ac.at
gottlob@dbai.tuwien.ac.at

Abstract

We propose \mathcal{L}_{Circ} , an extension of circumscription by allowing propositional combinations and nesting of circumscriptive theories. As shown, Lifschitz’s nested abnormality theories (NATs, introduced in AIJ, 1995) are naturally embedded into this language. We analyze the complexity of \mathcal{L}_{Circ} and NATs, and in particular the effect of nesting. The latter is found a source of complexity, as both formalisms are proved to be PSPACE-complete. We identify cases of lower complexity, including a tractable case. Our results give insight into the “cost” of using \mathcal{L}_{Circ} resp. NATs as a host language for expressing other formalisms, such as narratives.

1 Introduction

Circumscription is a very powerful method for knowledge representation and common-sense reasoning, which has been used for a variety of tasks, including temporal reasoning, diagnosis, and reasoning in inheritance networks. The basic semantical notion underlying circumscription is *minimization* of the extension of selected predicates. This is especially useful when a predicate is meant to represent an *abnormality* condition, e.g., a bird which does not fly. Circumscription is applied to a formula T , either propositional or first-order, and it is used to eliminate some *unintended* models of T .

Since the seminal definition of circumscription in [McCarthy, 1980], several extensions have been proposed (see, e.g., Lifschitz’s survey [1994]), all of them retaining the basic idea of minimization. In this paper, we propose \mathcal{L}_{Circ} , a language which extends propositional circumscription in two important and rather natural ways:

1. on one hand, we allow the *propositional combination* of circumscriptive theories;
2. on the other hand, we allow *nesting* of circumscriptions.

As for the former extension, we claim that it can be useful in several cases. As an example, when different sources of knowledge $Circ(T_1)$ and $Circ(T_2)$ coming from two equally trustable agents who perform circumscription are to be integrated, it seems plausible to take their disjunction, i.e.,

$Circ(T_1) \vee Circ(T_2)$ ¹. In \mathcal{L}_{Circ} , all propositional connectives are allowed.

As for the latter extension, the original idea of *nested abnormality theories* (NATs) has been proposed by Lifschitz [1995], and its usefulness for the formalization of narratives has been shown by Baral *et al.* [1998]. Nesting circumscription is useful for the modularization of a knowledge base. As an example concerning diagnosis of an artifact designed in a modular way, e.g., a car, a piece of knowledge $Circ(T_1)$ may model the intended behavior of a subpart, e.g., the engine. Analogously, $Circ(T_2)$ may model the intended behavior of the electrical part. Further unintended models can be eliminated by taking the circumscription of a suitable propositional combination of $Circ(T_1)$, $Circ(T_2)$, and a plain propositional formula encoding observations.

In this paper, we are mainly concerned with the computational properties of \mathcal{L}_{Circ} and NATs and with their relationship to plain circumscription. In particular, we tackle the following questions:

- Can NATs be embedded into \mathcal{L}_{Circ} ?
- What is the precise complexity of reasoning under nested circumscription? By *reasoning*, we mean both model checking and formula inference from an \mathcal{L}_{Circ} or NAT theory.
- Is there a simple restriction of NATs (and thus of \mathcal{L}_{Circ}) for which some relevant reasoning tasks are feasible in polynomial time?

We are able to give a satisfactory answer to all these questions. In particular, in Section 3, after providing a precise definition of \mathcal{L}_{Circ} , we prove the main results about its complexity: model checking and inference are shown to be PSPACE-complete (the latter even for literals); moreover, complexity is proven to increase w.r.t. the nesting. It appears that nesting, and not propositional combination, is responsible for the increase in complexity. Similar results are proven for NATs in Section 4, where we also prove that every NAT can be easily (and polynomially) translated into a formula of \mathcal{L}_{Circ} .

Given the high complexity of nested circumscription, we look for significant fragments of the languages in which complexity is lower, and focus on Horn NATs: it is proven in Sec-

¹We remind that $Circ(T_1) \vee Circ(T_2) \not\equiv Circ(T_1 \vee T_2)$ in general (take, e.g., $T_1 = a \wedge b$, $T_2 = b$).

tion 4 that here nesting can be efficiently eliminated if fixed variables are not allowed, and both model checking and inference are polynomial. Section 5 compares \mathcal{L}_{Circ} and NATs to other generalizations of circumscription such as *prioritized circumscription* [Lifschitz, 1985; 1994] and *theory curbing* [Eiter *et al.*, 1993; Eiter and Gottlob, 2000].

Our results prove that the expressive power that makes \mathcal{L}_{Circ} and NATs useful tools for the modularization of knowledge has indeed a cost, because the complexity of reasoning in such languages is higher than reasoning in a “flat” circumscriptive knowledge base. Anyway the PSPACE upper bound of the complexity of reasoning, and the similarity of their semantics with that of *quantified Boolean formulas* (QBFs), makes fast implementations possible by translating them into a QBF and then using one of the several available solvers, e.g., [Rintanen, 1999]. This approach could be used also for implementing meaningful fragments of NATs, such as the one in [Baral *et al.*, 1998], although this might be inefficient, like using a first-order theorem prover for propositional logic.

Given that QBFs can be polynomially encoded into NATs, we can show (in the full paper) that nested circumscription is more *succinct* than plain (unnested) circumscription, i.e., by nesting *Circ* operators (or NATs), we can express some circumscriptive theories in polynomial space, while they could be written in exponential space only, if nesting were not allowed. In this sense, we add new results to the *comparative linguistics of knowledge representation* [Gogic *et al.*, 1995].

2 Preliminaries

We assume a finite set At of propositional atoms, and let $\mathcal{L}(At)$ (for short, \mathcal{L} , if At does not matter or is clear from the context) be a standard propositional language over At . For any formula $\varphi \in \mathcal{L}$, we denote by $mod(\varphi)$ the set of its models. Capitals P, Q, Z etc stand for ordered sets of atoms, which we also view as lists. If $X = \{x_1, \dots, x_n\}$ and $X' = \{x'_1, \dots, x'_n\}$, then $X \leq X'$ denotes the formula $\bigwedge_{i=1}^n (x_i \rightarrow x'_i)$.

The extension of a model M on atoms A is denoted by $M[A]$. We denote by $\leq_{P;Z}$ the preference relation on models which minimizes P in parallel while Z is varying and all other atoms are fixed; i.e., $M \leq_{P;Z} M'$ (M is more or equally preferable to M') iff $M[P] \subseteq M'[P]$ and $M[Q] = M'[Q]$, where $Q = At \setminus P \cup Z$ and \subseteq and $=$ are taken componentwise. As usual, $M <_{P;Z} M'$ means $M \leq_{P;Z} M' \wedge M \neq M'$.

We denote by $Circ(\varphi; P; Z)$ the second-order circumscription [Lifschitz, 1985] of the formula φ where the atoms in P are minimized, the atoms in Z float, and all other atoms are fixed, which is defined as the following formula:

$$Circ(\varphi; P; Z) = \varphi[P; Z] \wedge \forall P' Z' ((\varphi[P'; Z'] \wedge P' \leq P) \rightarrow P \leq P'). \quad (1)$$

Here P' and Z' are lists of fresh atoms (not occurring in φ) corresponding to P and Z , respectively. The second-order formula (1) is a quantified Boolean formula (QBF) with free variables, whose semantics is defined in the standard way. Its models, i.e., assignments to the free variables such that the resulting sentence is valid, are the models M of φ which are

$P; Z$ -minimal, where a model M of φ is $P; Z$ -minimal, if no model M' of φ exists such that $M' <_{P;Z} M$.

3 Language \mathcal{L}_{Circ}

The language \mathcal{L}_{Circ} extends the standard propositional language \mathcal{L} (over a set of atoms At) by circumscriptive atoms. Formulas of \mathcal{L}_{Circ} are inductively built as follows:

1. $a \in \mathcal{L}_{Circ}$, for every $a \in At$;
2. if φ, ψ are in \mathcal{L}_{Circ} , then $\varphi \wedge \psi$ and $\neg\varphi$ are in \mathcal{L}_{Circ} ;
3. if $\varphi \in \mathcal{L}_{Circ}$ and P, Z are disjoint lists of atoms, then $Circ(\varphi; P; Z)$ is in \mathcal{L}_{Circ} (called *circumscriptive atom*).

Further Boolean connectives (\vee, \rightarrow , etc) are defined as usual. The semantics of any formula φ from \mathcal{L}_{Circ} is given in terms of models of a naturally associated QBF $\tau(\varphi)$, which is inductively defined as follows:

1. $\tau(a) = a$, if φ is an atom $a \in At$;
2. $\tau(\varphi \wedge \psi) = \tau(\varphi) \wedge \tau(\psi)$ and $\tau(\neg\varphi) = \neg\tau(\varphi)$;
3. $\tau(Circ(\varphi; P; Z)) = \tau(\varphi[P; Z]) \wedge \forall P' Z' ((\tau(\varphi[P'; Z']) \wedge P' \leq P) \rightarrow P \leq P')$.

Note that in 3, the second-order definition of circumscription is used to map the circumscriptive atom to a QBF which generalizes the circumscription formula in (1). In particular, if φ is an “ordinary” propositional formula ($\varphi \in \mathcal{L}$), then $\tau(Circ(\varphi; P; Z))$ coincides with the formula in (1).

Example 1 Consider the formula

$$\varphi = Circ(Circ(a \vee b; a; b) \vee Circ(b \vee c; b; c); a; c).$$

Applying rule 3 to inner circumscriptions we get $\tau(\varphi) = Circ((b \wedge \neg a) \vee (c \wedge \neg b); a; c)$. Applying rule 3 again we get $\tau(\varphi) = (\neg a \wedge \neg b \wedge c) \vee (\neg a \wedge b) = \neg a \wedge (b \vee c)$.

As usual, we write $M \models \varphi$ if M is a model of φ , and $\varphi \models \psi$ if ψ is a logical consequence of φ .

Eliminating fixed letters

As shown in [de Kleer and Konolige, 1989], the fixed letters can be removed from an ordinary circumscription. The same technique can be applied for formulas from \mathcal{L}_{Circ} as well. More precisely, let $\varphi = Circ(\psi; P; Z)$ be a circumscriptive atom. For each letter $q \notin P \cup Z$, introduce a fresh letter \bar{q} , add q, \bar{q} to P , and add a conjunct $q \leftrightarrow \neg\bar{q}$ to ψ . Let $\varphi' = Circ(\psi'; P'; Z)$ be the resulting circumscriptive atom. Then, the following holds.

Proposition 1 φ and φ' are equivalent modulo the set of all auxiliary letters \bar{q} .

Using this equivalence, we can eliminate all fixed letters from a formula $\alpha \in \mathcal{L}_{Circ}$, by replacing each circumscriptive atom φ in α with φ' , where the fresh atoms \bar{q} are made floating inside φ' and outside φ . Note that the resulting formula α' has size polynomial in the size of α .

3.1 Complexity results

Let the *Circ-nesting depth* (for short, nesting depth) of $\varphi \in \mathcal{L}_{Circ}$, denoted $nd(\varphi)$, be the maximum number of circumscriptive atoms along any path in the formula tree of φ .

Theorem 1 *Model checking for \mathcal{L}_{Circ} , i.e., deciding whether a given interpretation M is a model of a given formula $\varphi \in \mathcal{L}_{Circ}$, is PSPACE-complete. If $nd(\varphi) \leq k$ for a constant $k > 0$, then the problem is complete for $\Delta_{k+1}^P[O(\log n)]$, i.e., polynomial time with $O(\log n)$ many calls to an oracle for Σ_k^P , where n is the input size.*

Proof: (Sketch) Clearly, any formula $\varphi \in \mathcal{L}_{Circ}$ can be encoded, by definition, to a QBF $\tau(\varphi)$ (not necessarily in prenex form) in polynomial time. Thus deciding, $M \models \varphi$ is in PSPACE. By an inductive argument, we can see that for any circumscriptive atom $\varphi = Circ(\psi; P; Z)$ such that $nd(\varphi) \leq k$, deciding $M \models \varphi$ is in Π_k^P . Thus for any Boolean combination φ of atoms and circumscriptive atoms $\varphi_1, \dots, \varphi_m$ such that $nd(\varphi) (= \max_{i=1}^m nd(\varphi_i)) \leq k$, deciding $M \models \varphi$ is possible in Δ_{k+1}^P , i.e., polynomial time with one round of parallel Σ_k^P oracle calls. Since $\Delta_{k+1}^P = \Delta_{k+1}^P[O(\log n)]$, this proves the membership part for $nd(\varphi) \leq k$.

PSPACE-hardness of $M \models \varphi$ for general φ follows directly from the PSPACE-hardness of model checking for NAT theories shown below in Theorem 4, and from the fact that NATs can be polynomial-time embedded into \mathcal{L}_{Circ} (Corollary 2). Similarly, from the respective results on NATs we obtain that for circumscriptive atoms φ with $nd(\varphi) \leq k$ the problem is Π_k^P -hard (and thus Π_k^P -complete).

The $\Delta_{k+1}^P[O(\log n)]$ -hardness part for the case where φ is a Boolean combination of formulas $\varphi_1, \dots, \varphi_m \in \mathcal{L}_{Circ}$ such that $\max_{i=1}^m nd(\varphi_i) \leq k$ is then shown by a reduction from deciding, given m instances $(M_1, \varphi_1), \dots, (M_m, \varphi_m)$ of the model checking problem for circumscriptive atoms, whether the number of yes-instances among them is even. The $\Delta_{k+1}^P[O(\log n)]$ -completeness of this problem is an instance of Wagner's [1990] general result for all Π_k^P -complete problems. Moreover, we may use that all instances are on disjoint alphabets and the assertion [Wagner, 1990] that (M_i, φ_i) is a yes-instance only if (M_{i+1}, φ_{i+1}) is a yes-instance, for all $i \in \{1, \dots, m-1\}$. Then, we can define

$$\varphi = e \leftrightarrow \bigvee_{0 \leq 2i \leq m} \left(\bigwedge_{j=0}^{2i} \varphi_j \wedge \bigwedge_{j=2i+1}^n \neg \varphi_j \right)$$

where e is a fresh letter. The interpretation $M = \bigcup_{i=1}^m M_i \cup \{e\}$ is a model of φ if and only if the number of yes-instances among $(M_1, \varphi_1), \dots, (M_m, \varphi_m)$ is even. Clearly, φ and M can be constructed in polynomial time. \square

Theorem 2 *Deciding, given formulas $\varphi, \psi \in \mathcal{L}_{Circ}$ whether $\varphi \models \psi$ is PSPACE-complete. Hardness holds even if $\psi \in \mathcal{L}$. If the nesting depth of φ and ψ is bounded by the constant $k \geq 0$, then the problem is Π_{k+1}^P -complete.*

Proof: The problem is in PSPACE (resp., Π_{k+1}^P): An interpretation M such that $M \models \varphi \wedge \neg \psi$ can be guessed and verified in polynomial space (resp., $\Delta_{k+1}^P[O(\log n)]$), thus in polynomial time with an oracle for Π_k^P . Hence the problem is in NPSpace = PSPACE (resp., Π_{k+1}^P). Hardness follows from the polynomial time embedding of NATs into \mathcal{L}_{Circ} (Corollary 2) and Theorem 3. \square

Corollary 1 *Deciding satisfiability of a given formula $\varphi \in \mathcal{L}_{Circ}$ is PSPACE-complete. If the nesting depth is bounded by a constant $k \geq 0$, then the problem is Σ_{k+1}^P -complete.*

4 Nested Abnormality Theories (NATs)

Lifschitz [1995] proposed the hierarchical use of the circumscription principle, applied to building blocks of a more complex theory. This method was used, e.g., in [Baral *et al.*, 1998] for defining the formal semantics of narratives.

We assume that the atoms At include a set of distinguished atoms $Ab = \{ab_1, \dots, ab_k\}$ (which intuitively represent abnormality properties). *Blocks* are defined as the smallest set such that if c_1, \dots, c_m are distinct atoms not in Ab , and each of B_1, \dots, B_n is either a formula in \mathcal{L} or a block, then

$$B = \{c_1, \dots, c_m : B_1, \dots, B_n\},$$

is a block [Lifschitz, 1995], where c_1, \dots, c_m are called *described* by this block. The *nesting depth* of B , denoted $nd(B)$, is 0 if every B_i is from \mathcal{L} , and $1 + \max\{nd(B_i) \mid 1 \leq i \leq n\}$ otherwise.

A *nested abnormality theory* (NAT) [Lifschitz, 1995] is a collection $\mathcal{T} = B_1, \dots, B_n$ of blocks; its nesting depth, $nd(\mathcal{T})$, is defined as $\max\{nd(B_i) \mid 1 \leq i \leq n\}$.

The semantics of a NAT \mathcal{T} is defined by a mapping $\sigma(\mathcal{T})$ to a QBF as follows: $\sigma(\mathcal{T}) = \bigwedge_{B \in \mathcal{T}} \sigma(B)$, where for any block $B = \{C : B_1, \dots, B_n\}$,

$$\sigma(B) = \exists Ab. Circ(\bigwedge_{i=1}^n \sigma(B_i); Ab; C)$$

given that $\sigma(B_i) = B_i$ if B_i is a formula in \mathcal{L} .

Thus, a standard circumscription $Circ(\varphi; P; Z)$, where $\varphi \in \mathcal{L}$, is equivalent to a NAT $\mathcal{T} = \{Z : \varphi\}$ where P is viewed as the set of abnormality letters Ab ; notice that $nd(\mathcal{T}) = 0$.

We note the following useful proposition.

Proposition 2 *Let $\mathcal{T} = B_1, \dots, B_n$ be any NAT. Let $\mathcal{T}' = \{Z : B_1, \dots, B_n\}$ where Z is any subset of the atoms (disjoint with Ab). Then, \mathcal{T} and \mathcal{T}' have the same models.*

Indeed, \mathcal{T}' has void minimization of Ab (making each ab_j in Ab false), and fixed and floating letters can have any values.

Embedding NATs into \mathcal{L}_{Circ}

In the translation $\sigma(\mathcal{T})$, the minimized letters Ab are under an existential quantifier, and thus “projected” from the models of the formula $Circ(\dots)$. We can, modulo auxiliary letters, eliminate existential quantifiers as follows:

1. Rename every quantifier $\exists Ab$ in $\sigma(\mathcal{T})$ such that every quantified variable is different from any other variable.
2. In every circumscriptive subformula $Circ(\varphi; P; Z)$ of the renamed formula, add to the floating atoms all variables which are quantified in φ (including subformulas).
3. Drop all quantifiers.

Let $\sigma^*(\mathcal{T})$ be the resulting formula. Note that its size is polynomial (quadratic) in the size of $\sigma(\mathcal{T})$.

Example 2 *Let $AB = \{ab_1, ab_2\}$ and $\mathcal{T} = \{z : \mathcal{T}_1, ab_1 \leftrightarrow z\}$, where $\mathcal{T}_1 = \{z : ab_1 \leftrightarrow \neg ab_2, ab_1 \leftrightarrow z\}$. Then,*

$$\begin{aligned} \sigma(\mathcal{T}) &= \exists ab_1, ab_2. Circ(\sigma(\mathcal{T}_1) \wedge (ab_1 \leftrightarrow z); ab_1, ab_2; z), \\ \text{where} \\ \sigma(\mathcal{T}_1) &= \exists ab_1, ab_2. Circ((ab_1 \leftrightarrow \neg ab_2) \wedge (ab_1 \leftrightarrow z); ab_1, ab_2; z). \end{aligned}$$

In Step 1, we rename ab_1 and ab_2 in $\sigma(\mathcal{T}_1)$ to ab_3 and ab_4 , respectively, and add in Step 2 ab_3, ab_4 to the floating letter z of \mathcal{T} . After dropping quantifiers in Step 3, we obtain:

$$\begin{aligned}\sigma^*(\mathcal{T}) &= \text{Circ}(\sigma^*(\mathcal{T}_1) \wedge (ab_1 \leftrightarrow z); ab_1, ab_2; z, ab_3, ab_4), \\ \sigma^*(\mathcal{T}_1) &= \text{Circ}((ab_3 \leftrightarrow \neg ab_4) \wedge (ab_3 \leftrightarrow z); ab_3, ab_4; z).\end{aligned}$$

Let $A^*(\mathcal{T})$ be the set of atoms from Step 3 of the embedding $\sigma^*(\mathcal{T})$. The following result, which can be proven by induction on the nesting depth of \mathcal{T} , states the correctness of σ^* .

Proposition 3 *For any NAT \mathcal{T} , the formulas $\sigma(\mathcal{T})$ and $\sigma^*(\mathcal{T})$ are equivalent modulo $A^*(\mathcal{T})$.*

Corollary 2 *Modulo auxiliary letters, NAT is (semantically) a fragment of $\mathcal{L}_{\text{Circ}}$, and polynomial-time embedded via σ^* .*

Notice that is not possible to add in Step 2 of the embedding $\sigma^*(\mathcal{T})$ the quantified variables in φ to the fixed atoms.

Example 3 *Reconsider the NAT \mathcal{T} in Ex. 2. Note that \emptyset is the unique model of $\sigma(\mathcal{T})$. The formula $\sigma^*(\mathcal{T}_1)$ has, if we disregard ab_1, ab_2 (which are fixed in it), the models $M_1 = \{ab_3, z\}$ and $M_2 = \{ab_4\}$. They give rise to the two models $N_1 = \{ab_1, z, ab_3\}$ and $N_2 = \{ab_4\}$ of $\sigma^*(\mathcal{T}_1) \wedge (ab_1 \leftrightarrow z)$, of which N_2 is $ab_1, ab_2; ab_3, ab_4, z$ -minimal.*

However, if ab_3, ab_4 were fixed in $\sigma^(\mathcal{T})$, then both N_1 and N_2 would be models of $\sigma^*(\mathcal{T})$, as they are $ab_1, ab_2; z$ -minimal. Therefore, Proposition 3 would fail.*

Eliminating fixed letters

Every fixed letter q can be removed from a NAT \mathcal{T} similarly as from a formula $\varphi \in \mathcal{L}_{\text{Circ}}$. However, we must take into account that a fixed letter q may not be simply declared as a minimized letter in the rewriting, since there is a special set Ab of minimized letter which has restricted uses. We surpass this by introducing two special abnormality letters ab_q and $\overline{ab_q}$ in Ab , and by adding the formula $(q \leftrightarrow ab_q) \wedge (ab_q \leftrightarrow \neg \overline{ab_q})$ as a new block to \mathcal{T} , as well as $ab_q \leftrightarrow \neg \overline{ab_q}$ in every other block of \mathcal{T} . The letter q is declared floating in \mathcal{T} .

4.1 Complexity of NATs

Ordinary circumscription can express a QBF sentence $\Phi = \forall X \exists Y \psi$ (where $\psi \in \mathcal{L}$) as follows. Let u be a fresh variable.

Lemma 1 (cf. [Eiter and Gottlob, 1993]) Φ is true if and only if $\text{Circ}(\varphi; u; Y) \models \neg u$, where $\varphi = \psi \vee u$.

This circumscription can be easily stated as a NAT. Set

$$\mathcal{T}_1 = \{Y, u : \varphi, u \leftrightarrow ab\}.$$

Then Lemma 1 implies that $\mathcal{T}_1 \models \neg u$ iff Φ is true. In fact, denote for any interpretation M and set of atoms S by $t_M(S)$ the assignment to S as given by M . Then, every model M of \mathcal{T}_1 must be, if we fix the atoms in X to $t_M(X)$, a model of φ such that $M \models u$ if and only if $\psi[X/t_M(X)]$ is unsatisfiable.

Starting from this result, we prove PSPACE-hardness of inference $\mathcal{T} \models \varphi$ from a NAT theory \mathcal{T} . The basic technique is to introduce further variables as *parameters* V into the formula Φ from Lemma 1, which are kept fixed at the inner levels. At a new outermost level to be added, the letter u is used for evaluating the formula at a certain level. We must in alternation minimize and maximize the value of u .

Consider the case of a QBF $\Phi = \forall X \exists Y \psi[V]$, where V are free variables in it, considered as “parameters”. We nest \mathcal{T}_1 into the following theory \mathcal{T}_2 :

$$\mathcal{T}_2 = \{X \cup Y, u : \mathcal{T}_1, u \leftrightarrow \neg ab\}$$

This amounts to the following circumscription:

$$\sigma(\mathcal{T}_2) = \exists ab \text{Circ}(\exists ab \text{Circ}(\varphi \wedge (u \leftrightarrow ab); ab; Y, u) \wedge (u \leftrightarrow \neg ab); ab; X \cup Y, u).$$

The outer circumscription minimizes ab and thus maximizes u . $\sigma(\mathcal{T}_2)$ is, by Proposition 3, equivalent to the formula

$$\begin{aligned}\sigma^*(\mathcal{T}_2) &= \text{Circ}(\sigma^*(\mathcal{T}_1) \wedge (u \leftrightarrow \neg a_2); a_2; X \cup Y, u, a_1), \\ \sigma^*(\mathcal{T}_1) &= \text{Circ}(\varphi \wedge (u \leftrightarrow a_1); a_1; Y, u)\end{aligned}$$

modulo the atoms a_1 and a_2 . The following can be shown:

Lemma 2 $\mathcal{T}_2 \models u$ if and only if for every assignment $s(V)$, the QBF $\exists X \forall Y \neg \psi[V/s(V)]$ is true (i.e., $\Phi[V/s(V)]$ is false).

It follows from the lemma that deciding, given a NAT \mathcal{T}_2 of nesting depth 1 and $\psi \in \mathcal{L}$, whether $\mathcal{T}_2 \models \psi$ is Π_3^P -hard.

We generalize this pattern to encode the evaluation of a QBF

$$\Phi = Q_n X_n Q_{n-1} X_{n-1} \cdots \forall X_2 \exists X_1 \psi, \quad n \geq 1, \quad (2)$$

where the quantifiers Q_i alternate, into inference $\mathcal{T} \models \psi$ from a NAT theory \mathcal{T} as follows.

Let $\varphi = \psi \vee u$, where u is a fresh atom. Define inductively

$$\begin{aligned}\mathcal{T}_1 &= \{X_1, u : \varphi, u \leftrightarrow ab\}, \\ \mathcal{T}_{2k} &= \{X_1, \dots, X_{2k}, u : \mathcal{T}_{2k-1}, u \leftrightarrow \neg ab\}, \\ \mathcal{T}_{2k+1} &= \{X_1, \dots, X_{2k+1}, u : \mathcal{T}_{2k}, u \leftrightarrow ab\},\end{aligned}$$

for all $2k, 2k+1 \in \{2, \dots, n\}$, and let $\mathcal{T}_0 = \{ : \varphi \}$. Note that \mathcal{T}_0 is equivalent to φ , and that $nd(\mathcal{T}_i) = i-1$, for all $i \in \{1, \dots, n\}$ while $nd(\mathcal{T}_0) = 0$. We obtain the following.

Lemma 3 *For every $n \geq 1$, \mathcal{T}_{n-1} has some model, and*

- if n is odd, then $\mathcal{T}_{n-1} \models u$ iff Φ is false, i.e., $\neg \Phi$ is true;
- if n is even, then $\mathcal{T}_{n-1} \models \neg u$ iff Φ is true.

This statement can be proved by induction on $n \geq 1$.

Lemma 4 *Model checking for NATs, i.e., deciding whether a given interpretation M is a model of a given NAT \mathcal{T} , is in PSPACE. If $nd(\mathcal{T}) \leq k$ for constant $k \geq 0$, then it is in Π_{k+1}^P .*

Proof: For $k=0$, the problem amounts to model checking for ordinary circumscription, which is in coNP. For $k>0$, we can test recursively for each block B of \mathcal{T} whether M is a model of each block B_1, \dots, B_n in $B = \{C : B_1, \dots, B_n\}$, and that no model N of B_1, \dots, B_n exists such that $N <_{Ab;C} M$. The recursion depth is linear, and the procedure runs in quadratic space. Moreover, if $nd(\mathcal{T})$ is bounded by a constant k , then the check for M and the existence of N witnessing that M is not model of B , can be done in nondeterministic polynomial time with an oracle for Π_k^P . Thus, the problem is in PSPACE (resp., in Π_{k+1}^P). \square

The construction in Lemma 2 shows how it is possible to polynomially embed QBF evaluation into inference wrt a NAT. In turn, Proposition 3 shows that a NAT can be polynomially embedded into an $\mathcal{L}_{\text{Circ}}$ formula. The following theorem highlights the consequences of such relations on complexity of inference wrt a NAT.

Theorem 3 *Deciding, given a NAT theory \mathcal{T} and a propositional formula φ , whether $\mathcal{T} \models \varphi$ is PSPACE-complete. If $nd(\mathcal{T}) \leq k$ for constant $k \geq 0$, then it is Π_{k+2}^P -complete.*

Proof: The hardness part follows from Lemma 3 above. As for the membership part, a model M of \mathcal{T} such that $M \not\models \varphi$ can be guessed and verified in PSPACE (resp., with the help of a Π_{k+1}^P -oracle). Thus the problem is in co-NPSPACE = PSPACE (resp., Π_{k+2}^P). \square

The next theorem shows that the upper bounds on model checking for NATs have matching lower bounds. (This is expected from Theorem 3: if model checking were in the Polynomial Hierarchy (PH), then also inference would be in PH.)

Theorem 4 *Given a NAT theory \mathcal{T} and an interpretation M , deciding whether $M \models \mathcal{T}$ is PSPACE-complete. If $nd(\mathcal{T}) \leq k$ for a constant $k \geq 0$, then the problem is Π_{k+1}^P -complete.*

Proof: (Sketch) By Lemma 4, it remains to show the hardness parts. Note that for $k = 0$, this is immediate from results in [Eiter and Gottlob, 1993]. We can slightly adapt the above encoding of QBFs into NATs. Given a QBF Φ as in Eq. (2), let the NATs $\mathcal{T}_0, \dots, \mathcal{T}_n$ be as there, with the only difference that $\varphi = \psi \vee u$ is replaced by

$$\varphi' = (\psi \vee u \vee (X_n \wedge v)) \wedge ((X_n \wedge v) \rightarrow X_1 \cup \dots \cup X_{n-1} \cup \epsilon_n),$$

where $\epsilon_n = \{u\}$ if n is odd and $\epsilon_n = \{\neg u\}$ otherwise. Here v is a new letter, defined (i.e., floating) in \mathcal{T}_n and fixed elsewhere; X_n resp. $X_1 \cup \dots \cup X_{n-1} \cup \epsilon_n$ is viewed as conjunction of its formulas. Let $\mathcal{T}'_0, \dots, \mathcal{T}'_n$ be the resulting NATs.

Define $M = \bigcup_{i=1}^n X_i \cup \{v, u\}$ if n is odd and $M = \bigcup_{i=1}^n X_i \cup \{v\}$ if n is even. Note that $M \models \varphi'$ and M is, modulo u , the only candidate for a model of \mathcal{T}'_n in which X_n and v are true: any such model must satisfy φ' , and if $X_n \wedge v$ are true, then all other atoms have a unique truth value.

It holds that M is a model of \mathcal{T}'_n iff the QBF Φ is false (resp., true). It follows that model checking is PSPACE-hard and Π_{k+1}^P -hard if $nd(\mathcal{T}) \leq k$ for constant k ; note that $nd(\mathcal{T}'_n) = n - 1$. \square

4.2 A polynomial-time fragment

We call a block $\{C : B_1, \dots, B_n\}$ *Horn*, if each B_i is a Horn CNF if $B_i \in \mathcal{L}$, and recursively B_i is Horn otherwise. A NAT \mathcal{T} is *Horn*, if each of its blocks is Horn.

In [Cadoli and Lenzerini, 1994], it was shown that deciding $Circ(\varphi; P; \emptyset) \models \neg u$, where φ is a propositional Horn CNF and u is an atom, is coNP-complete. That is, already for Horn NATs \mathcal{T} without nesting (i.e., $nd(\mathcal{T}) = 0$) inference is intractable, and nesting may further add on complexity.

Horn NATs without fixed letters

The technique of removing fixed letters from a Horn NAT does not work, since it uses non-Horn clauses. Thus, Horn NATs without fixed letters do not immediately inherit complexity, in particular intractability, from general Horn NATs.

In fact, for this class we obtain positive results. Note that here still minimization of atoms q is possible via auxiliary atoms $ab_q \in Ab$ and Horn axioms $q \rightarrow ab_q, ab_q \rightarrow q$ in \mathcal{T} .

Call any $P; Z$ -minimal model of a NAT \mathcal{T} such that $P = At \setminus Ab$ and $Z = \emptyset$ a *minimal model* of \mathcal{T} .

Theorem 5 *Let \mathcal{T} be a Horn NAT without fixed letters. Then, (1) \mathcal{T} has a least (i.e., a unique minimal) model, and (2) \mathcal{T} is equivalent to a Horn CNF $\varphi(\mathcal{T})$. Furthermore, both $\varphi(\mathcal{T})$ and $M(\mathcal{T})$ are computable in polynomial time.*

Proof: (Sketch) Let, for any Horn CNF ψ and model $M \subseteq At$, be ψ^M the Horn CNF resulting from ψ after substituting each $ab_j \in Ab$ by \top (true) if $M \models ab_j$ and by \perp (false) if $M \models \neg ab_j$ and subsequent standard simplifications.

Let \mathcal{T} be a single block $B = \{Z : B_1, \dots, B_n\}$, where $Z = At \setminus Ab$. Define the Horn CNF $\varphi(B)$ recursively by

$$\varphi(B) := \bigwedge_{B_i \in \mathcal{L}} B_i^{M_0} \wedge \bigwedge_{B_i \notin \mathcal{L}} \varphi(B_i),$$

where $M_0 = M_0(B)$ is the least model of the Horn CNF

$$\psi(B) := \bigwedge_{B_i \in \mathcal{L}} B_i \wedge \bigwedge_{B_i \notin \mathcal{L}} \varphi(B_i),$$

and

$$M(B) := M_0[At \setminus Ab] (= M_0[Z]).$$

Then, by induction on $nd(B) \geq 0$, we can show that (1) $M(B)$ is the least model of B , and (2) $\varphi(B)$ is equivalent to B .

Let, for any formula α , block B , NAT \mathcal{T} etc denote $\|\alpha\|$, $\|B\|$, $\|\mathcal{T}\|$ etc the representation size of the respective object.

Obviously, we can compute $\varphi(B)$ inductively. For the Horn CNFs $\psi(B)$ and $\varphi(B)$, we have $\|\psi(B)\| \leq \|B\|$ and $\|\varphi(B)\| \leq \|B\|$. Of the model M_0 , we only need its restriction $M_0[A]$ to the atoms A occurring in B (all other atoms are irrelevant for computing $\varphi(B)$). We can compute $M_0[A]$ from $\psi(B)$ in $O(\|B\|)$ time (recall that the least model of a Horn CNF α is computable in $O(\|\alpha\|)$ time), and $\bigwedge_{B_i \in \mathcal{L}} B_i^{M_0}$ from $M_0[A]$ in $O(\|B\|)$ time. Overall, it follows that for $\mathcal{T} = B$, we can compute both $\varphi(\mathcal{T})$ and its least model $M(\mathcal{T})$ in time $O(\#\mathcal{T} \|\mathcal{T}\|)$, where $\#\mathcal{T}$ is the number of (recursive) blocks in \mathcal{T} , thus in polynomial time.

By Prop. 2, we can replace a multiple block $\mathcal{T} = B_1, \dots, B_n$ by $\mathcal{T}' = \{At \setminus Ab : B_1, \dots, B_n\}$, which is Horn and without fixed letters, and obtain analogous results. \square

Using sophisticated data structures, the (relevant parts of) models $M_0(B)$ above can be computed incrementally, where each clause in $\varphi(B)$ is fired at most once. Overall, $\varphi(\mathcal{T})$ and $M(\mathcal{T})$ are computable in $O(\|\mathcal{T}\|)$ time. We thus have:

Theorem 6 (Flat Normal Form) *Every Horn NAT \mathcal{T} without fixed letters can be rewritten to an equivalent Horn NAT $\{Z : \varphi\}$ without fixed letters, where $\varphi \in \mathcal{L}$ is a Horn CNF, in $O(\|\mathcal{T}\|)$ time (i.e., in linear time).*

Thus, nesting in Horn NATs without fixed letters does not increase the expressiveness, and can be efficiently eliminated.

We note some easy corollaries of the previous result.

Corollary 3 *Deciding satisfiability of a given Horn NAT \mathcal{T} without fixed letters is polynomial.*

Corollary 4 *Model checking for a given Horn NAT \mathcal{T} without fixed letters and model M is polynomial.*

For the inference problem, we obtain the following result.

Theorem 7 *Given a Horn NAT \mathcal{T} without fixed letters and $\varphi \in \mathcal{L}$, deciding $\mathcal{T} \models \varphi$ is coNP-complete. If φ is a CNF, then the problem is polynomial.*

Proof: Since model checking is polynomial, the problem is clearly in coNP. The coNP-hardness part follows from coNP-completeness of validity checking for a given $\varphi \in \mathcal{L}$ (ask whether $\{Z : \top\} \models \varphi$, where Z contains all letters).

We can reduce $\mathcal{T} \models \psi$ to $\varphi(\mathcal{T}) \models \psi$ in $O(\|\mathcal{T}\|)$ time, where $\varphi(\mathcal{T})$ is a Horn CNF. If $\psi = \bigwedge_{i=1}^m \alpha_i$ is a CNF of clauses α_i , the latter can be checked in $O(m\|\varphi(\mathcal{T})\| + \|\psi\|)$ time, thus in $O(m\|\mathcal{T}\| + \|\psi\|)$ time (check $\varphi(\mathcal{T}) \models \alpha_i$, which is $O(\|\varphi(\mathcal{T})\| + \|\alpha_i\|)$ time, for all $i \in \{1, \dots, m\}$). \square

5 Other Generalizations of Circumscription

As we have discussed above, fixed letters can be removed from \mathcal{L}_{Circ} and NAT theories, respectively. Using these techniques, the hardness results of Sections 3.1 and 4.1 can be sharpened to theories without fixed letters.

Prioritized circumscription generalizes circumscription $Circ(\varphi; P; Z)$ by partitioning the letters P into priority levels $P_1 > P_2 > \dots > P_n$, and informally pruning all models of φ which are not minimal on P_i , while $Z \cup P_{i+1} \cup \dots \cup P_n$ floats and $P_1 \cup \dots \cup P_{i-1}$ is fixed, for $i = 1, \dots, n$ (cf. [Lifschitz, 1994]). This can be readily expressed as the nested circumscription $Circ(\dots Circ(Circ(\varphi; P_1; Z \cup P_2 \cup \dots \cup P_n); P_2; Z \cup P_3 \cup \dots \cup P_n) \dots)$. This generalization differs from \mathcal{L}_{Circ} and NATs: The complexity does not increase, as inference and model checking remain Π_2^P -complete and coNP-complete, respectively. Intuitively, the reason is that prioritization allows only for a restricted change of the role of the same letter in iterations (floating to minimized and minimized to fixed) and forbids reconsidering minimized letters.

Curbing is yet another extension of circumscription [Eiter *et al.*, 1993; Eiter and Gottlob, 2000]. Rather than the (hierarchical) use of circumscription applied to blocks, curbing aims at softening minimization, and allows for inclusive interpretation of disjunction where ordinary circumscription returns exclusive disjunction. Semantically, $Curb(\varphi; P; Z)$ for a formula $\varphi \in \mathcal{L}$ is the smallest set $\mathcal{M} \subseteq mod(\varphi)$ which contains all models of $Circ(\varphi; P; Z)$ and is closed under minimal upper bounds in $mod(\varphi)$. A *minimal upper bound* (mub) of a set \mathcal{M}' of models in $mod(\varphi)$ is a model $M \in mod(\varphi)$ such that (1) $M' \leq_{P,Z} M$, for every $M' \in \mathcal{M}'$, and (2) there exists no $N \in mod(\varphi)$ satisfying item 1 such that $N <_{P,Z} M'$.

For example, $\varphi = a \vee b$ has two minimal models, if all letters are minimized ($P = At = \{a, b\}$): $\{a\}$ and $\{b\}$. The model $\{a, b\}$ is not a minimal model of φ , but is a curb model of φ . Thus, $Curb(a \vee b) \equiv a \vee b$. On the other hand, if $At = \{a, b, c\}$, then $Curb(a \vee b) \equiv (a \vee b) \wedge \neg c$ is different from $a \vee b$; the unsupported letter c is set to false. Note that $Circ(a \vee b) \equiv (a \leftrightarrow \neg b) \wedge c$ is different from $Curb(a \vee b)$.

Like for \mathcal{L}_{Circ} and NATs, inference and model checking for $Curb(\varphi; P; Z)$ are PSPACE-complete [Eiter and Gottlob, 2000]. However, in the first-order case, the formalisms have different expressiveness. Let us consider theories in a function-free language under *domain closure* (DC) and the *unique names assumption* (UNA), i.e., the (finite) “datalog” case. In this case, curbing can express some problems which \mathcal{L}_{Circ} and NATs can not. We can, e.g., write a (fixed) interpreter T_I in this language for curbing *varying* propositional 3CNF formulas φ , input as ground facts $F(\varphi)$, such that the

curb models of $T_I \cup F(\varphi)$ and φ are in 1-1 correspondence. Notice that curbing such 3CNFs φ is PSPACE-complete, and thus, by well-known results in complexity, this is not expressible by a fixed \mathcal{L}_{Circ} or NAT theory (unless PH=PSPACE).

Further relationships between \mathcal{L}_{Circ} resp. NATs and curbing, as well as other expressive KR formalisms [Baral *et al.*, 2000; Pollett and Remmel, 1997], remain to be explored.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) Project N. Z29-INF.

References

- [Baral *et al.*, 1998] C. Baral, A. Gabaldon, and A. Provetti. Formalizing narratives using nested circumscription. *Artificial Intelligence*, 104(1-2):107–164, 1998.
- [Baral *et al.*, 2000] C. Baral, V. Kreinovich, and R. Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122(1/2):241–267, 2000.
- [Cadoli and Lenzerini, 1994] M. Cadoli and M. Lenzerini. The complexity of propositional closed world reasoning and circumscription. *JCSS*, 43:165–211, 1994.
- [de Kleer and Konolige, 1989] J. de Kleer and K. Konolige. Eliminating the fixed predicates from a circumscription. *Artificial Intelligence*, 39:391–398, 1989.
- [Eiter and Gottlob, 1993] T. Eiter and G. Gottlob. Propositional circumscription and extended closed world reasoning are Π_2^P -complete. *Theoretical Computer Science*, 114(2):231–245, 1993.
- [Eiter and Gottlob, 2000] T. Eiter and G. Gottlob. On the complexity of theory curbing. In *Proc. LPAR-2000*, LNCS 1955, 1–19. 2000.
- [Eiter *et al.*, 1993] T. Eiter, G. Gottlob, and Y. Gurevich. Curb your theory! A circumscriptive approach for inclusive interpretation of disjunctive information. In *Proc. IJCAI-93*, 634–639. Morgan Kaufman, 1993.
- [Gogic *et al.*, 1995] G. Gogic, H.A. Kautz, Ch.H. Papadimitriou, and B. Selman. The Comparative Linguistics of Knowledge Representation. In *Proc. IJCAI-95*, 862–869.
- [Lifschitz, 1985] V. Lifschitz. Computing circumscription. In *Proc. IJCAI-85*, 121–127, 1985.
- [Lifschitz, 1994] V. Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, eds, *Handbook of Logic in AI and Logic Programming*, III, 297–352. 1994.
- [Lifschitz, 1995] V. Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74(2):351–365, 1995.
- [McCarthy, 1980] J. McCarthy. Circumscription – A form of non-monotonic reasoning. *Artif. Intell.*, 13:27–39, 1980.
- [Pollett and Remmel, 1997] C. Pollett and J. Remmel. Non-monotonic reasoning with quantified boolean constraints. In *Proc. LPNMR'97*, 18–39, LNCS 1265. Springer, 1997.
- [Rintanen, 1999] J. Rintanen. Improvements to the evaluation of quantified boolean formulae. In *Proc. IJCAI '99*, 1192–1197. AAAI Press, 1999.
- [Wagner, 1990] K. Wagner. Bounded query classes. *SIAM Journal of Computing*, 19(5):833–846, 1990.

A Perspective on Knowledge Compilation

Adnan Darwiche

Computer Science Department
University of California
Los Angeles, CA 90095, USA
darwiche@cs.ucla.edu

Pierre Marquis

CRIL/Université d'Artois
F-62307, Lens Cedex, France
marquis@cril.univ-artois.fr

Abstract

We provide a perspective on knowledge compilation which calls for analyzing different compilation approaches according to two key dimensions: the succinctness of the target compilation language, and the class of queries and transformations that the language supports in polytime. We argue that such analysis is necessary for placing new compilation approaches within the context of existing ones. We also go beyond classical, flat target compilation languages based on CNF and DNF, and consider a richer, nested class based on directed acyclic graphs, which we show to include a relatively large number of target compilation languages.

1 Introduction

Knowledge compilation has emerged recently as a key direction of research for dealing with the computational intractability of general propositional reasoning [Darwiche, 1999a; Cadoli and Donini, 1997; Boufkhad *et al.*, 1997; Khardon and Roth, 1997; Selman and Kautz, 1996; Schrag, 1996; Marquis, 1995; del Val, 1994; Dechter and Rish, 1994; Reiter and de Kleer, 1987]. According to this direction, a propositional theory is compiled off-line into a target language, which is then used on-line to answer a large number of queries in polytime. The key motivation behind knowledge compilation is to push as much of the computational overhead into the off-line phase, which is amortized over all on-line queries. But knowledge compilation can serve other important purposes as well. For example, target compilation languages and their associated algorithms can be very simple, allowing one to develop on-line reasoning systems for simple software and hardware platforms. Moreover, the simplicity of algorithms that operate on compiled languages help in streamlining the effort of algorithmic design into a single task: that of generating the smallest compiled representations possible, as that turns out to be the main computational bottleneck in compilation approaches.

There are three key aspects of any knowledge compilation approach: the succinctness of the target language into which the propositional theory is compiled; the class of queries that can be answered in polytime based on the compiled representation; and the class of transformations that can be applied to the representation in polytime. The AI literature has thus

far focused mostly on target compilation languages which are variations on DNF and CNF formulas, such as Horn theories and prime implicates. Moreover, it has focused mostly on clausal entailment queries, with very little discussion of tractable transformations on compiled theories.

The goal of this paper is to provide a broad perspective on knowledge compilation by considering a relatively large number of target compilation languages and analyzing them according to their *succinctness* and the class of *queries/transformations* that they admit in polytime.

Instead of focusing on classical, *flat* target compilation languages based on CNF and DNF, we consider a richer, *nested* class based on representing propositional sentences using directed acyclic graphs, which we refer to as NNF. We identify a number of target compilation languages that have been presented in the AI, formal verification, and computer science literature and show that they are special cases of NNF. For each such class, we list the extra conditions that need to be imposed on NNF to obtain the specific class, and then identify the set of queries and transformations that the class supports in polytime. We also provide cross-rankings of the different subsets of NNF, according to their succinctness and the polytime operations they support.

The main contribution of this paper is then a *map* for deciding the target compilation language that is most suitable for a particular application. Specifically, we propose that one starts by identifying the set of queries and transformations needed for their given application, and then choosing the most succinct language that supports these operations in polytime.

This paper is structured as follows. We start by formally defining the NNF language in Section 2, where we list a number of conditions on NNF that give rise to a variety of target compilation languages. We then study the succinctness of these languages in Section 3 and provide a cross-ranking that compares them according to this measure. We consider a number of queries and their applications in Section 4 and compare the different target compilation languages according to their tractability with respect to these queries. Section 5 is then dedicated to a class of transformations, their applications, and their tractability with respect to the different target compilation languages. We finally close in Section 6 by some concluding remarks. Proofs of theorems are omitted for space limitations but can be found in [Darwiche and Marquis, 2001].

2 The NNF Language

We consider more than a dozen languages in this paper, all of which are subsets of the NNF language, which is defined formally as follows [Darwiche, 1999a; 1999b].

Definition 2.1 Let PS be a finite set of prop. variables. A sentence in NNF_{PS} is a rooted, directed acyclic graph (DAG) where each leaf node is labeled with *true*, *false*, X or $\neg X$, $X \in PS$; and each internal node is labeled with \wedge or \vee and can have arbitrarily many children. The size of a sentence in NNF_{PS} is the number of its DAG edges. Its height is the maximum number of edges from the root to some leaf in the DAG.

Figure 1 depicts a sentence in NNF, which represents the odd parity function (we omit reference to variables PS when no confusion is anticipated). Any propositional sentence can be represented as a sentence in NNF, so the NNF language is complete.

It is important here to distinguish between a *representation* language and a *target compilation* language. The former should be natural enough to enable one to encode knowledge directly, while the latter should be tractable enough to permit some polytime queries and/or transformations. We will consider a number of target compilation languages that do not qualify as representation languages from this perspective, as they are not suitable for humans to construct or interpret. We will also consider a number of representation languages that do not qualify as target compilation languages.¹

For a language to qualify as a target compilation language, we require that it permits a polytime clausal entailment test. Therefore, NNF does not qualify as a target compilation language (unless $P=NP$) [Papadimitriou, 1994], but many of its subsets do. We define a number of these subsets below, each of which is obtained by imposing further conditions on NNF.

We will distinguish between two key subsets of NNF: *flat* and *nested* subsets. We first consider flat subsets, which result from imposing combinations of the following properties:

- **Flatness:** The height of each sentence is at most 2. The sentence in Figure 4 is flat, but the one in Figure 1 is not.
- **Simple-disjunction:** The children of each or-node are leaves that share no variables (the node is a *clause*).
- **Simple-conjunction:** The children of each and-node are leaves that share no variables (the node is a *term*). The sentence in Figure 4 satisfies this property.

Definition 2.2 The language \mathbb{F} -NNF is the subset of NNF satisfying flatness. The language CNF is the subset of \mathbb{F} -NNF satisfying simple-disjunction. The language DNF is the subset of \mathbb{F} -NNF satisfying simple-conjunction.

CNF does not permit a polytime clausal entailment test and, hence, does not qualify as a target compilation language (unless $P=NP$). But its DNF dual does.

¹Admittedly, the notion of a representation language is not a formal one, yet the distinction is important. For example, we believe that when proposing target compilation languages in AI, there is usually an implicit requirement that the proposed language is also a representation language. As we shall see later, however, the most powerful target compilation languages are not suitable for humans to specify or interpret directly.

The following subset of CNF, *prime implicates*, has been quite influential in computer science:

Definition 2.3 The language PI is the subset of CNF in which each clause entailed by the sentence is subsumed by a clause that appears in the sentence; and no clause in the sentence is subsumed by another.

A dual of PI , *prime implicants* IP , can also be defined.²

Definition 2.4 The language IP is the subset of DNF in which each term entailing the sentence subsumes some term that appears in the sentence; and no term in the sentence is subsumed by another term.

We now consider *nested* subsets of the NNF language, which do not impose any restriction on the height of a sentence. Instead, these subsets result from imposing one or more of the following conditions: *decomposability*, *determinism*, *smoothness*, *decision*, and *ordering*. We start by defining the first three properties. From here on, if C is a node in an NNF, then $\text{Vars}(C)$ denotes the set of all variables that label the descendants of node C .

- **Decomposability** [Darwiche, 1999a; 1999b]: An NNF satisfies this property if for each conjunction C in the NNF, the conjuncts of C do not share variables. That is, if C_1, \dots, C_n are the children of and-node C , then $\text{Vars}(C_i) \cap \text{Vars}(C_j) = \emptyset$ for $i \neq j$. Consider the and-node marked in Figure 1(a). This node has two children, the first contains variables A, B while the second contains variables C, D . This and-node is then decomposable since the two children do not share variables. Each other and-node in Figure 1(a) is also decomposable and, hence, the NNF in this figure is decomposable.
- **Determinism** [Darwiche, 2000]: An NNF satisfies this property if for each disjunction C in the NNF, each two disjuncts of C are logically contradictory. That is, if C_1, \dots, C_n are the children of or-node C , then $C_i \wedge C_j \models \text{false}$ for $i \neq j$. Consider the or-node marked in Figure 1(b), which has two children corresponding to sub-sentences $\neg A \wedge B$ and $\neg B \wedge A$. These two sub-sentences are logically contradictory. The or-node is then deterministic and so are the other or-nodes in Figure 1(b). Hence, the NNF in this figure is deterministic.
- **Smoothness** [Darwiche, 2000]: An NNF satisfies this property if for each disjunction C in the NNF, each disjunct of C mentions the same variables. That is, if C_1, \dots, C_n are the children of or-node C , then $\text{Vars}(C_i) = \text{Vars}(C_j)$ for $i \neq j$. Consider the marked or-node in Figure 1(c). This node has two children, each of which mentions variables A, B . This or-node is then smooth and so are the other or-nodes in Figure 1(c). Hence, the NNF in this figure is smooth.³

The properties of decomposability, determinism and smoothness lead to a number of interesting subsets of NNF.

²Horn theories (and renamable Horn theories) represent another target compilation subset of CNF, but we do not consider it here since we restrict our attention to complete languages only.

³Any sentence in NNF can be smoothed in polytime, while preserving decomposability and determinism. Preserving flatness, however, may blow-up the size of given NNF.

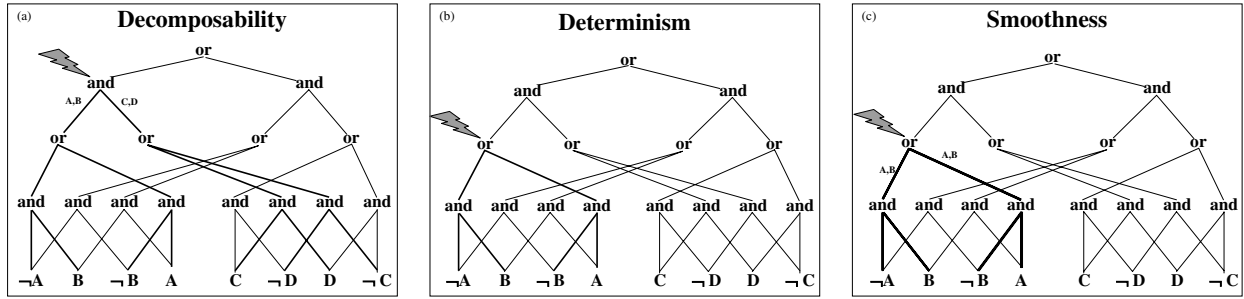


Figure 1: A sentence in NNF. Its size is 30 and height is 4.

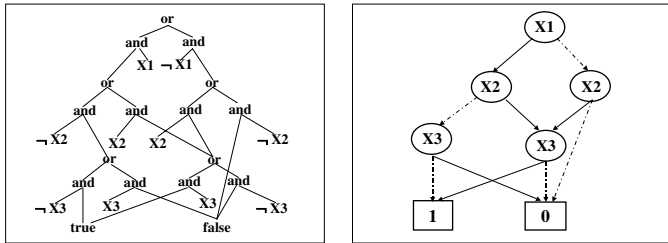


Figure 2: On the left, a sentence in the BDD language. On the right, its corresponding binary decision diagram.

Definition 2.5 The language DNNF is the subset of NNF satisfying decomposability; \bar{d} -NNF is the subset satisfying determinism; s -NNF is the subset satisfying smoothness; \bar{d} -DNNF is the subset satisfying decomposability and determinism; and $s\bar{d}$ -DNNF is the subset satisfying decomposability, determinism and smoothness.

Note that DNF is a strict subset of DNNF [Darwiche, 1999a; 1999b]. The following *decision* property comes from the literature on *binary decision diagrams* [Bryant, 1986].

Definition 2.6 (Decision) A decision node N in an NNF sentence is one which is labeled with *true*, *false*, or is an *or*-node having the form $(X \wedge \alpha) \vee (\neg X \wedge \beta)$, where X is a variable, α and β are decision nodes. In the latter case, $dVar(N)$ denotes the variable X .

Definition 2.7 The language BDD is the subset of NNF, where the root of each sentence is a decision node.

The NNF sentence in Figure 2 belongs to the BDD subset.

The BDD language corresponds to *binary decision diagrams (BDDs)*, as known in the formal verification literature [Bryant, 1986]. Binary decision diagrams are depicted using a more compact notation though: the labels *true* and *false* are denoted by 1 and 0, respectively; and each decision node

$(X \wedge \alpha) \vee (\neg X \wedge \beta)$ is denoted by $\alpha \begin{matrix} \text{or} \\ \text{and} \end{matrix} \beta$. The BDD sentence on the left of Figure 2 corresponds to the binary decision diagram on the right of Figure 2. Obviously enough, every NNF sentence that satisfies the decision property is also deterministic. Therefore, BDD is a subset of \bar{d} -NNF.

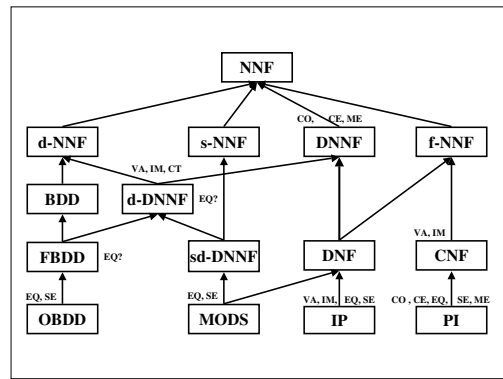


Figure 3: The set of DAG-based languages considered in this paper. An edge $L_1 \rightarrow L_2$ means that L_1 is a subset of L_2 . Next to each subset, we list the polytime queries supported by the subset but not by any of its ancestors (see Section 4).

As we show later, BDD does not qualify as a target compilation language (unless $P=NP$), but the following subset does.

Definition 2.8 FBDD is the intersection of DNNF and BDD.

That is, each sentence in FBDD is decomposable and satisfies the decision property. The FBDD language corresponds to *free binary decision diagrams*, as known in formal verification [Gergov and Meinel, 1994]. An FBDD is usually defined as a BDD that satisfies the *read-once property*: on each path from the root to a leaf, a variable can appear at most once. Read-once is equivalent to the decomposability of BDDs.⁴

A more influential subset of the BDD language is obtained by imposing the *ordering property*:

Definition 2.9 (Ordering) Let $<$ be a total ordering on the variables PS . The language $OBDD_{<}$ is the subset of FBDD satisfying the following property: if N and M are *or*-nodes, and if N is an ancestor of node M , then $dVar(N) < dVar(M)$.

The $OBDD_{<}$ language corresponds to the well-known *ordered binary decision diagrams* [Bryant, 1986].

The languages we have discussed so far are depicted in Figure 3, where arrows denote set inclusion. The figure contains one additional language that we have not discussed yet:

⁴FBDDs are also known as read-once branching programs.

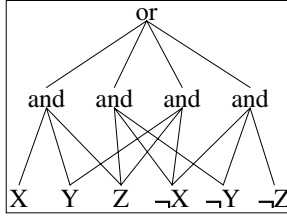


Figure 4: A sentence in language MODS.

Definition 2.10 MODS is the intersection of sd-DNNF and DNF.

Figure 4 depicts a sentence in MODS. As we show later, MODS is the most tractable NNF subset we shall consider (together with $\text{OBDD}_{<}$). This is not surprising since from the syntax of a sentence in MODS, one can immediately recover the sentence models.

3 On the Succinctness of Compiled Theories

We have discussed more than a dozen subsets of the NNF language. Some of these subsets are well known and have been studied extensively in the computer science literature. Others, such as DNNF [Darwiche, 1999b; 1999a] and d-DNNF [Darwiche, 2000], are relatively new. The question now is: What subset should one adopt for a particular application? As we argue in this paper, that depends on three key properties of the language: its succinctness, the class of tractable queries it supports, and the class of tractable transformations it admits.

Our goal in this and the following sections is to construct a map on which we place different subsets of the NNF language according to the above criteria. This map will then serve as a guide to system designers in choosing the target compilation language most suitable to their application. It also provides an example paradigm for studying and evaluating further target compilation languages. We start with a study of succinctness⁵ in this section [Gogic *et al.*, 1995].

Definition 3.1 (Succinctness) Let L_1 and L_2 be two subsets of NNF. L_1 is at least as succinct as L_2 , denoted $L_1 \leq L_2$, iff for every sentence $\alpha \in L_2$, there exists an equivalent sentence $\beta \in L_1$ where the size of β is polynomial in the size of α .⁶

The relation \leq is clearly reflexive and transitive, hence, a pre-ordering. One can also define the relation $<$, where $L_1 < L_2$ iff $L_1 \leq L_2$ and $L_2 \not\leq L_1$.

Proposition 3.1 The results in Table 1 hold.

Figure 5 summarizes the results of Proposition 3.1 in terms of a directed acyclic graph.

A classical result in knowledge compilation states that it is not possible to compile any propositional formula α into a

⁵A more general notion of space efficiency (model preservation for polysize reductions) exists [Cadoli *et al.*, 1996], but we do not need its full generality here.

⁶We stress here that we do not require that there exists a function that computes β given α in *polytime*; we only require that a *polysize* β exists. Yet, our proofs in [Darwiche and Marquis, 2001] contain specific algorithms for computing β from α in certain cases.

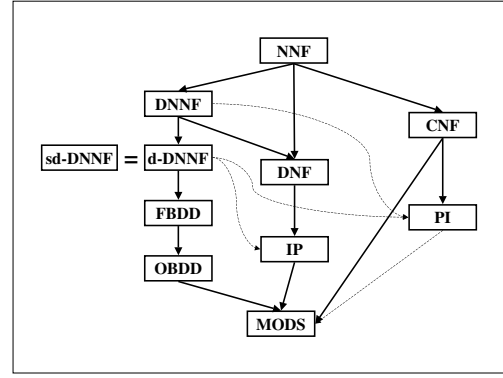


Figure 5: An edge $L_1 \rightarrow L_2$ indicates that L_1 is strictly more succinct than L_2 , $L_1 < L_2$; $L_1 = L_2$ indicates that L_1 and L_2 are equally succinct, $L_1 \leq L_2$ and $L_2 \leq L_1$. Dotted arrows indicate unknown relationships.

polysize data structure β such that: α and β entail the same set of clauses, and clausal entailment on β can be decided in time polynomial in its size, unless $\text{NP} \subseteq \text{P/poly}$ [Selman and Kautz, 1996; Cadoli and Donini, 1997]. This last assumption implies the collapse of the polynomial hierarchy at the second level [Karp and Lipton, 1980], which is considered very unlikely. We use this classical result from knowledge compilation in some of our proofs of Proposition 3.1, which explains why some of its parts are conditioned on the polynomial hierarchy not collapsing.

We have excluded the subsets s-NNF , d-NNF and f-NNF from Table 1 since they do not qualify as target compilation languages (see Section 4). We kept NNF and CNF though given their importance. Consider Figure 5 which depicts Table 1 graphically. With the exception of NNF and CNF, all other languages depicted in Figure 5 qualify as target compilation languages. Moreover, with the exception of language PI , DNNF is the most succinct among all target compilation languages—we know that PI is not more succinct than DNNF, but we do not know whether DNNF is more succinct than PI .

In between DNNF and MODS, there is a succinctness ordering of target compilation languages:

$$\text{DNNF} < \text{d-DNNF} < \text{FBDD} < \text{OBDD}_{<} < \text{MODS}.$$

DNNF is obtained by imposing decomposability on NNF; d-DNNF by adding determinism; FBDD by adding decision; and $\text{OBDD}_{<}$ by adding ordering. Adding each of these properties reduces language succinctness.

One important fact to stress here is that adding smoothness to d-DNNF does not affect its succinctness: the sd-DNNF and d-DNNF languages are equally succinct. It is also interesting to compare sd-DNNF (which is more succinct than the influential FBDD and $\text{OBDD}_{<}$ languages) with MODS, which is a most tractable language. Both sd-DNNF and MODS are smooth, deterministic and decomposable. MODS, however, is flat and obtains its decomposability from the stronger condition of simple-conjunction. Therefore, sd-DNNF can be viewed as the result of relaxing from MODS the flatness and

L	NNF	DNNF	d-DNNF	sd-DNNF	FBDD	OBDD _{<}	DNF	CNF	PI	IP	MODS
NNF	<	<	<	<	<	<	<	<	<	<	<
DNNF	<*	<	<	<	<	<	<	<*	?	<	<
d-DNNF	<*	<*	<	<	<	<	<	<*	<*	?	<
sd-DNNF	<*	<*	<	<	<	<	<	<*	<*	?	<
FBDD	<	<	<	<	<	<	<	<	<	<	<
OBDD _{<}	<	<	<	<	<	<	<	<	<	<	<
DNF	<	<	<	<	<	<	<	<	<	<	<
CNF	<	<	<	<	<	<	<	<	<	<	<
PI	<	<	<	<	<	<	<	<	<	<	?
IP	<	<	<	<	<	<	<	<	<	<	<
MODS	<	<	<	<	<	<	<	<	<	<	<

Table 1: Succinctness of target compilation languages. * means that the result holds unless the polynomial hierarchy collapses.

simple-conjunction conditions, while maintaining decomposability, determinism and smoothness. Relaxing these conditions moves the language three levels up the succinctness hierarchy, although it compromises only the polytime tests for entailment and equivalence as we show in Section 4.

4 Querying a Compiled Theory

In evaluating the suitability of a target compilation language to a particular application, the succinctness of the language must be balanced against the set of queries and transformations that it supports in polytime. We consider in this section a number of queries, each of which returns valuable information about a propositional theory, and then identify target compilation languages which provide polytime algorithms for answering such queries.⁷

The queries we consider are tests for consistency, validity, implicates (clausal entailment), implicants, equivalence, and sentential entailment. We also consider counting and enumerating theory models.⁸

From here on, \mathbf{L} denotes a subset of language NNF.

Definition 4.1 (CO, VA) \mathbf{L} satisfies **CO (VA)** iff there exists a polytime algorithm that maps every formula Σ from \mathbf{L} to 1 if Σ is consistent (valid), and to 0 otherwise.

One of the main applications of compiling a theory is to enhance the efficiency of answering clausal entailment queries:

Definition 4.2 (CE) \mathbf{L} satisfies **CE** iff there exists a polytime algorithm that maps every formula Σ from \mathbf{L} and every clause γ from NNF to 1 if $\Sigma \models \gamma$ holds, and to 0 otherwise.

A key application for clausal entailment is in testing equivalence. Specifically, suppose we have a design expressed as a set of clauses $\Delta^d = \bigwedge_i \alpha_i$ and a specification expressed also as a set of clauses $\Delta^s = \bigwedge_j \beta_j$, and we want to test whether the design and specification are equivalent. By compiling each of Δ^d and Δ^s to targets Γ^d and Γ^s that support a polytime clausal entailment test, we can test the equivalence

⁷We restrict our attention in this paper to the *existence* of polytime algorithms for answering queries, but we do not present the algorithms themselves. The interested reader is referred to [Darwiche, 1999b; 2000; 1999a; Bryant, 1986] for some of these algorithms and to the proofs of theorems in [Darwiche and Marquis, 2001] for others.

⁸One can also consider computing the probability of a propositional sentence, assuming that all variables are probabilistically independent. For the subsets we consider, however, this can be done in polytime whenever models can be counted in polytime.

L	CO	VA	CE	IM	EQ	SE	CT	ME
NNF	o	o	o	o	o	o	o	o
DNNF	√	o	√	o	o	o	o	√
d-DNNF	o	o	o	o	o	o	o	o
s-DNNF	o	o	o	o	o	o	o	o
f-DNNF	o	o	o	o	o	o	o	o
d-DNNF	√	√	√	√	?	o	√	√
sd-DNNF	√	√	√	√	?	o	√	√
BDD	o	o	o	o	o	o	o	o
FBDD	√	√	√	√	?	o	√	√
OBDD _{<}	√	√	√	√	√	√	√	√
DNF	√	o	√	o	o	o	o	√
CNF	o	√	o	√	o	o	o	o
PI	√	√	√	√	√	√	o	√
IP	√	√	√	√	√	√	o	√
MODS	√	√	√	√	√	√	√	√

Table 2: Subsets of the NNF language and their corresponding polytime queries. √ means “satisfies” and o means “does not satisfy unless P = NP.”

of Δ^d and Δ^s in polytime. That is, Δ^d and Δ^s are equivalent iff $\Gamma^d \models \beta_j$ for all j and $\Gamma^s \models \alpha_i$ for all i .

A number of the target compilation languages we shall consider, however, support a direct polytime equivalent test:

Definition 4.3 (EQ, SE) \mathbf{L} satisfies **EQ (SE)** iff there exists a polytime algorithm that maps every pair of formulas Σ, Φ from \mathbf{L} to 1 if $\Sigma \equiv \Phi$ ($\Sigma \models \Phi$) holds, and to 0 otherwise.

Note that sentential entailment (**SE**) is stronger than clausal entailment and equivalence. Therefore, if a language \mathbf{L} satisfies **SE**, it also satisfies **CE** and **EQ**.

For completeness, we consider the following dual to **CE**:

Definition 4.4 (IM) \mathbf{L} satisfies **IM** iff there exists a polytime algorithm that maps every formula Σ from \mathbf{L} and every term γ from NNF to 1 if $\gamma \models \Sigma$ holds, and to 0 otherwise.

Finally, we consider counting and enumerating models:

Definition 4.5 (CT) \mathbf{L} satisfies **CT** iff there exists a polytime algorithm that maps every formula Σ from \mathbf{L} to a nonnegative integer that represents the number of models of Σ (in binary notation).

Definition 4.6 (ME) \mathbf{L} satisfies **ME** iff there exists a polynomial $p(\cdot, \cdot)$ and an algorithm that outputs all models of an arbitrary formula Σ from \mathbf{L} in time $p(n, m)$, where n is the size of Σ and m is the number of its models (over variables occurring in Σ).

Proposition 4.1 *The results in Table 2 hold.*

The results of Proposition 4.1 are summarized in Figure 3. One can draw a number of conclusions based on the results in

this figure. First, NNF, s -NNF, d -NNF, f -NNF, and BDD fall in one equivalence class that does not support any polytime queries and CNF satisfies only **VA** and **IM**; hence, none of them qualifies as a target compilation language in this case. But the remaining languages all support polytime tests for consistency and clausal entailment. Therefore, simply imposing either of smoothness (s -NNF), determinism (d -NNF), flatness (f -NNF), or decision (BDD) on the NNF language does not lead to tractability with respect to any of the queries we consider—neither of these properties seem to be significant in isolation. Decomposability (DNNF), however, is an exception and leads immediately to polytime tests for both consistency and clausal entailment, and to a polytime algorithm for model enumeration.

Recall the succinctness ordering $DNNF < d\text{-DNNF} < FBDD < OBDD_{<} < MODS$ from Figure 5. By adding decomposability (DNNF), we obtain polytime tests for consistency and clausal entailment, in addition to a polytime model enumeration algorithm. By adding determinism to decomposability (d -DNNF), we obtain polytime tests for validity, implicant and model counting, which are quite significant. It is not clear, however, whether the combination of decomposability and determinism leads to a polytime test for equivalence. Moreover, adding the decision property on top of decomposability and determinism (FBDD) does not appear to increase tractability with respect to the given queries⁹, although it does lead to reducing language succinctness as shown in Figure 5. On the other hand, adding the ordering property on top of decomposability, determinism and decision ($OBDD_{<}$), leads to polytime tests for sentential entailment and equivalence.

As for the succinctness ordering $NNF < DNNF < DNF < IP < MODS$ from Figure 5, note that DNNF is obtained by imposing decomposability on NNF, while DNF is obtained by imposing flatness and simple-conjunction (which is stronger than decomposability). What is interesting is that DNF is less succinct than DNNF, yet does not support any more polytime queries; see Figure 3. However, the addition of smoothness on top of flatness and simple-conjunction (MODS) leads to five additional polytime queries, including equivalence and entailment tests.¹⁰

We close this section by noting that determinism appears to be necessary (but not sufficient) for polytime model counting: only deterministic languages, d -DNNF, sd -DNNF, FBDD, $OBDD_{<}$ and MODS, support polytime counting. Moreover, polytime counting implies a polytime test of validity, but the opposite is not true.

⁹Deciding the equivalence of two sentences in FBDD, d -DNNF, or in sd -DNNF, can be easily shown to be in **coNP**. However, we do not have a proof of **coNP**-hardness, nor do we have deterministic polytime algorithms for deciding these problems. Actually, the latter case is quite unlikely as the equivalence problem for FBDD has been intensively studied, with no such algorithm in sight. Note, however, that the equivalence of two sentences in FBDD can be decided probabilistically in polytime [Blum *et al.*, 1980].

¹⁰Formally, we also need determinism to obtain MODS (see Definition 2.10). But given flatness, simple-conjunction and smoothness, we can obtain determinism by simply removing duplicated terms.

5 Transforming a Compiled Theory

A query is an operation that returns information about a theory without changing it. A transformation, on the other hand, is an operation that returns a modified theory, which is then operated on using queries. Many applications require a combination of transformations and queries.

Definition 5.1 ($\wedge C, \vee C$) \mathbf{L} satisfies $\wedge C$ ($\vee C$) iff there exists a polytime algorithm that maps every finite set of formulas $\Sigma_1, \dots, \Sigma_n$ from \mathbf{L} to a formula of \mathbf{L} that represents $\Sigma_1 \wedge \dots \wedge \Sigma_n$ ($\Sigma_1 \vee \dots \vee \Sigma_n$).

Definition 5.2 ($\neg C$) \mathbf{L} satisfies $\neg C$ iff there exists a polytime algorithm that maps every formula Σ from \mathbf{L} to a formula of \mathbf{L} that represents $\neg\Sigma$.

If a language satisfies one of the above properties, we will say that it is *closed* under the corresponding operator. Closure under logical connectives is important for two key reasons. First, it has implications on how compilers are constructed for a given target language. For example, if a clause can be easily compiled into some language \mathbf{L} , then closure under conjunction implies that compiling a CNF sentence into \mathbf{L} is easy. Second, it has implications on the class of polytime queries supported by the target language: If a language \mathbf{L} satisfies **CO** and is closed under negation and conjunction, then it must satisfy **SE** (to test whether $\Delta \models \Gamma$, all we have to do, by the Refutation Theorem, is test whether $\Delta \wedge \neg\Gamma$ is inconsistent). Similarly, if a language satisfies **VA** and is closed under negation and disjunction, it must satisfy **SE** by the Deduction Theorem.

It is important to stress here that some languages are closed under a logical operator, only if the number of operands is bounded by a constant. We will refer to this as *bounded-closure*.

Definition 5.3 ($\wedge BC, \vee BC$) \mathbf{L} satisfies $\wedge BC$ ($\vee BC$) iff there exists a polytime algorithm that maps every pair of formulas Σ and Φ from \mathbf{L} to a formula of \mathbf{L} that represents $\Sigma \wedge \Phi$ ($\Sigma \vee \Phi$).

We now turn to another important transformation:

Definition 5.4 (Conditioning) [Darwiche, 1999a] Let Σ be a propositional formula, and let γ be a consistent term. The conditioning of Σ on γ , noted $\Sigma \mid \gamma$, is the formula obtained by replacing each variable X of Σ by *true* (resp. *false*) if X (resp. $\neg X$) is a positive (resp. negative) literal of γ .

Definition 5.5 (CD) \mathbf{L} satisfies **CD** iff there exists a polytime algorithm that maps every formula Σ from \mathbf{L} and every consistent term γ to a formula from \mathbf{L} that represents $\Sigma \mid \gamma$.

Conditioning has a number of applications, and corresponds to *restriction* in the literature on Boolean functions. The main application of conditioning is due to a theorem, which says that $\Sigma \wedge \gamma$ is consistent iff $\Sigma \mid \gamma$ is consistent [Darwiche, 1999b; 1999a]. Therefore, if a language satisfies **CO** and **CD**, then it must also satisfy **CE**. Conditioning also plays a key role in building compilers that enforce decomposability. If two sentences Δ_1 and Δ_2 are both decomposable (belong to DNNF), their conjunction $\Delta_1 \wedge \Delta_2$ is not necessarily decomposable since the sentences may share variables. Conditioning can be used to ensure decomposability in this case

since $\Delta_1 \wedge \Delta_2$ is equivalent to $\bigvee_{\gamma}(\Delta_1 \mid \gamma) \wedge (\Delta_2 \mid \gamma) \wedge \gamma$, where γ is a term covering all variables shared by Δ_1 and Δ_2 . Note that $\bigvee_{\gamma}(\Delta_1 \mid \gamma) \wedge (\Delta_2 \mid \gamma) \wedge \gamma$ must be decomposable since $\Delta_1 \mid \gamma$ and $\Delta_2 \mid \gamma$ do not mention variables in γ . The previous proposition is indeed a generalization to multiple variables of the well-known Shannon expansion in the literature on Boolean functions. It is also the basis for compiling CNF into DNNF [Darwiche, 1999a; 1999b].

Another critical transformation we shall consider is that of *forgetting* [Lin and Reiter, 1994]:

Definition 5.6 (Forgetting) Let Σ be a propositional formula, and let \mathbf{X} be a subset of variables from PS. The forgetting of \mathbf{X} from Σ , denoted $\exists \mathbf{X}.\Sigma$, is a formula that does not mention any variable from \mathbf{X} and for every formula α that does not mention any variable from \mathbf{X} , we have $\Sigma \models \alpha$ precisely when $\exists \mathbf{X}.\Sigma \models \alpha$.

Therefore, to forget variables from \mathbf{X} is to remove any reference to \mathbf{X} from Σ , while maintaining all information that Σ captures about the complement of \mathbf{X} . Note that $\exists \mathbf{X}.\Sigma$ is unique up to logical equivalence.

Definition 5.7 (FO, SFO) \mathbf{L} satisfies **FO** iff there exists a polytime algorithm that maps every formula Σ from \mathbf{L} and every subset \mathbf{X} of variables from PS to a formula from \mathbf{L} equivalent to $\exists \mathbf{X}.\Sigma$. If the property holds for singleton \mathbf{X} , we say that \mathbf{L} satisfies **SFO**.

Forgetting is an important transformation as it allows us to focus/project a theory on a set of variables. For example, if we know that some variables \mathbf{X} will never appear in entailment queries, we can forget these variables from the compiled theory while maintaining its ability to answer such queries correctly. Another application of forgetting is in counting/enumerating the instantiations of some variables \mathbf{Y} , which are consistent with a theory Δ . This query can be answered by counting/enumerating the models of $\exists \mathbf{X}.\Delta$, where \mathbf{X} is the complement of \mathbf{Y} . Forgetting also has applications to planning, diagnosis and belief revision. For instance, in the SATPLAN framework, compiling away fluents or actions amounts to forgetting variables. In model-based diagnosis, compiling away every variable except the abnormality ones does not remove any piece of information required to compute the conflicts and the diagnoses of a system [Darwiche, 1999b]. Forgetting has also been used to design update operators with valuable properties [Herzig and Rifi, 1999].

Proposition 5.1 *The results in Table 3 hold.*

One can draw a number of observations regarding Table 3. First, all languages we consider satisfy **CD** and, hence, lend themselves to efficient application of the conditioning transformation. As for forgetting multiple variables, only DNNF, DNF, PI and MODS permit that in polytime. It is important to stress here that neither FBDD nor OBDD_< permit polytime forgetting of multiple variables. This is noticeable since some of the recent applications of OBDD_< to planning depends crucially on the operation of forgetting and it may be more suitable to use a language that satisfies **FO** in this case. Note, however, that OBDD_< allows the forgetting of a single variable in polytime, but FBDD does not allow even that. d-DNNF is similar to FBDD as it satisfies neither **FO** nor **SFO**.

L	CD	FO	SFO	$\wedge C$	$\wedge BC$	$\vee C$	$\vee BC$	$\neg C$
NNF	✓	○	✓	✓	✓	✓	✓	✓
DNNF	✓	✓	✓	○	○	✓	✓	○
d-NNF	✓	○	?	✓	✓	?	?	?
s-NNF	✓	○	✓	✓	✓	✓	✓	✓
f-NNF	✓	○	✓	•	✓	•	✓	✓
d-DNNF	✓	○	○	○	○	○	○	?
sd-DNNF	✓	○	○	○	○	○	○	?
BDD	✓	○	✓	✓	✓	✓	✓	✓
FBDD	✓	○	○	•	○	•	○	✓
OBDD _{<}	✓	○	✓	•	✓	•	✓	✓
DNF	✓	✓	✓	•	✓	✓	✓	•
CNF	✓	○	✓	✓	✓	•	✓	•
PI	✓	✓	✓	•	•	•	✓	•
IP	✓	•	○	•	✓	•	•	•
MODS	✓	✓	✓	•	✓	•	•	•

Table 3: Subsets of the NNF language and their polytime transformations. ✓ means “satisfies,” • means “does not satisfy,” while ○ means “does not satisfy unless P=NP.”

It is also interesting to observe that none of the target compilation languages is closed under conjunction. A number of them, however, are closed under bounded-conjunction, including OBDD_<, DNF, IP and MODS.

As for disjunction, the only target compilation languages that are closed under disjunction are DNNF and DNF. The OBDD_< and PI languages, however, are closed under bounded disjunction. Again, the d-DNNF and FBDD languages are closed under neither.

The only target compilation languages that are closed under negation are FBDD and OBDD_<, while it is not known whether d-DNNF or sd-DNNF are closed under this operation. Note that d-DNNF and FBDD support the same set of polytime queries (equivalence checking is unknown for both) so they are indistinguishable from that viewpoint. Moreover, the only difference between the two languages in Table 3 is the closure of FBDD under negation, which does not seem to be that significant in light of no closure under either conjunction or disjunction. Note, however, that d-DNNF is more succinct than FBDD as given in Figure 5.

Finally, OBDD_< is the only target compilation language that is closed under negation, bounded conjunction, and bounded disjunction. This closure actually plays an important role in compiling propositional theories into OBDD_< and is the basis of state-of-the-art compilers for this purpose [Bryant, 1986].

6 Conclusion

The main contribution of this paper is a methodology for analyzing propositional compilation approaches according to two key dimensions: the succinctness of the target compilation language, and the class of queries and transformations it supports in polytime. The second main contribution of the paper is a comprehensive analysis, according to the proposed methodology, of more than a dozen languages for which we have produced a knowledge compilation map, which cross-ranks these languages according to their succinctness, and the polytime queries and transformations they support. This map allows system designers to make informed decisions on which target compilation language to use: after the class of queries/transformations have been decided based on the application of interest, the designer chooses the most succinct target compilation language that supports such operations in

polytime. Another key contribution of this paper is the uniform treatment we have applied to diverse target compilation languages, showing how they all are subsets of the NNF language. Specifically, we have identified a number of simple, yet meaningful, properties, including decomposability, determinism, decision and flatness, and showed how combinations of these properties give rise to different target compilation languages. The studied subsets include some well known languages such as PI , which has been influential in AI; $\text{OBDD}_{<}$, which has been influential in formal verification; and CNF and DNF, which have been quite influential in computer science. The subsets also include some relatively new languages such as DNNF and \bar{d} -DNNF, which appear to represent interesting, new balances between language succinctness and query/transformation tractability.

Acknowledgements

We wish to thank Ingo Wegener for his help with some of the issues discussed in the paper. This work has been done while the second author was a visiting researcher with the Computer Science Department at UCLA. The first author has been partly supported by NSF grant IIS-9988543 and MURI grant N00014-00-1-0617. The second author has been partly supported by IUT de Lens and the Région Nord/Pas-de-Calais.

References

- [Blum *et al.*, 1980] Manuel Blum, Ashok K. Chandra, and Mark N. Wegman. Equivalence of free Boolean graphs can be decided probabilistically in polynomial time. *Information Processing Letters*, 10(2):80–82, 1980.
- [Boufkhad *et al.*, 1997] Y. Boufkhad, E. Grégoire, P. Marquis, B. Mazure, and L. Saïs. Tractable cover compilations. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 122–127, 1997.
- [Bryant, 1986] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.
- [Cadoli and Donini, 1997] M. Cadoli and F.M. Donini. A survey on knowledge compilation. *AI Communications*, 10:137–150, 1997 (printed in 1998).
- [Cadoli *et al.*, 1996] M. Cadoli, F.M. Donini, P. Liberatore, and M. Schaerf. Comparing space efficiency of propositional knowledge representation formalisms. In *Proc. of International Conference on Knowledge Representation and Reasoning (KR'96)*, pages 364–373, 1996.
- [Darwiche and Marquis, 2001] Adnan Darwiche and Pierre Marquis. A perspective on knowledge compilation. Technical Report D-116, Cognitive Systems Laboratory, UCLA, Ca 90095, 2001.
- [Darwiche, 1999a] Adnan Darwiche. Compiling knowledge into decomposable negation normal form. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 284–289, 1999.
- [Darwiche, 1999b] Adnan Darwiche. Decomposable negation normal form. Technical Report D-109, Cognitive Systems Laboratory, UCLA, Ca 90095, 1999. To appear in *Journal of the ACM*.
- [Darwiche, 2000] Adnan Darwiche. On the tractable counting of theory models and its application to belief revision and truth maintenance. Technical Report D-113, Cognitive Systems Laboratory, UCLA, Ca 90095, 2000. To appear in *Journal of Applied Non-Classical Logics*.
- [Dechter and Rish, 1994] R. Dechter and I. Rish. Directional resolution: the Davis-Putnam procedure, revisited. In *Proc. of International Conference on Knowledge Representation and Reasoning (KR'94)*, pages 134–145, 1994.
- [del Val, 1994] A. del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In *Proc. of International Conference on Knowledge Representation and Reasoning (KR'94)*, pages 551–561, 1994.
- [Gergov and Meinel, 1994] J. Gergov and Ch. Meinel. Efficient analysis and manipulation of OBDD's can be extended to FBDD's. *IEEE Transactions on Computers*, 43(10):1197–1209, 1994.
- [Gogic *et al.*, 1995] G. Gogic, H. Kautz, Ch. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 862–869, 1995.
- [Herzig and Rifi, 1999] A. Herzig and O. Rifi. Propositional belief base update and minimal change. *Artificial Intelligence*, 115(1):107–138, 1999.
- [Karp and Lipton, 1980] R.M. Karp and R.J. Lipton. Some connections between non-uniform and uniform complexity classes. In *Proc. of ACM Symposium on Theory of Computing (STOC'80)*, pages 302–309, 1980.
- [Khardon and Roth, 1997] R. Khardon and D. Roth. Learning to reason. *Journal of the ACM*, 44(5):697–725, 1997.
- [Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it! In *Proc. of the AAAI Fall Symposium on Relevance*, pages 154–159, New Orleans, 1994.
- [Marquis, 1995] Pierre Marquis. Knowledge compilation using theory prime implicates. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 837–843, 1995.
- [Papadimitriou, 1994] Ch. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
- [Reiter and de Kleer, 1987] Ray Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proc. of National Conference on Artificial Intelligence (AAAI'87)*, pages 183–188, 1987.
- [Schrag, 1996] R. Schrag. Compilation for critically constrained knowledge bases. In *Proc. of National Conference on Artificial Intelligence (AAAI'96)*, pages 510–515, 1996.
- [Selman and Kautz, 1996] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

Phase Transitions of PP-Complete Satisfiability Problems*

Delbert D. Bailey, Víctor Dalmau, Phokion G. Kolaitis

Computer Science Department

University of California, Santa Cruz

Santa Cruz, CA 95064, U.S.A

{dbailey, dalmau, kolaitis}@cse.ucsc.edu

Abstract

The complexity class PP consists of all decision problems solvable by polynomial-time probabilistic Turing machines. It is well known that PP is a highly intractable complexity class and that PP-complete problems are in all likelihood harder than NP-complete problems. We investigate the existence of phase transitions for a family of PP-complete Boolean satisfiability problems under the fixed clauses-to-variables ratio model. A typical member of this family is the decision problem $\#3SAT(\geq 2^{n/2})$: given a 3CNF-formula, is it satisfied by at least the square-root of the total number of possible truth assignments? We provide evidence to the effect that there is a critical ratio $r_{3,2}$ at which the asymptotic probability of $\#3SAT(\geq 2^{n/2})$ undergoes a phase transition from 1 to 0. We obtain upper and lower bounds for $r_{3,2}$ by showing that $0.9227 \leq r_{3,2} \leq 2.595$. We also carry out a set of experiments on random instances of $\#3SAT(\geq 2^{n/2})$ using a natural modification of the Davis-Putnam-Logemann-Loveland (DPLL) procedure. Our experimental results suggest that $r_{3,2} \approx 2.5$. Moreover, the average number of recursive calls of this modified DPLL procedure reaches a peak around 2.5 as well.

1 Introduction and Summary of Results

During the past several years, there has been an intensive investigation of random Boolean satisfiability in probability spaces parametrized by a fixed clauses-to-variables ratio. More precisely, if $k \geq 2$ is an integer, n is a positive integer and r is a positive rational such that rn is an integer, then $F_k(n, r)$ denotes the space of random k CNF-formulas with n variables x_1, \dots, x_n and rn clauses that are generated uniformly and independently by selecting k variables without replacement from the n variables and then negating each variable with probability $1/2$. Much of the work in this area is aimed at establishing or at least providing evidence for the conjecture, first articulated by [Chvátal and

Reed, 1992], that a phase transition occurs in the probability $p_k(n, r)$ of a random formula in $F_k(n, r)$ being satisfiable, as $n \rightarrow \infty$. Specifically, this conjecture asserts that, for every $k \geq 2$, there is a positive real number r_k such that if $r < r_k$, then $\lim_{n \rightarrow \infty} p_k(n, r) = 1$, whereas if $r > r_k$, then $\lim_{n \rightarrow \infty} p_k(n, r) = 0$.

So far, this conjecture has been established only for $k = 2$ by showing that $r_2 = 1$ [Chvátal and Reed, 1992; Fernandez de la Vega, 1992; Goerdts, 1996]. For $k \geq 3$, upper and lower bounds for r_k have been obtained analytically and experiments have been carried out that provide evidence for the existence of r_k and estimate its actual value. For $k = 3$, in particular, it has been proved that $3.26 \leq r_3 \leq 4.596$ [Achlioptas and Sorkin, 2000; Janson *et al.*, 2000] and extensive experiments have suggested that $r_3 \approx 4.2$ [Selman *et al.*, 1996]. Moreover, the experiments reveal that the median running time of the Davis-Putnam-Logemann-Loveland (DPLL) procedure for satisfiability attains a peak around 4.2. Thus, the critical ratio at which the probability of satisfiability undergoes a phase transition coincides with the ratio at which this procedure requires maximum computational effort to decide whether a random formula is satisfiable.

Boolean satisfiability is the prototypical NP-complete problem. Since many reasoning and planning problems in artificial intelligence turn out to be complete for complexity classes beyond NP, in recent years researchers have embarked on an investigation of phase transitions for such problems. For instance, it is known that STRIPS planning is complete for the class PSPACE of all polynomial-space solvable problems [Bylander, 1994]. A probabilistic analysis of STRIPS planning and an experimental comparison of different algorithms for this problem have been carried out in [Bylander, 1996]. In addition to STRIPS planning, researchers have also investigated phase transitions for the prototypical PSPACE-complete problem QSAT, which is the problem of evaluating a given quantified Boolean formula [Cadoli *et al.*, 1997; Gent and Walsh, 1999]. Actually, this investigation has mainly focused on the restriction of QSAT to random quantified Boolean formulas with two alternations (universal-existential) of quantifiers, a restriction which forms a complete problem for the class Π_2P at the second level of the polynomial hierarchy PH. The lowest level of PH is NP, while higher levels of this hierarchy consist of all decision

*Research of the authors was partially supported by NSF Grants No. CCR-9610257, CCR-9732041, and IIS-9907419

problems (or of the complements of all decision problems) computable by nondeterministic polynomial-time Turing machines using oracles from lower levels (see [Papadimitriou, 1994] for additional information on PH and its levels). Another PSPACE-complete problem closely related to QSAT is stochastic Boolean satisfiability SSAT, which is the problem of evaluating an expression consisting of existential and randomized quantifiers applied to a Boolean formula. Experimental results on phase transitions for SSAT have been reported in [Littman, 1999] and [Littman *et al.*, 2001].

Between NP and PSPACE lie several other important complexity classes that contain problems of significance in artificial intelligence. Two such classes, closely related to each other and of interest to us here, are #P and PP. The class #P, introduced and first studied by [Valiant, 1979a; 1979b], consists of all functions that count the number of accepting paths of nondeterministic polynomial-time Turing machines. The prototypical #P-complete problem is #SAT, i.e., the problem of counting the number of truth assignments that satisfy a CNF-formula. It is well known that numerous #P-complete problems arise naturally in logic, algebra, and graph theory [Valiant, 1979a; 1979b]. Moreover, #P-complete problems are encountered in artificial intelligence; these include the problem of computing Dempster's rule for combining evidence [Orponen, 1990] and the problem of computing probabilities in Bayesian belief networks [Roth, 1996]. Recently, researchers have initiated an experimental investigation of extensions of the DPLL procedure for solving #SAT. Specifically, a procedure for solving #SAT, called Counting Davis-Putnam (CDP), was presented and experiments on random 3CNF formulas from the space $F_3(n, r)$ were carried out in [Birnbbaum and Lozinskii, 1999]. The main experimental finding was that the median running time of CDP reaches its peak when $r \approx 1.2$. A different DPLL extension for solving #SAT, called Decomposing Davis-Putnam (DDP), was presented in [Bayardo and Pehoushek, 2000]; this procedure is based on recursively identifying connected components in the constraint graph associated with a CNF-formula. Additional experiments on random 3CNF-formulas from $F_3(n, r)$ were conducted and it was found out that the median running time of DDP reaches its peak when $r \approx 1.5$.

In the case of the NP-complete problems k SAT, $k \geq 3$, the peak in the median running time of the DPLL procedure occurs at the critical ratio at which the probability of satisfiability appears to undergo a phase transition. Since #SAT is a counting problem (returning numbers as answers) and not a decision problem (returning "yes" or "no" as answers), it is not meaningful to associate with it a probability of getting a "yes" answer; therefore, it does not seem possible to correlate the peak in the median running times of algorithms for #SAT with a structural phase transition of #SAT. Nonetheless, there exist decision problems that in a certain sense embody the intrinsic computational complexity of #P-complete problems. These are the problems that are complete for the class PP of all decision problems solvable using a polynomial-time *probabilistic Turing machine*, i.e., a polynomial-time nondeterministic Turing machine M that accepts a string x if and only if at least half of the computations of M on input x are accepting. The class PP was first studied by [Simon, 1975]

and [Gill, 1977], where several problems were shown to be PP-complete under polynomial-time reductions. In particular, the following decision problem, also called #SAT, is PP-complete: given a CNF-formula φ and a positive integer i , does φ have at least i satisfying truth assignments? This problem constitutes the decision version of the counting problem #SAT, which justifies the innocuous overload of notation. Another canonical PP-complete problem, which is actually a special case of #SAT, is MAJORITY SAT: given a CNF-formula, is it satisfied by at least half of the possible truth assignments to its variables? In addition, several evaluation and testing problems in probabilistic planning under various domain representations have recently been shown to be PP-complete [Littman *et al.*, 1998].

It is known that the class PP contains both NP and coNP, and is contained in PSPACE (see [Papadimitriou, 1994]). Moreover, as pointed out by [Angluin, 1980], there is a tight connection between #P and PP. Specifically, $P^{\#P} = P^{PP}$, which means that the class of decision problems computable in polynomial time using #P oracles coincides with the class of decision problems computable in polynomial time using PP oracles. This is precisely the sense in which PP-complete problems embody the same intrinsic computational complexity as #P-complete problems. Moreover, PP-complete problems (and #P-complete problems) are considered to be substantially harder than NP-complete problems, since in a technical sense they dominate all problems in the polynomial hierarchy PH. Indeed, the main result in [Toda, 1989] asserts that $PH \subseteq P^{PP} = P^{\#P}$. In particular, Toda's result implies that no PP-complete problem lies in PH, unless PH collapses at one of its levels, which is considered to be a highly improbable state of affairs in complexity theory.

In [Littman, 1999], initial experiments were carried out to study the median running time of an extension of the DPLL procedure on instances (φ, i) of the PP-complete decision problem #SAT in which φ was a random 3CNF-formula drawn from $F_3(n, rn)$ and $i = 2^t$, for some nonnegative integer $t \leq n$. These experiments were also reported in [Littman *et al.*, 2001], which additionally contains a discussion on possible phase transitions for the decision problem #SAT and preliminary results concerning coarse upper and lower bounds for the critical ratios at which phase transitions may occur (in these two papers #SAT is called MAJSAT). As noted earlier, the main emphasis of both [Littman, 1999] and [Littman *et al.*, 2001] is not on #SAT or on PP-complete problems, but on stochastic Boolean satisfiability SSAT, which is a PSPACE-complete problem containing #SAT as a special case.

In this paper, we embark on a systematic investigation of phase transitions for a large family of PP-complete satisfiability problems. Specifically, for every integer $k \geq 3$ and every integer $t \geq 2$, let #kSAT($\geq 2^{n/t}$) be the following decision problem: given a k CNF-formula φ with n variables, does φ have at least $2^{n/t}$ satisfying truth assignments? In particular, for $t = 2$ and for every $k \geq 3$, we have the decision problem #kSAT($\geq 2^{n/2}$): given a k CNF-formula, is it satisfied by at least the square-root of the total number of possible truth assignments? Clearly, each problem in this family is a restriction of the decision problem #SAT. Note

that, while an instance of #SAT is a pair (φ, i) , an instance of #kSAT ($\geq 2^{n/t}$) is just a k CNF-formula φ ; this makes it possible to study the behavior of random #kSAT ($\geq 2^{n/t}$) in the same framework as the one used for random k SAT. One may also consider the behavior of random MAJORITY k SAT, $k \geq 3$. In Section 3.2, however, we observe that the asymptotic behavior of random MAJORITY k SAT is trivial and that, in particular, it does not undergo any phase transition. In contrast, the state of affairs for random #kSAT ($\geq 2^{n/t}$) will turn out to be by far more interesting.

We first show that, for every $k \geq 3$ and every $t \geq 2$, the problem #kSAT ($\geq 2^{n/t}$) is indeed PP-complete. We conjecture that each of these problems undergoes a phase transition at some critical ratio $r_{k,t}$ of clauses to variables: as $n \rightarrow \infty$, for ratios $r < r_{k,t}$, almost all formulas in $F_k(n, r)$ are “yes” instances of #kSAT ($\geq 2^{n/t}$), whereas for ratios $r > r_{k,t}$, almost all formulas in $F_k(n, r)$ are “no” instances of #kSAT ($\geq 2^{n/t}$). As a first step towards this conjecture, we establish analytically upper and lower bounds for $r_{k,t}$. A standard application of Markov’s inequality easily yields that $\frac{(t-1)}{t} \frac{1}{(k-\log(2^k-1))}$ is an upper bound for $r_{k,t}$ (this was also implicit in [Littman *et al.*, 2001]). Using an elementary argument and the fact that the probability of satisfiability of random 2CNF-formulas undergoes a phase transition at $r_2 = 1$, we show that $(1 - 1/t)$ is a coarse lower bound for $r_{k,t}$. In particular, these results imply that the critical ratio $r_{3,2}$ of #3SAT ($\geq 2^{n/2}$) obeys the following bounds: $0.5 \leq r_{3,2} \leq 2.595$. After this, we analyze a randomized algorithm, called Extended Unit Clause (EUC), for #3SAT ($\geq 2^{n/2}$) and show that it almost surely returns a “yes” answer when $r < 0.9227$; therefore, $r_{3,2} \geq 0.9227$. Although EUC is a simple heuristic, its analysis is rather complex. This analysis is carried out by adopting and extending the powerful methodology of differential equations, first used by [Achlioptas, 2000] to derive improved lower bounds for the critical ratio r_3 of random 3CNF-formulas.

Finally, we complement these analytical results with a set of experiments for #3SAT ($\geq 2^{n/2}$) by implementing a modification of the Counting Davis-Putnam procedure (CDP) and running it on formulas drawn from $F_3(n, rn)$. Our experimental results suggest that the probability of #3SAT ($\geq 2^{n/2}$) undergoes a phase transition when $r \approx 2.5$. Thus, the 2.595 upper bound for $r_{3,2}$ obtained using Markov’s inequality turns out to be remarkably close to the value of $r_{3,2}$ suggested by the experiments. Moreover, the average number of recursive calls of the modified CDP procedure reaches a peak around the same critical ratio 2.5.

2 PP-completeness of #kSAT ($\geq 2^{n/t}$)

In Valiant [Valiant, 1979a], the counting problem #SAT was shown to be #P-complete via *parsimonious* reductions, i.e., every problem in #P can be reduced to #SAT via a polynomial-time reduction that preserves the number of solutions. Moreover, the same holds true for the counting versions of many other NP-complete problems, including #kSAT, the restriction of #SAT to k CNF-formulas. We now use this fact to identify a large family of PP-complete problems.

Proposition 2.1: For every integer $k \geq 3$ and every integer $t \geq 2$, the decision problem #kSAT ($\geq 2^{n/t}$) is PP-complete. In particular, #3SAT ($\geq 2^{n/2}$) is PP-complete.

Proof: For concreteness, in what follows we show that #3SAT ($\geq 2^{n/2}$) is PP-complete. Let Q be the following problem: given a 3CNF-formula ψ and a positive integer i , does ψ have at least 2^i satisfying truth assignments? Since #3SAT is a #P-complete problem under parsimonious reductions, there is a polynomial-time transformation such that, given a CNF-formula φ with variables x_1, \dots, x_n , it produces a 3CNF-formula ψ whose variables include x_1, \dots, x_n and has the same number of satisfying truth assignments as φ . Consequently, φ is a “yes” instance of MAJORITY SAT (i.e., it has at least 2^{n-1} satisfying truth assignments) if and only if $(\psi, n-1)$ is a “yes” instance of Q . Consequently, Q is PP-complete.

We now show that there is a polynomial-time reduction of Q to #3SAT ($\geq 2^{n/2}$). Given a 3CNF-formula ψ with variables x_1, \dots, x_n and a positive integer i , we can construct in polynomial time a 3CNF-formula χ with variables $x_1, \dots, x_n, y_1, \dots, y_n$ that is tautologically equivalent to the CNF-formula $\psi \wedge y_{n-i+1} \wedge \dots \wedge y_n$. It is clear that $\#\chi = 2^{n-i} \#\psi$, where $\#\chi$ and $\#\psi$ denote the numbers of truth assignments that satisfy χ and ψ respectively. Consequently, ψ has at least 2^i satisfying truth assignments if and only if χ has at least $2^n = 2^{2n/2}$ satisfying truth assignments. ■

3 Upper and Lower Bounds for #kSAT ($\geq 2^{n/t}$)

Let $X_k^{n,r}$ be the random variable on $F_k(n, r)$ such that $X_k^{n,r}(\varphi)$ is the number of truth assignments on x_1, \dots, x_n that satisfy φ , where φ is a random k CNF-formula in $F_k(n, r)$. Thus, φ is a “yes” instance of #kSAT ($\geq 2^{n/t}$) if and only if $X_k^{n,r}(\varphi) \geq 2^{n/t}$. We now have all the notation in place to formulate the following conjecture for the family of problems #kSAT ($\geq 2^{n/t}$), where $k \geq 3$ and $t \geq 2$.

Conjecture 3.1: For every integer $k \geq 3$ and every integer $t \geq 2$, there is a positive real number $r_{k,t}$ such that:

- If $r < r_{k,t}$, then $\lim_{n \rightarrow \infty} \Pr[X_k^{n,r} \geq 2^{n/t}] = 1$.
- If $r > r_{k,t}$, then $\lim_{n \rightarrow \infty} \Pr[X_k^{n,r} \geq 2^{n/t}] = 0$.

We have not been able to settle this conjecture, which appears to be as difficult as the conjecture concerning phase transitions of random k SAT, $k \geq 3$. In what follows, however, we establish certain analytical results that yield upper and lower bounds for the value of $r_{k,t}$; in particular, these results demonstrate that the asymptotic behavior of random #kSAT ($\geq 2^{n/t}$) is non-trivial.

3.1 Upper Bounds for #kSAT ($\geq 2^{n/t}$)

Let X be a random variable taking nonnegative values and having finite expectation $E(X)$. Markov’s inequality is a basic result in probability theory which asserts that if s is a positive real number, then $\Pr[X \geq s] \leq \frac{E(X)}{s}$. The special case of this inequality with $s = 1$ has been used in the past to obtain a coarse upper bound for the critical ratio r_k in random k SAT. We now use the full power of Markov’s inequality to

obtain an upper bound for $r_{k,t}$. As usual, $\lg(x)$ denotes the logarithm of x in base 2.

Proposition 3.2: *Let $k \geq 3$ and $t \geq 2$ be two integers. For every positive rational number $r > \frac{(t-1)}{t} \frac{1}{(k-\lg(2^k-1))}$,*

$$\lim_{n \rightarrow \infty} \Pr[X_k^{n,r} \geq 2^{n/t}] = 0.$$

It follows that if $r_{k,t}$ exists, then $r_{k,t} \leq \frac{(t-1)}{t} \frac{1}{(k-\lg(2^k-1))}$. In particular, $r_{3,2} \leq \frac{1}{2} \frac{1}{(3-\lg 7)} \approx 2.595$.

Proof: For every truth assignment α on the variables x_1, \dots, x_n , let I_α be the random variable on $F_k(n, r)$ such that $I_\alpha(\varphi) = 1$, if α satisfies φ , and $I_\alpha(\varphi) = 0$, otherwise. Each I_α is a Bernoulli random variable with mean $(1 - 1/2^k)^{rn}$. Since $X_k^{n,r} = \sum_\alpha I_\alpha$, the linearity of expectation implies that $E(X_k^{n,r}) = (1 - 1/2^k)^{rn} 2^n$. By Markov's inequality, we have that

$$\Pr[X_k^{n,r} \geq 2^{n/t}] \leq (1 - 1/2^k)^{rn} 2^{(1-1/t)n}.$$

It follows that if r is such that $(1 - 1/2^k)^r 2^{(1-1/t)} < 1$, then $\lim_{n \rightarrow \infty} \Pr[X_k^{n,r} \geq 2^{n/t}] = 0$. The result then is obtained by taking logarithms in base 2 in both sides of the above inequality and solving for r . ■

Several remarks are in order now. First, note if k is kept fixed while t is allowed to vary, then the smallest upper bound is obtained when $t = 2$. Moreover, the quantity $\frac{1}{(k-\lg(2^k-1))}$ is the coarse upper bound for the critical ratio r_k for random k SAT obtained using Markov's inequality. In particular, for random 3SAT this bound is ≈ 5.91 , which is twice the bound for $r_{3,2}$ given by Proposition 3.2.

Let MAJORITY k SAT be the restriction of MAJORITY SAT to k CNF-formulas, $k \geq 2$. Obviously, a formula φ in $F_k(n, r)$ is a "yes" instance of MAJORITY k SAT if and only if $X_k^{n,r}(\varphi) \geq 2^{n-1}$. Markov's inequality implies that $\Pr[X_k^{n,r} \geq 2^{n-1}] \leq 2(1 - 1/2^k)^{rn}$, from which it follows that $\lim_{n \rightarrow \infty} \Pr[X_k^{n,r} \geq 2^{n-1}] = 0$, for every $k \geq 2$. Thus, for every $k \geq 2$, the asymptotic behavior of random MAJORITY k SAT is trivial; in particular, MAJORITY k SAT does not undergo any phase transition.

3.2 Lower Bounds for #kSAT($\geq 2^{n/t}$)

We say that a partial truth assignment α covers a clause c if it satisfies at least one of the literals of c . We also say that α covers a CNF-formula φ with n variables if α covers every clause of φ . Perhaps the simplest sufficient condition for φ to have at least $2^{n/t}$ satisfying truth assignments is to ensure that there is a partial assignment over $\lfloor n - n/t \rfloor$ variables covering φ . The next proposition shows that if r is small enough, then this sufficient condition is almost surely true for formulas in $F_k(n, r)$, as $n \rightarrow \infty$.

Proposition 3.3: *Let $k \geq 3$ and $t \geq 2$ be two integers. If $0 < r < 1 - 1/t$, then, as $n \rightarrow \infty$, almost all formulas in $F_k(n, r)$ are covered by a partial truth assignment on $\lfloor n - n/t \rfloor$ variables. Consequently, if $0 < r < 1 - 1/t$, then*

$$\lim_{n \rightarrow \infty} \Pr[X_k^{n,r} \geq 2^{n/t}] = 1.$$

It follows that if $r_{k,t}$ exists, then $r_{k,t} \geq 1 - 1/t$. In particular, $r_{3,2} \geq 0.5$.

Proof: In [Chvátal and Reed, 1992; Fernandez de la Vega, 1992; Goerdt, 1996]), it was shown that if $r < 1$, then 2CNF-formulas in $F_{2,n,r}$ are satisfiable with asymptotic probability 1. Fix a ratio $r < 1 - 1/t$ and consider a random formula φ in $F_k(n, r)$. By removing $(k - 2)$ literals at random from every clause of φ , we obtain a random 2CNF-formula φ^* which is almost surely satisfiable. Let β be a satisfying truth assignment of φ^* and let α be the partial truth assignment obtained from β by taking for each clause a literal satisfied by α . Since $r < 1 - 1/t$, we have that α is a truth assignment on $\lfloor n - n/t \rfloor$ variables that covers φ^* ; hence, α covers φ as well. ■

The preceding Propositions 3.2 and 3.3 imply that, unlike MAJORITY k SAT, for every $k \geq 3$ and every $t \geq 2$, the asymptotic behavior of #kSAT($\geq 2^{n/t}$) is non-trivial.

3.3 An Improved Lower Bound for #3SAT($\geq 2^{n/2}$)

In what follows, we focus on #3SAT($\geq 2^{n/2}$). So far, we have established that if $r_{3,2}$ exists, then $0.5 \leq r_{3,2} \leq 2.595$. The main result is an improved lower bound for $r_{3,2}$.

Theorem 3.4: *For every positive real number $r < 0.9227$,*

$$\lim_{n \rightarrow \infty} \Pr[X_3^{n,r} \geq 2^{n/2}] = 1.$$

It follows that if $r_{3,2}$ exists, then $r_{3,2} \geq 0.9227$.

The remainder of this section is devoted to a discussion of the methodology used and an outline of the proof of this result. We adopt an algorithmic approach, which originated in [Chao and Franco, 1986] and has turned out to be very fruitful in establishing lower bound for the critical ratio r_3 of random 3SAT (see [Achlioptas, 2001] for an overview). We consider a particular randomized algorithm, called Extended Unit Clause (EUC), that takes as an input a 3CNF-formula φ on n variables and attempts to construct a *small* partial assignment α covering all clauses of φ . Algorithm EUC succeeds if the number of variables assigned by α is $\leq \lfloor n/2 \rfloor$, and fails otherwise. Our goal is to show that algorithm EUC succeeds almost surely on formulas φ from $F_3(n, r)$ for each $r < 0.9227$. Consequently, $r_{3,2} \geq 0.9227$.

EUC Algorithm:

For $t := 1$ to n do

If there are any 1-clauses, (forced step)

pick a 1-clause uniformly at random and satisfy it.

Otherwise, (free step)

pick an unassigned variable uniformly at random and remove all literals involving that variable in all remaining clauses.

Return *true* if the number of assigned variables is $\leq \lfloor n/2 \rfloor$;

otherwise, return *false*.

To analyze the average performance of algorithm EUC we use the *differential equations methodology* (DEM), initially introduced in [Achlioptas, 2000] and described more extensively in [Achlioptas, 2001]. Due to space limitations, we give only a high-level description of how DEM can be applied to the analysis of algorithm EUC and also outline the steps in the derivation of the improved lower bound 0.9227.

Since DEM was developed to analyze satisfiability testing algorithms, it should not be surprising that certain modifications are needed so that it can be applied to counting algorithms, such as EUC. The main component of EUC not handled directly by DEM is the *free step*, since in a satisfiability context it is always a better strategy to assign a value to the selected variable, instead of removing all the literals involving that variable. We will describe *where* and *how* we extend DEM to handle free steps.

Let $V(t)$ be the random set of variables remaining at iteration t ($0 \leq t \leq n$) and let $S_i(t)$ denote the set of random i -clauses ($1 \leq i \leq 3$) remaining at iteration t . To trace the value of $|S_i(t)|$, we rely on the assumption that, at every iteration of the execution of the algorithm being considered, a property called *uniform randomness* is maintained. This property asserts that in every iteration $0 \leq t \leq n$, conditional on $|V(t)| = n'$ and $|S_i(t)| = m'$, $S_i(t)$ is drawn from $F_i(n', m')$. In [Achlioptas, 2001], a protocol, called *card game*, is presented; this protocol restricts the possible ways in which a variable can be selected and assigned. It is shown that any algorithm obeying that protocol, satisfies the uniform randomness property. Unfortunately, due to the presence of the free steps, algorithm EUC does not satisfy the card protocol, unlike the majority of algorithms for satisfiability analyzed so far. It is tedious, but straightforward, to show that the natural generalization of the card game in which we allow the elimination of the literals involving the selected variable guarantees the uniform randomness property.

We analyze the algorithm EUC by studying the evolution of the random variables that count the number of clauses in $S_i(t)$, $C_i(t) = |S_i(t)|$. We also need a random variable $F(t)$ that counts the number of variables assigned up to iteration t . We trace the evolution of $C_i(t)$ and $F(t)$ by using a result in [Wormald, 1995], which states that if a set of random variables $X_j(t)$, $1 \leq j \leq k$ evolving jointly with t such that (1) in each iteration t , the random variable $\Delta X_j(t) = X_j(t+1) - X_j(t)$ with high probability (w.h.p.) is very close to its expectation and (2) $X_j(t)$ evolves smoothly with t , then the entire evolution of $X_j(t)$ will remain close to its mean path, that is, the path that $X_j(t)$ would follow if $\Delta X_j(t)$ was, in each iteration, the value of its expectation. Furthermore, this mean path can be expressed as the set of solutions of a system of differential equations obtained by considering the scaled version of the space-state of the process obtained by dividing every parameter by n .

As discussed in [Achlioptas, 2001], Wormald's theorem guarantees that the value of the random variables considered differs in $o(n)$ from its mean path. This possible deviation produces some difficulties in our analysis; indeed, at each iteration we need to know the value of $C_1(t)$ with far more precision, because depending on the *exact* value taken by $C_1(t)$ algorithm EUC performs different operations. To settle this technical difficulty, Achlioptas derived an elegant solution, called the *lazy-server* lemma. Intuitively, this lemma states that the aforementioned difficulty can be overcome if, instead of handling unit clauses deterministically as soon as they appear, at iteration t we take care of unit clauses with probability p and perform a free step with probability $(1 - p)$, where p has to be chosen appropriately.

An additional technical difficulty remains. As discussed in [Achlioptas, 2001], condition (2) of Wormald's theorem does not hold when the iteration t is getting close to n . This second problem is fixed by determining an iteration $t^* = \lfloor (1 - \epsilon)n \rfloor$ at which our algorithm will stop the iterative process and it will deal with the remaining formula in a deterministic fashion. We now modify algorithm EUC by incorporating the features described above and obtain the following algorithm:

EUC with lazy-server policy:

For $t := 1$ to t^* do

Set $U(t) = 1$ with probability p

1. If $U(t) = 1$

a) If there are 1-clauses,

pick a 1-clause uniformly at random and satisfy it.

b) Otherwise

pick an unset variable uniformly at random and assign it uniformly at random

2. Otherwise

Pick an unset variable uniformly at random and remove all literals with that underlying variable in all remaining clauses.

Find a minimal covering for the remaining clauses.

Return *true* if the number of assigned variables is $\lfloor n/2 \rfloor$; otherwise, return *false*.

Next, we compute the equation that determine the expected value for the evolution of $C_2(t)$, $C_3(t)$ and $F(t)$, conditional on the history of the random variables considered up to iteration t . Let $\mathbf{H}(t)$ be a random variable representing this history (i.e., $\mathbf{H}(t) = \langle C(0), \dots, C(t) \rangle$, where $C(t) = (C_2(t), C_3(t), F(t))$). Since $S_i(t)$ distributes as $F_i(n - t, C_i(t))$, the expected number of clauses in $C_i(t)$ containing a given literal l is equal to $iC_i(t)/2(n - t)$. Using this, we obtain the following system of equations describing the evolution of $C_2(t)$, $C_3(t)$ and $F(t)$.

$$\mathbf{E}(\Delta C_3(t) | \mathbf{H}(t)) = -\frac{3C_3(t)}{n-t}$$

$$\mathbf{E}(\Delta C_2(t) | \mathbf{H}(t)) = -\frac{2C_2(t)}{n-t} + p\frac{3C_3(t)}{2(n-t)} + (1-p)\frac{3C_3(t)}{n-t}$$

$$\mathbf{E}(\Delta F(t) | \mathbf{H}(t)) = p$$

with initial conditions $C_2(0) = 0$, $C_3(0) = rn$, $F(0) = 0$.

It is time to fix the value of p . According to the lazy-server lemma any value such that

$$p > p\frac{C_2(t)}{n-t} + (1-p)\frac{2C_2(t)}{n-t}$$

would suffice. This inequality is solved by setting

$$p = (1 + \theta)\frac{(2C_2(t)/(n-t))}{1 + C_2(t)/(n-t)}$$

with $\theta > 0$.

To obtain the set of differential equations associated to the process, as discussed in [Achlioptas, 2001], we consider the scaled version of the process, x , $c_2(x)$, $c_3(x)$, $f(x)$ obtained by dividing every parameter t , $C_2(t)$, $C_3(t)$, $F(t)$ by n . We obtain the following differential equations.

$$\frac{dc_3}{dx} = -\frac{3c_3(x)}{1-x}$$

$$\frac{dc_2}{dx} = -\frac{2c_2(x)}{1-x} + p(x, c_2(x))\frac{3c_3(x)}{2(1-x)} + (1-p(x, c_2(x)))\frac{3c_3(x)}{1-x}$$

$$\frac{df}{dx} = p(x, c_2(x))$$

with $p(x, c_2(x)) = (1 + \theta) \frac{2c_2(x)/(1-x)}{1+c_2(x)/(1-x)}$ and initial conditions $c_3(0) = r$, $c_2(0) = 0$ and $f(0) = 0$. We solve the system numerically using the utility `dsolve` of `mapple` (it is easy to get $c_3(x) = r(1-x)^3$ analytically) for $r = 0.9227$, $\epsilon = 10^{-2}$ and $\theta = 10^{-5}$.

Now we are almost done. First, we can see that for every $1 \leq t \leq t^*$ w.h.p. $S_0(t) = \emptyset$ by testing numerically that $p(x, c_2(x)) < 1 - 10^{-1}$ if $x \in [0, 1 - 10^{-2}]$ and appealing to the lazy-server lemma. Moreover, we get $f(1 - 10^{-2}) < 0.49978$ and, transforming this result back to the randomized space-state, we can infer that $f(t^*) < 0.49978n + o(n)$. Similarly, the number of remaining clauses at that iteration is $C_2(t^*) + C_3(t^*) = c_2(1 - 10^{-2}) + c_3(1 - 10^{-2}) + o(n) < 0.00021n + o(n)$. It is easy to verify that the ratio of clauses to variables at iteration t^* is smaller than 1, and we can apply again the argument used to obtain the first naive lower bound of $1/2$ to guarantee that there exists a covering partial assignment with size $< 0.00021n$. Thus, by adding the previous quantities, we get $0.49999n < n/2$, which means that the algorithm succeeds.

4 Experimental Results for #3SAT($\geq 2^{n/2}$)

Preliminary experiments were run for random 3CNF-formulas with 4, 8, 16 and 32 variables on a SUN Ultra 5 workstation. For each space we generated 1200 random 3CNF-formulas with sizes ranging from 1 to 160 clauses in length. Each clause was generated by randomly selecting 3 variables without replacement and then negating each of them with probability of $1/2$.

Our goal was to test the formulas for being “yes” instances of #3SAT($\geq 2^{n/2}$), i.e., for having at least as many satisfying assignments as the square-root of the total number of truth assignments. For this, we implemented a threshold DPLL algorithm by modifying the basic Counting Davis-Putnam algorithm in [Birbaum and Lozinskii, 1999] to include tracking of lower and upper bounds on the count and early termination if the threshold is violated by the upper bound or satisfied by the lower bound.

The results are depicted in Figures 1 and 2. In both figures the horizontal axis is the ratio of the number of clauses to number of variables in the space. The ranges of formula sizes represented in the graphs are 1 to 20, 1 to 40, 1 to 80, and 1 to 160 for the 4, 8, 16 and 32 variable spaces respectively.

The phase transition graphs show for each test point the fraction of 1200 newly generated random formulas that had a number of satisfying truth assignments greater than or equal to the square-root of the total number of truth assignments. They strongly suggest that 2.5 is a critical ratio around which a phase transition occurs. The performance graphs show the average number of recursive calls required to test each formula and they exhibit a peak around the same ratio. In the test runs a range of 1 to 160 clauses was used for each space and the run-times on the SUN Ultra 5 were approximately 10, 15 and 35 minutes for the 4, 8 and 16 variable cases, and 7 hours for the 32 variable case.

After a larger set of experiments is carried out, we plan to apply finite-size scaling to further analyze the phase transition phenomenon exhibited by #3SAT($\geq 2^{n/2}$).

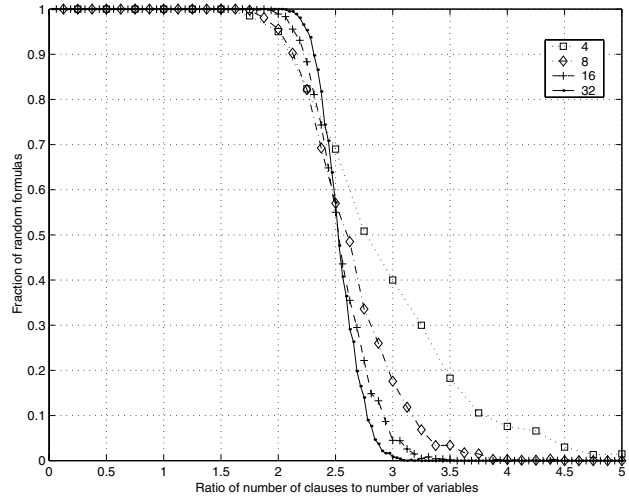


Figure 1: Phase Transition Graphs

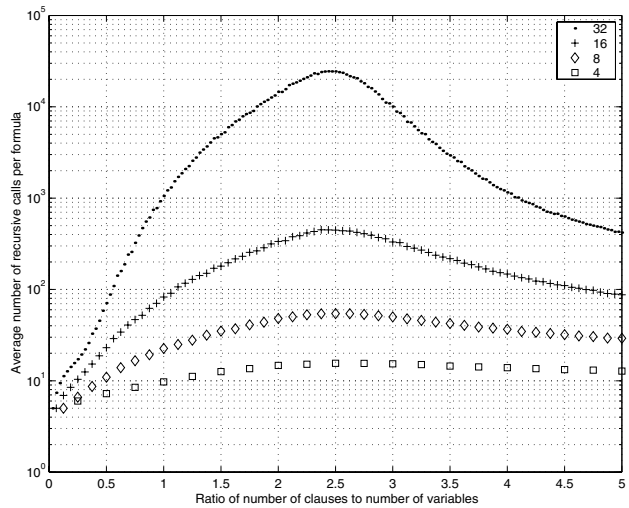


Figure 2: Performance Graphs

We conclude by pointing out that in Section 3.1.4 of [Littman *et al.*, 2001] it was suggested that for every $0 \leq \alpha \leq 1$, the critical ratio of #3SAT($\geq 2^{\alpha n}$) is given approximately by the formula $4.2(1 - \alpha)$. By taking $\alpha = 1/2$, this formula suggests that the critical ratio $r_{3,2}$ of #3SAT($\geq 2^{n/2}$) should be approximately 2.1, which is at odds with our experimental finding of 2.5 as the approximate value of $r_{3,2}$. We stand behind our experimental results; actually, we believe that this discrepancy is not caused by any significant difference in the outcome between the experiments carried out by [Littman *et al.*, 2001] and ours, but rather is due to the way in which the above formula was extrapolated from the experiments in [Littman *et al.*, 2001]. Specifically, in [Littman *et al.*, 2001] experiments were carried out by varying α and the ratio r of clauses to variables, but keeping the number of variables to a fixed value $n = 30$. The above formula $4.2(1 - \alpha)$ was then derived by visual inspection of the resulting sur-

face. We believe that, instead, the value of the critical ratio should be estimated by the crossover points of the curves obtained from experiments for different values of the number n of variables. In any case, we see no theoretical argument or experimental evidence that a linear relationship between the critical ratio and α should hold.

Acknowledgments: We are grateful to Dimitris Achlioptas for stimulating discussions and pointers to his work on lower bounds for 3SAT via differential equations. We also wish to thank Peter Young for generously sharing with us his expertise on phase transition phenomena, and Moshe Y. Vardi for listening to an informal presentation of the results reported here and offering valuable suggestions on an earlier draft of this paper.

References

- [Achlioptas and Sorkin, 2000] D. Achlioptas and G.B. Sorkin. Optimal myopic algorithms for random 3-SAT. In *41st Annual Symposium on Foundations of Computer Science*, pages 590–600, 2000.
- [Achlioptas, 2000] D. Achlioptas. Setting two variables at a time yields a new lower bound for random 3-SAT. In *32nd Annual ACM Symposium on Theory of Computing*, pages 28–37, 2000.
- [Achlioptas, 2001] D. Achlioptas. Lower bounds for random 3-SAT via differential equations. *Theoretical Computer Science*, 2001. To appear.
- [Angluin, 1980] D. Angluin. On counting problems and the polynomial-time hierarchy. *Theoretical Computer Science*, 12:161–173, 1980.
- [Bayardo and Pehoushek, 2000] R.J. Bayardo and J.D. Pehoushek. Counting models using connected components. In *7th Nat'l Conf. on Artificial Intelligence*, 2000.
- [Birnbaum and Lozinskii, 1999] E. Birnbaum and E.L. Lozinskii. The good old Davis-Putnam procedure helps counting models. *Journal of Artificial Intelligence Research*, 10:457–477, 1999.
- [Bylander, 1994] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:161–204, 1994.
- [Bylander, 1996] T. Bylander. A probabilistic analysis of propositional STRIPS planning. *Artificial Intelligence*, 81:241–271, 1996.
- [Cadoli et al., 1997] M. Cadoli, A. Giovanardi, and M. Schaerf. Experimental analysis of the computational cost of evaluating quantified Boolean formulas. In *Proc. 5th Conference of the Italian Association for Artificial Intelligence*, volume 218 of *LNAI*, pages 207–218. Springer-Verlag, 1997.
- [Chao and Franco, 1986] M.-T. Chao and J. Franco. Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM J. Comput.*, 15(4):1106–1118, 1986.
- [Chvátal and Reed, 1992] V. Chvátal and B. Reed. Mick gets some (the odds are on his side). In *33th Annual Symposium on Foundations of Computer Science*, pages 620–627, 1992.
- [Fernandez de la Vega, 1992] W. Fernandez de la Vega. On random 2-SAT. Manuscript, 1992.
- [Gent and Walsh, 1999] I.P. Gent and T. Walsh. Beyond NP: the QSAT phase transition. In *Proc. 16th National Conference on Artificial Intelligence*, pages 653–658, 1999.
- [Gill, 1977] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6(4):675–695, 1977.
- [Goerdts, 1996] A. Goerdts. A threshold for unsatisfiability. *J. Comput. System Sci.*, 53(3):469–486, 1996.
- [Janson et al., 2000] S. Janson, Y.C. Stamatiou, and M. Vamvakari. Bounding the unsatisfiability threshold of random 3-SAT. *Random Structures and Algorithms*, 17(2):103–116, 2000.
- [Littman et al., 1998] M.L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Automated Reasoning*, 1998.
- [Littman et al., 2001] M.L. Littman, S.M. Majercik, and T. Pitassi. Stochastic Boolean satisfiability. *Journal of Automated Reasoning*, 2001. To appear.
- [Littman, 1999] M. Littman. Initial experiments in stochastic satisfiability. In *Proc. of the 16th National Conference on Artificial Intelligence*, pages 667–672, 1999.
- [Orponen, 1990] P. Orponen. Dempster's rule of combination is #P-complete. *Artificial Intelligence*, 44:245–253, 1990.
- [Papadimitriou, 1994] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
- [Roth, 1996] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.
- [Selman et al., 1996] B. Selman, D. G. Mitchell, and H. J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81(1-2):17–29, 1996.
- [Simon, 1975] J. Simon. *On some central problems in computational complexity*. PhD thesis, Cornell University, Computer Science Department, 1975.
- [Toda, 1989] S. Toda. On the computational power of PP and $\oplus P$. In *Proceedings 30th IEEE Symposium on Foundations of Computer Science (FOCS'89)*, Research Triangle Park (North Carolina, USA), pages 514–519, 1989.
- [Valiant, 1979a] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.
- [Valiant, 1979b] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
- [Wormald, 1995] N. C. Wormald. Differential equations for random processes and random graphs. *Ann. Appl. Probab.*, 5(4):1217–1235, 1995.

KNOWLEDGE REPRESENTATION AND REASONING

DESCRIPTION LOGICS AND
CONCEPTUAL GRAPHS

Decision Procedures for Expressive Description Logics with Intersection, Composition, Converse of Roles and Role Identity*

Fabio Massacci

Dip. di Ingegneria dell'Informazione – Università di Siena
via Roma 56 – 53100 Siena – Italy
massacci@dii.unisi.it

Abstract

In the quest for expressive description logics for real-world applications, a powerful combination of constructs has so far eluded practical decision procedures: intersection and composition of roles.

We propose tableau-based decision procedures for the satisfiability of logics extending \mathcal{ALC} with the intersection \sqcap , composition \circ , union \sqcup , converse \cdot^{-} of roles and role identity $id(\cdot)$. We show that

1. the satisfiability of $\mathcal{ALC}(\sqcap, \circ, \sqcup)$, for which a 2-EXPTIME upper bound was given by tree-automata techniques, is PSPACE-complete;
2. the satisfiability of $\mathcal{ALC}(\sqcap, \circ, \sqcup, \cdot^{-}, id(\cdot))$, an open problem so far, is in NEXPTIME.

1 Introduction and Motivations

Description Logics (DLs) are a popular knowledge representation formalism based on *concepts* and *roles*, where concepts model classes of individuals, and roles model relationships between individuals [Baader *et al.*, 2001, Chap. 1-2].

In the last years, the investigation on DLs has been driven by the modeling needs of applications ranging from semi-structured data to planning [Baader *et al.*, 2001, Part III], which have stimulated the usage of expressive constructs. For instance, to model semi-structured data one needs to represent arbitrary relations (graphs with labeled edges), and use constructs for stating irreflexivity of relations, their transitive closure, or their intersection. For reasoning about actions, one may want to express the fact that two long sequences of actions (roles) must be completed in parallel.

These applications have called for decision procedures to make reasoning services up to the modeling duties (see [Baader *et al.*, 2001, Chap. 2] or Sec. 6). Yet, a powerful combination of constructs has so far eluded this quest: composition and intersection of roles. There is only an involved automata-based algorithm by Danecki [1984] giving a 2-EXPTIME upper bound.

*This work was done while the author was on leave at IRIT - Toulouse partly supported by CNR grant 203-7-27. I am greatly indebted to P. Balbiani for directing my attention to logics with intersection and for countless suggestions and discussions. Discussions with F. M. Donini, I. Horrocks and U. Sattler were helpful.

Decision procedures are hard to find because DLs with intersection and composition (even without converse and role identity¹) haven't the tree model property. With composition and intersection, we can write concepts whose models are directed acyclic graphs. Adding role identity, we can force a relation to be well-founded or a model to be a cyclic graph.

For instance, suppose we want to model the tangled web of corporate ownerships by using the roles *owns*, *app-board*, *app-CEO* (denoting that a corporation owns another company, appoints the company's board of directors, or its CEO). We can represent the corporations having a doubly indirectly controlled subsidiary with the concept

$$\text{corp} \sqcap \exists (\text{app-board} \circ \text{owns}) \sqcap (\text{owns} \circ \text{app-CEO}). \text{corp}$$

If regulators forbid corporations to be owners of themselves, we can forbid it too, without ad-hoc well-founded constructs:

$$\forall \text{owns} \sqcap id(\text{corp}). \perp$$

We can also model, self-owned “Chinese-box” corporations:

$$\exists (\text{owns} \circ \text{has-shares} \circ \text{app-board}) \sqcap id(\text{corp}). \top$$

These models are not representable with the “classical” expressive DLs such as \mathcal{ALC}_{reg} or $D\mathcal{LR}$.

Here, we consider the DL $\mathcal{ALC}(\sqcap, \circ, \sqcup, \cdot^{-}, id(\cdot))$ which, in addition to the constructs of \mathcal{ALC} , provides *intersection*, *composition*, *union*, *converse of roles* and *role identity*. $\mathcal{ALC}(\sqcap, \circ, \sqcup)$ is the fragment without inverse and identity.

These logics are siblings of three extensions of \mathcal{ALC} : \mathcal{ALC}_{reg} with composition and transitive/reflexive closure of roles but without intersection [Baader *et al.*, 2001, Chap.2]; \mathcal{ALB} with intersection, union and negation [Lutz and Sattler, 2000], and possibly converse [Hustadt and Schmidt, 2001]; $D\mathcal{LR}$ [Calvanese *et al.*, 1998] with intersection and role difference but only on atomic roles and with additional limitations. In the correspondence with PDL by Schild [1991], $\mathcal{ALC}(\sqcap, \circ, \sqcup)$ corresponds to the the *-free and test-free fragment of IPDL and $\mathcal{ALC}(\sqcap, \circ, \sqcup, \cdot^{-}, id(\cdot))$ is the *-free fragment of Converse-IPDL [Harel, 1984].

In the next sections we recall some preliminaries. Then we present our tableau calculus (Sec.3), transform it into algorithms (Sec.4), sketch the complexity results (Sec.5), and discuss related works (Sec.6).

¹Role identity is the construct which, given a concept C , allows one to build a role connecting each instance of C to itself.

2 Preliminaries

Let A and P denote atomic concepts and atomic roles respectively. Concepts C, D and roles R, S are formed as follows:

$$\begin{aligned} C, D &::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C \\ R, S &::= P \mid R \sqcup S \mid R \circ S \mid R \sqcap S \mid id(C) \mid R^- \end{aligned}$$

A *TBox* \mathcal{T} is a finite set of *inclusions* $C \sqsubseteq D$. In the sequel we focus on the satisfiability of concepts wrt empty TBoxes. However, we will build non-empty TBoxes of inclusions with a special form during the proof search itself.

An *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of \mathcal{I} — whose members are called *elements* — and a function $\cdot^{\mathcal{I}}$, the *interpretation function* of \mathcal{I} , that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We refer to Borgida [1996] or Baader et al. [2001, Chap. 2] for details.

An interpretation \mathcal{I} *satisfies a concept* C if there exists an element $d \in \Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$, i.e. if $C^{\mathcal{I}} \neq \emptyset$.

3 A Tableau Calculus

Taming intersection and composition requires the novel combination of a number of intuitions from the literature and some new features. They are highlighted here, for the reader familiar with the literature.

- We use the *graph-based representation of rules* for tableau-structures originally due to Kripke and used by Castillo et al. [1997] and Horrocks and Sattler [1999].
- We borrow the idea by Danecki [1984] of *pebble games* to mark elements linked by both R and S roles so that the *labeling an element with both pebbles marks the intersection*, triggering the insertion of suitable concepts;
- We *internalize pebbles in the calculus by introducing new propositional constants* as done by De Giacomo and Massacci [1997] for eventualities in CPDL.
- We use *skolemization for generating “new” nodes and atomic concepts* in the proof search, exploiting the results from DL translations into first-order logics [Hustadt and Schmidt, 2001].
- We *exploit some semantical properties of intersection* by Balbiani and Vakarelov [2001] to show we are sound.
- We use the idea of *on-the-fly modification of TBoxes* proposed by Massacci [1998] for the universal modality.
- We borrow the idea of *lazy unfolding of axioms* proposed by Horrocks and Tobies [2000] for plain \mathcal{ALC} .
- We use a *new role name to denote equivalence of individuals* but only for the proof search.

We use *tableau-structures* i.e. labeled graphs:

$$\mathcal{S} = \langle \mathcal{N}, \mathcal{E}, \mathcal{C}(\cdot), \mathcal{R}(\cdot, \cdot) \rangle$$

where \mathcal{N} is a finite nonempty set of nodes denoted by x, y, z possibly with indices; $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of oriented edges; $\mathcal{C}(\cdot)$ maps nodes to sets of concepts, and $\mathcal{R}(\cdot, \cdot)$ maps edges to sets of roles. We also use an initially empty TBox \mathcal{T} . Inclusions of the form $A_{x,R} \sqcap A_{x,S} \sqsubseteq C$ are added on-the-fly.

Axiom: If $x:A_1 \in \mathcal{S}$, $x:A_2 \in \mathcal{S}$, and $A_1 \sqcap A_2 \sqsubseteq C \in \mathcal{T}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:C\}$.

Conjunction: If $x:C \sqcap D \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:C, x:D\}$.

Disjunction: If $x:C \sqcup D \in \mathcal{S}$ then non-deterministically either $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:C\}$ or $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:D\}$.

Universal atomic roles: If $x:\forall P.C \in \mathcal{S}$, $\langle x, y \rangle : P \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{y:C\}$.

Universal role concatenation: If $x:\forall R \circ S.C \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:\forall R.\forall S.C\}$.

Universal role intersection: If $x : \forall R \sqcap S.C \in \mathcal{S}$ then let $A_{x,R}$ and $A_{x,S}$ be new atomic concepts and $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:\forall R.A_{x,R}, x:\forall S.A_{x,S}\}$ and $\mathcal{T} \Rightarrow \mathcal{T} \cup \{A_{x,R} \sqcap A_{x,S} \sqsubseteq C\}$.

Universal role union: If $x : \forall R \sqcup S.C \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:\forall R.C, x:\forall S.C\}$.

Existential restriction: If $x:\exists R.C \in \mathcal{S}$ then let y be a new node and $\mathcal{S} \Rightarrow \mathcal{S} \cup \{\langle x, y \rangle : R, y:C\}$.

Role concatenation: If $\langle x, y \rangle : R \circ S \in \mathcal{S}$ then let z be a new node and $\mathcal{S} \Rightarrow \mathcal{S} \cup \{\langle x, z \rangle : R, \langle z, y \rangle : S\}$.

Role intersection: If $\langle x, y \rangle : R \sqcap S \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{\langle x, y \rangle : R, \langle x, y \rangle : S\}$.

Role union: If $\langle x, y \rangle : R \sqcup S \in \mathcal{S}$ then non-deterministically either $\mathcal{S} \Rightarrow \mathcal{S} \cup \{\langle x, y \rangle : R\}$ or $\mathcal{S} \Rightarrow \mathcal{S} \cup \{\langle x, y \rangle : S\}$.

Figure 1: The reduction rules for $\mathcal{ALC}(\sqcap, \circ, \sqcup)$.

We start with a structure with a single node labeled with an input concept C , and apply the tableau rules to build a reduced structures without contradictions that can be used as a model for C . If none can be build, C is unsatisfiable.

The *reduction rules* in Fig. 1 are applicable to a tableau-structure \mathcal{S} and a set of inclusions \mathcal{T} for $\mathcal{ALC}(\sqcap, \circ, \sqcup)$. W.l.o.g. we assume that negation and converse are pushed down to atomic concepts and roles. We also abuse the DL syntax for tableau systems [Buchheit et al., 1993]: we write $x : C \in \mathcal{S}$ to indicate that $C \in \mathcal{C}(x)$, and $\langle x, y \rangle : R \in \mathcal{S}$ when $R \in \mathcal{R}(x, y)$. Thus, by $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:C\}$ we mean that we obtain a new structure \mathcal{S}' such that $\mathcal{N}' = \mathcal{N} \cup \{x\}$, $\mathcal{E}' = \mathcal{E}$, $\mathcal{C}'(y) = \mathcal{C}(y)$ if $y \neq x$ and $\mathcal{C}'(x) = \mathcal{C}(x) \cup \{C\}$, $\mathcal{R}'(\cdot, \cdot) = \mathcal{R}(\cdot, \cdot)$. Similarly for roles.

Most rules are close to other tableau approaches [Buchheit et al., 1993; Castilho et al., 1997; De Giacomo and Massacci, 1997; Horrocks and Sattler, 1999]. A difference is that they only label edges with atomic roles whereas we also use complex roles. So, we have rules for reducing these roles.

The role of *pebbles* is played by the new atoms introduced by the universal role intersection rule. The idea is that since $A_{x,R}$ is new, it can label a node y only by reducing $\forall R.A_{x,R}$. thus y must be an R -successors of x . If both $A_{x,R}$ and $A_{x,S}$ label node y , then y is linked to x by an $R \sqcap S$ role. Then we can add C to y .

Since we cannot forecast where $A_{x,R}$ and $A_{x,S}$ will appear, we must potentially check every node. To internalize this check into the calculus, we use the idea of *on-the-fly modification of TBoxes* by [Massacci, 1998]. With the intersection

rule we “update” \mathcal{T} with the inclusion $A_{x,R} \sqcap A_{x,S} \sqsubseteq C$ and then we use the Axiom rule to add C to nodes on demand.

Notice that we apply the Axiom rule only when both $A_{x,R}$ and $A_{x,S}$ are present in the same node. This technique, borrowed from the lazy unfolding of Horrocks and Tobies [2000], works because our inclusions are acyclic.

Notice also that the new concept $A_{x,R}$ introduced by the reduction of $x : \forall R \sqcap S.C$ must depend on the node where the concept is located. Following Danecki [1984, Pag. 44], one may think that it suffices to make them dependent only on the subformula (e.g. $A_{R,\forall R \sqcap S.C}$), possibly distinguishing between occurrences. This would be unsound. The concept

$$(\forall R.\forall R \sqcap S.C) \sqcap (\exists (R \circ R) \sqcap (R \circ S).\neg C)$$

is satisfiable but introducing the same “new” concepts at different nodes for the reduction of the only occurrence of $\forall R \sqcap S.C$ would result in a “clash”. By applying the rules of Fig. 1, one can see that the pebbles $A_{R,\forall R \sqcap S.C}$ and $A_{S,\forall R \sqcap S.C}$ would propagate along the “wrong” path.

The generation of new nodes and new atomic concept symbols may clearly lead to redundant rule applications or even to a non-terminating process. However, we do not need to generate really “new” nodes or atomic concepts. We can use the *equivalent of skolemization* (or Hilbert’s ϵ -terms) introduced by Ohlbach and later refined by Hustadt and Schmidt [2001] for the translations of DLs into first-order logic.

The generation of skolem functions is standard and we assume that for “new” concepts we use the function $\text{GENCONCEPT}(x, R)$, and for “new” nodes we use $\text{GENNODECONCEPT}(x, C)$ and $\text{GENNODEROLE}(x, y, R)$. So, when reducing $x : \exists R.C$, we call $y = \text{GENNODECONCEPT}(x, \exists R.C)$ and add $y : C$ and $\langle x, y \rangle : R$ to the structure.

Definition 1 A tableau structure \mathcal{S} and a set of inclusions \mathcal{T} are reduced for a rule R if the application of R maps \mathcal{S} and \mathcal{T} into themselves. A structure and a set of inclusions are reduced if they are reduced for all rules.

Definition 2 A tableau-structure contains a clash if there is a node x and a concept C such that $\{C, \neg C\} \subseteq \mathcal{C}(x)$.

Theorem 1 A concept C of $\mathcal{ALC}(\sqcap, \sqcup, \sqcup)$ is satisfiable iff there is a non-deterministic application of the rules in Fig. 1 to an initially empty TBox and a structure with only one node labeled with C leading to a reduced and clash-free structure.

For every feature, the proof cleverly combines the techniques from the cited works in the literature. The twist is the soundness of the universal role intersection rule. To this extent, we set $A_{x,R}^{\mathcal{I}} = \{y \mid \langle x, y \rangle \in R^{\mathcal{I}}\}$ and similarly for S . In words, the newly introduced concepts are fulfilled by the appropriate individuals R -reachable, or S -reachable, from x . Due to lack of space, details are left to the full paper.

Next, we define the tableau rules for the full language. Beside converse, the tricky bit is the presence of the intersection between role identity and complex roles: $id(C) \sqcap R$ is a self-loop describing individuals in the class C that are in relation R with themselves (e.g. the self-owned companies). To this extent we employ an atomic role symbol (not occurring in the input concept) for equality of nodes \approx .

Universal converse of roles: If $y : \forall R^-.C \in \mathcal{S}$, $\langle x, y \rangle : R \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:C\}$.

Universal role identity: If $x : \forall id(D).C \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x:\neg D \sqcup C\}$.

Role converse: If $\langle x, y \rangle : R^- \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{\langle y, x \rangle : R\}$.

Role identity: If $\langle x, y \rangle : id(C) \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x \approx y, x:C\}$

Role identity distribution: If $x \approx y \in \mathcal{S}$ and $x : C \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{y:C\}$.

Role identity symmetry: If $x \approx y \in \mathcal{S}$ then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{y \approx x\}$.

Role identity transitivity: If $x \approx y \in \mathcal{S}$ and $y \approx z \in \mathcal{S}$ for then $\mathcal{S} \Rightarrow \mathcal{S} \cup \{x \approx z\}$.

Figure 2: The additional reduction rules for \cdot^- and $id(\cdot)$

Algorithm WORLDS;

input node \underline{x}_0 ; set of concepts \mathcal{C} ; set of inclusions \mathcal{T} ;

output sat if \mathcal{C} is satisfiable, *unsat* otherwise;

variables node x ; labeled graph \mathcal{S} ;

begin

$\mathcal{N} = \{\underline{x}_0\}$; $\mathcal{E} = \emptyset$; $\mathcal{C}(\cdot) = \{\langle \underline{x}_0, C \rangle\}$; $\mathcal{R}(\cdot, \cdot) = \emptyset$;

if CLASH(\mathcal{S}) **then return**(*unsat*);

while $x = \text{CHOOSENODE}(\mathcal{S}, \mathcal{T}) \neq \text{none}$ **do**

$R = \text{CHOOSERULE}(x, \mathcal{S}, \mathcal{T})$;

 “Apply R to $x, \mathcal{S}, \mathcal{T}$ updating \mathcal{S} and \mathcal{T} ”

if CLASH(\mathcal{S}) **then return**(*unsat*);

forall $x \in \mathcal{N} \setminus \{\underline{x}_0\}$ **do**

if WORLDS($x, \mathcal{C}(x), \mathcal{T}$) == *unsat*

then return(*unsat*);

return sat

end;

Figure 3: The algorithm for $\mathcal{ALC}(\sqcap, \sqcup, \sqcup)$

The reduction rules in Figure 1 and the additional rules in Figure 2 are applicable to a tableau-structure \mathcal{S} and a set of inclusions \mathcal{T} for the logic $\mathcal{ALC}(\sqcap, \sqcup, \sqcup, \cdot^-, id(\cdot))$.

The rules for $\cdot \approx \cdot$ might be eliminated if the rule for $id(\cdot)$ collapses nodes. However, this would complicate the algorithms, as we would need a unique way for collapsing nodes.

Theorem 2 A concept C of $\mathcal{ALC}(\sqcap, \sqcup, \sqcup, \cdot^-, id(\cdot))$ is satisfiable iff there is a non-det. application of the rules in Fig. 1, 2 to an initially empty TBox and a structure with only one node labeled with C yielding a reduced and clash-free structure.

4 From Calculi to Algorithms

The PSPACE-algorithm for $\mathcal{ALC}(\sqcap, \sqcup, \sqcup)$ is called WORLDS after Ladner’s WORLD for modal logic K. It is plural because each call works on a fragment of a Kripke model and not just on a set of concepts true for an individual. It is in Fig. 3.

To check the satisfiability of a concept C one calls $\text{WORLDS}(0, \{C\}, \emptyset)$ and applies the rules from Fig. 1.

In the algorithms, we use the symbol “=” for assignment, and “==” for equality testing. We assume all functionalities for working with labeled direct graphs, an auxiliary function CLASH(\mathcal{S}) which detects clashes, and auxiliary procedures for selecting objects: CHOOSENODE selects a node to be reduced; CHOOSERULE selects an applicable rule for a node.

Algorithm CWORLDS;
input node \underline{x}_0 ; set of concepts \mathcal{C} ;
output set of concepts $\mathcal{D} \supseteq \mathcal{C}$ if \mathcal{C} is satisfiable, *unsat* otherwise;
variables node x ; \mathcal{S} labeled graph; \mathcal{D} set of concepts;
global variables set of inclusions \mathcal{T} ;
begin
 $\mathcal{N} = \{\underline{x}_0\}$; $\mathcal{E} = \emptyset$; $\mathcal{C}(\cdot) = \{\langle \underline{x}_0, \mathcal{C} \rangle\}$; $\mathcal{R}(\cdot, \cdot) = \emptyset$;
if CLASH(\mathcal{S}) **then return** *unsat*;
start: **while** $x = \text{CHOOSE NODE}(\mathcal{S}, \mathcal{T}) \neq \text{none}$ **do**
| $R = \text{CHOOSE RULE}(x, \mathcal{S}, \mathcal{T})$
| “Apply R to x , \mathcal{S} and \mathcal{T} updating \mathcal{S} and \mathcal{T} ”
| **if** CLASH(\mathcal{S}) **then return** *unsat*;
forall $x \in \mathcal{N} \setminus \{\underline{x}_0\}$ **do**
| $\mathcal{D} = \text{CWORLDS}(x, \mathcal{C}(x))$;
| **if** $\mathcal{D} = \text{unsat}$ **then return** *unsat*;
| **else if** $\mathcal{C}(x) \neq \mathcal{D}$;
| **then** $\mathcal{C}(x) = \mathcal{D}$;
| **goto** **start**;
return($\mathcal{C}(\underline{x}_0)$)
end;

Figure 4: The CWORLDS algorithm

These procedures must work in time polynomial in the size of the input and respect the following constraints:

Search Criteria 1 *A rule is not selected if the structure is already reduced for it. A node is not selected if the structure is already reduced for all rules applicable to the node.*

Search Criteria 2 *The Existential restriction rule is selected only for reduction of concepts labeling the initial node \underline{x}_0 .*

If the procedures cannot select anything respecting the above constraints they return the value *none*.

The tricky bit is showing that the algorithm is in PSPACE (Sec. 5). We just highlight the differences from trace-based methods used for \mathcal{ALC} by Ladner [1977], Schmidt-Schauss and Smolka [1991] or Tobies [2001].

The classical tableau-based algorithm for \mathcal{ALC} would have been identical to ours except for the last **for** cycle:

```
for all  $\exists R.C \in \mathcal{C}(\underline{x}_0)$  do
| if  $\text{WORLD}(\{C\} \cup \{D \mid \forall R.D \in \mathcal{C}(\underline{x}_0)\}) = \text{unsat}$ 
| then return(unsat)
```

In words, for \mathcal{ALC} we explore the one-step R -successors of \underline{x}_0 by recursively calling the procedure. So, we only examine one node (\underline{x}_0) at every call.

The WORLDS algorithm examines the initial node \underline{x}_0 and reduces all existential concepts labeling \underline{x}_0 , i.e. all one step R -successors of \underline{x}_0 . Thus, it builds a fragment of the Kripke structure. However, the reduction of existential-concepts labeling successors nodes is deferred to recursive calls. The rules for intersection and composition may transform a “one-step” R -successors into “many-steps” atomic P -successors. Still, the number of intermediate P -steps is linearly bounded by the size of the input concept.

The full algorithm is called CWORLDS because it returns a set of concepts. It is shown on Figure 4. The rules that can be applied to the algorithm are those from Fig. 1,2. It must respect an additional search constraint:

Search Criteria 3 *If $x_1 \approx x_2$, $x_2 \approx x_3, \dots, x_{n-1} \approx x_n$ are present in the structure then only one node $x_i \in \{x_1, \dots, x_n\}$ can be selected for the recursive call of CWORLDS in the forall cycle. If \underline{x}_0 is among them, none can be selected.*

We must return a set of concepts rather than just *sat* to cope with converse and role identity. The idea is borrowed from modal logics with symmetric relations [Massacci, 2000] and DLs with converse [Tobies, 2001].

For example, the concept $\exists R \circ id(\exists id(\forall R.C). \top) \circ R. \neg C$ is unsatisfiable. If we used algorithm WORLDS with the full set of rules it would return *sat*. The explanation is that we would introduce a node y labeled by $\exists id(\forall R.C). \top$ and would evaluate the existential concept in the next recursive call. In the recursive call we would add $\forall R.C$ to the initial node (which would correspond to y) but we could not use this information to derive a clash in the initial call.

The use of skolemization, rather than truly fresh names, is necessary to guarantee that CWORLDS is both correct and terminating. For instance, if fresh names are used in the recursive calls following a restart, the concept $\exists R. \exists S. \forall S. \neg \square T.C$ would make CWORLDS non-terminating.

5 Complexity Analysis

We denote by n the size of the input concept C which must be proved (un)satisfiable, measured as the number of symbols.

We first prove that WORLDS requires only polynomial space in n . The idea is that $\mathcal{ALC}(\square, \circ, \sqcup)$ lost the tree-model property but kept the cactus-model property.

Loosely speaking, our models can be arranged into the shape of a cactus, i.e. a tree in which the edge connecting two nodes of the tree is not a slim branch but a “fat”, cactus-like stem. This fat stem is made by other nodes and edges, but its size is polynomially bounded by n . Many stems sprout from each stem, as in a real cactus, but their number is still polynomially bounded. Since the height of the cactus is also polynomially bounded we are done.

Concepts labeling nodes can be seen as spikes. We must also prove that they aren’t exponentially many. For most DLs this is obvious: we only introduce subconcepts of the input concept. Here, we introduce new atomic concepts.

The next tree lemmata make these intuitions precise. At first, we say that the cactus height is polynomially bounded:

Lemma 1 *The call stack of WORLDS is at most $O(n)$ deep.*

Then the size of each cactus stem is bounded:

Lemma 2 *Each invocation of WORLDS generate a structure with at most $O(n)$ nodes.*

Finally, we say that the the number of spikes is bounded.

Lemma 3 *The number of new atoms occurring as (sub)concepts in the the labels of the nodes of an invocation of WORLDS is bounded by $O(n^3)$.*

To prove Lemma 1 we cannot use the standard proof that each interaction reduces the modal depth (nesting of universal and existential roles) of the set of concepts labeling a node because of the Axiom rule. We need a new notion of modal depth: we associate different depths to atomic concepts from the input concept and to “invented” atomic concepts.

- if A is from input concept then $d(A) = 0$
- if $A_{x,R}$ is an atomic concept introduced by the reduction of one or more $x : \forall R \sqcap S.C$ then $d(A_{x,R})$ is equal to $\max_{S,C} \{d(C) \mid x: \forall R \sqcap S.C\}$
- $d(\forall R.C) = |R| + d(C)$ where $|R|$ is the size of R , measured as the number of symbols.

Now by induction on the number of applied rules we can show that for all nodes $x \neq \underline{x}_0$ and all concepts $C \in \mathcal{C}(x)$, there is a concept $D \in \mathcal{C}(\underline{x}_0)$ such $d(D) > d(C)$. Since the depth of every concept, and hence the maximal depth of a set of concepts, is bounded by n we are done.

For the proof of Lemma 2, observe that new nodes in the structure \mathcal{S} can only be created by two rules: the rules reducing existential and the rules reducing composition of roles. The first rule can only be applied to \underline{x}_0 , thus bounding the nodes so introduced by $O(n)$. For the second rule, the number of nodes thus introduced is bounded by $O(n)$.

For Lemma 3, observe that two “new” concepts are introduced only when we reduce an universally quantified role intersection. The newly generated concepts depend on the same node and on the roles which are immediate subroles of an intersection of roles occurring in the input concept. Hence, by Lemma 2, for each invocation we can create at most $O(n \cdot n)$ different new concepts. By Lemma 1 the number of nested recursive calls of WORLDS is at most $O(n)$. When a leaf call is reached we generated at most $O(n^2 \cdot n)$ different concepts.

Theorem 3 WORLDS can be implemented using only polynomial space in n .

For the proof observe that the stack is at most $O(n)$ deep by Lemma 1 and for each call of WORLDS at depth i we have

- $O(n) + O(n)$ new nodes and at most $O(n^2)$ edges,
- $O(n^4)$ concepts for each node and each concepts taking at most $O(n)$ space,
- $O(n)$ roles labeling each edge and each role taking at most $O(n)$ space
- $O(i \cdot n^5)$ inclusions, which can be added by reducing a concept $x : \forall R \sqcap S.C$ where the number of different x s is bounded by Lemma 2, $R \sqcap S$ occurs in the input concept and C either occurs in the input concept or is one of $O(n^3)$ new concepts invented at any $j \leq i$ steps.

PSPACE-hardness follows from \mathcal{ALC} .

Corollary 4 The satisfiability of $\mathcal{ALC}(\sqcap, \exists, \sqcup)$ concepts is PSPACE-complete.

The NEXPTIME-upper bound of $\mathcal{ALC}(\sqcap, \exists, \sqcup, \cdot^-, id(\cdot))$ satisfiability, has a more involved proof. The analogous of Lemma 1 and Lemma 2 can be proved with a similar argument. The only twist is observing that, by Criteria 3, we only reduce the existential concepts in one of the node linked by the equality predicate. Thus, we never recursively call CWORLDS on the nodes “equal” to the root node \underline{x}_0 (as for them the induction would fail). The equivalent of Lemma 3 fails and we only have a global bound:

Lemma 4 The total number of new atoms occurring as (sub)concepts in the the labels of the nodes throughout the execution of WORLDS is bounded by $O(n^n) = 2^{O(n \log n)}$.

Using Lemma 4, we bound the number of concepts labeling a formula to $O(n) \cdot 2^{O(n \log n)} = 2^{O(n \log n)}$ and the number of inclusions by doubling the multiplicative constant in the $O(n \log n)$ expression at the exponent.

Theorem 5 CWORLDS terminates after $2^{O(n^2 \log n)}$ time.

For the proof observe that CWORLDS can be “restarted” at most finitely many times. More precisely, (i) the set of concepts returned by CWORLDS is larger or equal to the initial set upon which CWORLDS is called (in symbols $\mathcal{C}(\underline{x}_0) \supseteq C$), and (ii) the total number of concepts is bounded by $2^{O(n \log n)}$. Then the total number of restarts for a structure in a single invocation of CWORLDS is bounded by $O(n) \cdot 2^{O(2n \log n)}$. The bound on the stack does the rest.

Corollary 6 The satisfiability of $\mathcal{ALC}(\sqcap, \exists, \sqcup, \cdot^-, id(\cdot))$ concepts is in NEXPTIME.

An intriguing question is why the standard technique for giving PSPACE-bounds for DLs with converse [Tobies, 2001] or symmetric modal logics [Massacci, 2000] fails here. According this technique, we take the “converse-free” algorithm and add some “restarts” (this is what CWORLDS does). For the bound, we observe that (i) restarts add a bounded number of new concepts (indeed subconcepts of the input concept) and (ii) the time wasted by restarts doesn’t matter.

The argument fails here because converse or role identity and intersection plus composition can force CWORLDS to label a node with exponentially many new atomic concepts.

To force this exponential generation of concepts, we start by constructing a concept G_n whose model is a cyclic graph with edges labeled by two roles R and S . Looking only at role R , the graph is a binary tree with height n , with G_n labeling its root. Each S -edge connects the child node of an R -edge to its parent. The key feature of G_n is that we can start at the root, traverse forward a path of R -edges, reach a leaf and then continue to traverse S -edges forward to the root. Once there, we can take another R -path, and so on. The model of G_n has a path whose length is exponential in n . Pictorially, G_n can be seen as a daisy with exponentially many petals.

This construction is impossible in \mathcal{ALC} , \mathcal{ALC} with converse, or $\mathcal{ALC}(\sqcap, \exists, \sqcup)$ because there are always acyclic models. In \mathcal{ALC} we can keep on traversing edges (roles) on one path until we arrive at dead-end. Since the depth of the longest path is linear in the size of the concept to be verified we are done. In \mathcal{ALC} with converse, edges can be oriented forward or backward: we get a two-ways path but still it terminates into a dead-end. With intersection we must check that two paths meet at certain points but the idea is the same.

Still, CWORLDS only uses polynomial space for visiting G_n . The problems start when we add to G_n one or more concepts that ask to verify the intersection of two paths. Loosely speaking, one path reaches the top of a petal, the other goes back to center, then round another petal and back to the top of the first petal. The problem is that we don’t know a priori which petal we must visit among the exponentially many that are available. We can put one of these concepts in each leaf. Then, the number of pebbles that accumulate in the root, restart after restart, will be exponential in n .

6 Related Methods

A number of decision procedures and complexity results for logics extending \mathcal{ALC} (or multimodal K) can be found in the literature. Yet, none of them fully tackle our results.

With respect to complexity results based on encodings, De Giacomo and Lenzerini [1995] proved the EXPTIME-completeness of a DL with composition, converse and intersection restricted to atomic roles with additional limitations. Calvanese et al. [1998] proved the EXPTIME-completeness of \mathcal{DLR} , where intersection, union and difference of roles are allowed but composition is not permitted. Baader and Sattler [1999] proved the undecidability of a number of combination of expressive DLs with number restrictions and intersection.

With respect to decision procedures, Horrocks et al. [1999; 2000] tamed expressive DLs with converse and role-hierarchies where intersection between atomic roles can be simulated. They do not allow for composition of roles. Baader and Sattler [1999] have proposed an EXPTIME-calculi for \mathcal{ALC} with number restrictions with composition, intersection and union of role chains. However, concepts are restricted to plain \mathcal{ALC} . Lutz and Sattler [2000] give a decision procedure for \mathcal{ALB} (\mathcal{ALC} plus intersection, union, and negation of roles but without composition). Tobies [2001] gives a PSPACE-algorithm for DLs with number restrictions, converse and intersection of atomic relations.

In the realm of modal and dynamic logic, Danecki [1984] has shown, using automata-based techniques, that IPDL (PDL with Intersection) can be decided in 2-EXPTIME. IPDL strictly extends the logic $\mathcal{ALC}(\sqcap, \circ, \sqcup)$ allowing for the transitive and reflexive closure of roles and for role identity. However, Danecki claims that it is possible to use pebbles which depends only on the occurrence of the subformula. For our calculus, this is unsound and Danecki's involved construction may need to be checked against our counterexample. For Converse-PDL automata theoretic techniques have been proposed by Vardi and Wolper [1986] and a tableau calculus has been given by De Giacomo and Massacci [1997].

Hustadt and Schmidt [2001] have shown a decision procedure for boolean modal logic with converse. Their procedure is based on a clever translation into first-order logic of modal formulae followed by a decision procedure for the guarded fragment. They only prove decidability results. The possibility of a PSPACE-algorithm for the extension of \mathcal{ALC} with intersection, converse and union is just claimed.

Decision procedures for first order logic with two variables (see the survey by Grädel and Otto [1999]) can be used for DLs with intersection but without composition, via first-order translations. One could also use the decision procedures by Ganzinger and De Nivelle [1999] for the guarded fragment. Current methods based on translations into decidable fragments of first order logic cannot treat intersection and composition at the same time, as this requires to have at least three variables and does not allow for guards in predicate clauses.

References

[Baader and Sattler, 1999] F. Baader and U. Sattler. Expressive number restrictions in description logics. *JLC*, 9:319–350, 1999.
[Baader et al., 2001] F. Baader, D. McGuinness, D. Nardi, and P. Patel Schneider, editors. *The Description Logic Handbook:*

Theory, implementation and applications. Cambridge Univ. Press. 2001. To appear.
[Balbiani and Vakarelov, 2001] P. Balbiani and D. Vakarelov. Iteration-free PDL with intersection: a complete axiomatization. *Fundamenta Informaticae*, 2001. To appear.
[Borgida, 1996] A. Borgida. On the relative expressiveness of description logics and predicate logics. *AIJ*, 82:353–367, 1996.
[Buchheit et al., 1993] M. Buchheit, F. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *JAIR*, 1:109–138, 1993.
[Calvanese et al., 1998] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pp.149–158, 1998.
[Castilho et al., 1997] M. Castilho, L. Fariñas del Cerro, O. Gasquet, and A. Herzig. Modal tableaux with propagation rules and structural rules. *Fund. Inf.*, 32:281–297, 1997.
[Danecki, 1984] R. Danecki. Nondeterministic Propositional Dynamic Logic with Intersection is decidable. In *Proc. of the 5th Sym. on Comp. Theory*, LNCS 208, pp.34–53. Springer, 1984.
[De Giacomo and Lenzerini, 1995] G. De Giacomo and M. Lenzerini. What's in an aggregate: foundations for description logics with tuples and sets. In *Proc. of IJCAI'95*, pp.801–807, 1995.
[De Giacomo and Massacci, 1997] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 162:117–137, 2001. Short version in *Proc. of CADE-96*.
[Ganzinger and de Nivelle, 1999] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. of LICS'99*, pp.295–304, 1999.
[Grädel and Otto, 1999] E. Grädel and M. Otto. On logics with two variables. *TCS*, 224:73–113, 1999.
[Harel, 1984] D. Harel. Dynamic logic. In *Handbook of Philosophical Logic*, vol. II, pp.497–640. Reidel, 1984.
[Horrocks and Sattler, 1999] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *JLC*, 9:385–410, 1999.
[Horrocks and Tobies, 2000] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of KR 2000*, pp.285–296. 2000.
[Horrocks et al., 2000] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. *Logic J. of the IGPL*, 8:239–263, 2000.
[Hustadt and Schmidt, 2001] U. Hustadt and R. A. Schmidt. Using resolution for testing modal satisfiability and building models. *JAR*, 2001. To appear.
[Ladner, 1977] R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM JoC*, 6:467–480, 1977.
[Lutz and Sattler, 2000] C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logic. In *Proc. of AiML 2000*.
[Massacci, 1998] F. Massacci. Tableaux methods for formal verification in multi-agent distributed systems. *JLC*, 8:373–400, 1998.
[Massacci, 2000] F. Massacci. Single step tableaux for modal logics: methodology, computations, algorithms. *JAR*, 24:319–364, 2000.
[Schild, 1991] K. Schild. A correspondence theory for terminological logics. In *Proc. of IJCAI'91*, pp.466–471. 1991.
[Schmidt-Schauß and Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *AIJ*, 48:1–26, 1991.
[Tobies, 2001] S. Tobies. PSPACE reasoning for graded modal logic. *JLC*, 2001, To appear.
[Vardi and Wolper, 1986] M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *JCSS*, 32:183–221, 1986.

Ontology Reasoning in the $\mathcal{SHOQ}(\mathcal{D})$ Description Logic

Ian Horrocks

Department of Computer Science
University of Manchester, UK
horrocks@cs.man.ac.uk

Ulrike Sattler

LuFg Theoretical Computer Science
RWTH Aachen, Germany
sattler@informatik.rwth-aachen.de

Abstract

Ontologies are set to play a key rôle in the “Semantic Web” by providing a source of shared and precisely defined terms that can be used in descriptions of web resources. Reasoning over such descriptions will be essential if web resources are to be more accessible to automated processes. $\mathcal{SHOQ}(\mathcal{D})$ is an expressive description logic equipped with named individuals and concrete datatypes which has almost exactly the same expressive power as the latest web ontology languages (e.g., OIL and DAML). We present sound and complete reasoning services for this logic.

1 Introduction

The recent explosion of interest in the World Wide Web has also fuelled interest in ontologies.¹ This is due both to the use of ontologies in existing Web based applications and to their likely rôle in the future development of the Web [van Heijst *et al.*, 1997; McGuinness, 1998; Uschold and Grüninger, 1996]. In particular, it has been predicted that ontologies will play a pivotal rôle in the *Semantic Web*—the World Wide Web Consortium’s vision of a “second generation” Web in which Web resources will be more readily accessible to automated processes [Berners-Lee, 1999].

A key component of the Semantic Web will be the annotation of web resources with meta-data that describes their content, with ontologies providing a source of shared and precisely defined terms that can be used in such meta-data. This requirement has led to the extension of Web markup languages in order to facilitate content description and the development of Web based ontologies, e.g., XML Schema, RDF (Resource Description Framework), and RDF Schema [Decker *et al.*, 2000]. RDF Schema (RDFS) in particular is recognisable as an ontology/knowledge representation language: it talks about classes and properties (binary relations), range and domain constraints (on properties), and subclass and subproperty (subsumption) relations. However, RDFS is a very primitive language (the above is an almost

¹The word ontology has been used—some would say abused—in a wide range of contexts. In this paper it will be taken to mean a formally defined model of (part of) the domain of interest.

complete description of its functionality), and more expressive power would clearly be necessary/desirable in order to describe resources in sufficient detail. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes.

These considerations have led to the development of OIL [Fensel *et al.*, 2000] and DAML [Hendler and McGuinness, 2001], two ontology languages that extend RDFS with a much richer set of modelling primitives. Both languages have been designed in such a way that they can be mapped onto a very expressive description logic (DL).² This mapping provides them with a formal semantics, a clear understanding of the characteristics of various reasoning problems (e.g., subsumption/satisfiability), and the possibility of exploiting existing decision procedures. OIL, in particular, was designed so that reasoning services could be provided, via a mapping to the \mathcal{SHIQ} DL, by the FaCT system [Horrocks *et al.*, 1999; Horrocks, 2000].

Unfortunately, these mappings are currently incomplete in two important respects. Firstly, any practical ontology language will need to deal with *concrete datatypes* (numbers, strings, etc.) [Baader and Hanschke, 1991]. E.g., ontologies used in e-commerce may want to classify items according to weight, and to reason that an item weighing more than 50 kilogrammes is a kind of item that requires special shipping arrangements. OIL already supports the use of integers and strings in class descriptions, and it is anticipated that DAML+OIL, a new language developed from a merging of the DAML and OIL efforts, will support (most of) the datatypes defined or definable by XML Schema. However, the \mathcal{SHIQ} logic implemented in the FaCT system does not include any concrete datatypes, so there is no mechanism for reasoning with this part of the language.

Secondly, realistic ontologies typically contain references to named individuals within class descriptions. E.g., “Italians” might be described as persons who are citizens of “Italy”, where Italy is a named individual [Schaerf, 1994]. The required functionality can be partially simulated by treating such individuals as pairwise disjoint atomic classes (this is the approach taken in the existing OIL \rightarrow FaCT mapping), but this can result in incorrect inferences.

In this paper we will present a new DL that overcomes

²In fact they can be viewed as syntactic variants of such a logic.

both of the above deficiencies by taking the logic SHQ and extending it with individuals (\mathcal{O}) and concrete datatypes (\mathbf{D}) to give $SHOQ(\mathbf{D})$. The starting point for these extensions is SHQ rather than $SHIQ$ (i.e., without inverse roles), because reasoning with inverse roles is known to be difficult and/or highly intractable when combined with either concrete datatypes or named individuals: the concept satisfiability problem is known to be NExpTime hard even for the basic DL \mathcal{ALC} augmented with inverse roles and either concrete datatypes or named individuals [Lutz, 2000; Tobies, 2000]. This hardness result for concrete datatypes is not yet directly applicable to $SHOQ(\mathbf{D})$ as it depends on comparisons of concrete values (binary predicates), but the addition of such comparisons would be a natural future extension to $SHOQ(\mathbf{D})$. Moreover, the presence of nominals in any DL leads to the loss of the tree/forest model property, which becomes particularly problematical in the presence of inverse roles, number restrictions, and general axioms. As a result, to the best of our knowledge, there is no (practicable) decision procedure for $SHIQ$ with nominals or converse-DPDL with nominals, the latter being a close relative of $SHIQ$ from dynamic logics [Streets, 1982]. Finally, since individuals and concrete datatypes are much more widely used in ontologies than inverse roles [Corcho and Pérez, 2000], $SHOQ(\mathbf{D})$ is a very useful addition to our reasoning armoury.

2 Preliminaries

In this section, we will describe our choice of concrete datatypes and named individuals, and introduce the syntax and semantics of $SHOQ(\mathbf{D})$.

Concrete Datatypes Concrete datatypes are used to represent literal values such as numbers and strings. A type system typically defines a set of “primitive” datatypes, such as *string* or *integer*, and provides a mechanism for deriving new datatypes from existing ones. For example, in the XML schema type system the *nonNegativeInteger* datatype is derived from the *integer* datatype by constraining values of *nonNegativeInteger* to be greater than or equal to zero [Biron and Malhorta, 2000].

In order to represent concepts such as “persons whose age is at least 21”, we can extend our concept language with a set \mathbf{D} of concrete datatypes and concepts of the form $\exists R.d$ and $\forall R.d$, where $d \in \mathbf{D}$. To be more precise, we assume that we have a set of datatypes \mathbf{D} , and, with each $d \in \mathbf{D}$, a set $d^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$ is associated, where $\Delta_{\mathbf{D}}$ is the domain of all datatypes. We will assume that:

1. the domain of interpretation of all concrete datatypes $\Delta_{\mathbf{D}}$ (the *concrete domain*) is disjoint from the domain of interpretation of our concept language (the *abstract domain*), and
2. there exists a sound and complete decision procedure for the emptiness of an expression of the form $d_1^{\mathbf{D}} \cap \dots \cap d_n^{\mathbf{D}}$, where d_i is a (possibly negated) concrete datatype from \mathbf{D} (where $\neg d$ is interpreted as $\Delta_{\mathbf{D}} \setminus d^{\mathbf{D}}$).

We will say that a set of datatypes is *conforming* if it satisfies the above criteria.

The disjointness of the abstract and concrete domains is motivated by both philosophical and pragmatic considerations. On the one hand, concrete datatypes are considered to be already sufficiently structured by the type system, which may include a derivation mechanism and built-in ordering relations; therefore, we do not need the DL mechanism to form new datatypes as in [Baader and Hanschke, 1991]. On the other hand, it allows us to deal with an arbitrary conforming set of datatypes without compromising the compactness of our concept language or the soundness and completeness of our decision procedure.

This scheme can be trivially extended to include boolean combinations of datatypes and number restrictions qualified with data types, but to simplify the presentation we will only consider (possibly negated) atomic datatypes and exists/value restrictions. The type system can be as complex as that defined for XML schema, or as simple as the one defined in the OIL ontology language [Fensel *et al.*, 2000], where the only primitive datatypes are integer and string, and new types are derived by adding minimum and maximum value constraints. Using the OIL typesystem we could, for example, define the type (min 21) and use it in the concept $\text{Person} \sqcap \exists \text{age}.\text{(min 21)}$.

Named Individuals Allowing named individuals to occur in concepts provides additional expressive power that is useful in many applications; *nominals* (as such individuals can be called) are a prominent feature of hybrid logics [Blackburn and Seligman, 1998], and various extensions of modal and description logics with nominals have already been investigated (see, e.g., [Schaefer, 1994; De Giacomo, 1995; Areces *et al.*, 2000]). As we have seen, nominals occur naturally in ontologies as names for specific persons, companies, countries etcetera.

From a semantic point of view, it is important to distinguish between a nominal and an atomic concept/simple class, since the nominal stands for exactly one individual—in contrast to a concept, which is interpreted as some *set* of individuals. Modelling nominals as pairwise disjoint atomic concepts can lead to incorrect inferences, in particular with respect to implicit maximum cardinality constraints. For example, if Italy is modelled as an atomic concept, then it would not be possible to infer that persons who are citizens *only* of Italy cannot have dual-nationality (i.e., cannot be citizens of more than one country).

Finally, nominals can be viewed as a powerful generalisation of DL *Abox individuals* [Schaefer, 1994]: in an Abox we can assert that an individual is an instance of a concept or that a pair of individuals is an instance of a role (binary relation), but Abox individuals cannot be used *inside* concepts. For example, if Giuseppe and Italy are Abox individuals, we could assert that the pair (Giuseppe, Italy) is an instance of the *citizen-of* role, but we could not describe the concept Italian as a Person who is a *citizen-of* Italy. Using nominals, not only can we express this concept (i.e., $\text{Person} \sqcap \exists \text{citizen-of}.\text{Italy}$), but we can also capture Abox assertions with concept inclusion axioms of the form Giuseppe \sqsubseteq Italian (Giuseppe is an Italian) and Giuseppe $\sqsubseteq \exists \text{citizen-of}.\text{Italy}$ (Giuseppe is a *citizen-of* Italy).

Construct Name	Syntax	Semantics
atomic concept \mathbf{C}	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
abstract role \mathbf{R}_A	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
concrete role \mathbf{R}_D	T	$T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_D^{\mathcal{I}}$
nominals \mathbf{I}	o	$o^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}, \#o^{\mathcal{I}} = 1$
datatypes \mathbf{D}	d	$d^{\mathbf{D}} \subseteq \Delta_D$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
	$\neg d$	$(\neg d)^{\mathcal{I}} = \Delta_D \setminus d^{\mathcal{I}}$
exists restriction	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$
atleast restriction	$(\geq nS.C)$	$(\geq nS.C)^{\mathcal{I}} = \{x \mid \#(\{y. \langle x, y \rangle \in S^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \geq n\}$
atmost restriction	$(\leq nS.C)$	$(\leq nS.C)^{\mathcal{I}} = \{x \mid \#(\{y. \langle x, y \rangle \in S^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \leq n\}$
datatype exists	$\exists T.d$	$(\exists T.d)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y \rangle \in T^{\mathcal{I}} \text{ and } y \in d^{\mathbf{D}}\}$
datatype value	$\forall T.d$	$(\forall T.d)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y \rangle \in T^{\mathcal{I}} \text{ implies } y \in d^{\mathbf{D}}\}$

Figure 1: Syntax and semantics of $\mathcal{SHOQ}(\mathbf{D})$

$\mathcal{SHOQ}(\mathbf{D})$ Syntax and Semantics

Definition 1 Let \mathbf{C} , $\mathbf{R} = \mathbf{R}_A \uplus \mathbf{R}_D$, and \mathbf{I} be disjoint sets of *concept names*, abstract and concrete *role names*, and *individual names*.

For R and S roles, a *role axiom* is either a role inclusion, which is of the form $R \sqsubseteq S$ for $R, S \in \mathbf{R}_A$ or $R, S \in \mathbf{R}_D$, or a transitivity axiom, which is of the form $\text{Trans}(R)$ for $R \in \mathbf{R}_A$. A *role box* \mathcal{R} is a finite set of role axioms.

A role R is called *simple* if, for \sqsubseteq^* the transitive reflexive closure of \sqsubseteq on \mathcal{R} and for each role S , $S \sqsubseteq^* R$ implies $\text{Trans}(S) \notin \mathcal{R}$.

The set of $\mathcal{SHOQ}(\mathbf{D})$ -concepts is the smallest set such that each concept name $A \in \mathbf{C}$ is a concept, for each individual name $o \in \mathbf{I}$, o is a concept, and, for C and D concepts, R an abstract role, T a concrete role, S a simple role, and $d \in \mathbf{D}$ a concrete datatype, complex concepts can be built using the operators shown in Figure 1.

The semantics is given by means of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty domain $\Delta^{\mathcal{I}}$, disjoint from the concrete domain Δ_D , and a mapping $\cdot^{\mathcal{I}}$, which maps atomic and complex concepts, roles, and nominals according to Figure 1 ($\#$ denotes set cardinality). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* a role inclusion axiom $R_1 \sqsubseteq R_2$ iff $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$, and it satisfies a transitivity axiom $\text{Trans}(R)$ iff $R^{\mathcal{I}} = (R^{\mathcal{I}})^+$. An interpretation satisfies a role box \mathcal{R} iff it satisfies each axiom in \mathcal{R} .

A $\mathcal{SHOQ}(\mathbf{D})$ -concept C is *satisfiable* w.r.t. a role box \mathcal{R} iff there is an interpretation \mathcal{I} with $C^{\mathcal{I}} \neq \emptyset$ that satisfies \mathcal{R} . Such an interpretation is called a *model* of C w.r.t. \mathcal{R} . A concept C is *subsumed* by a concept D w.r.t. \mathcal{R} iff $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$ for each interpretation \mathcal{I} satisfying \mathcal{R} . Two concepts are said to be *equivalent* (w.r.t. \mathcal{R}) iff they mutually subsume each other (w.r.t. \mathcal{R}).

Some remarks are in order: In the following, if \mathcal{R} is clear from the context, we use $\text{Trans}(R)$ instead of $\text{Trans}(R) \in \mathcal{R}$.

Please note that the domain of each role is the abstract domain, and that we distinguish those roles whose range is also the abstract domain (*abstract roles*), and those whose range is the concrete domain (*concrete roles*). In the following, we use R for the former and T for the latter form of roles (possibly with index). We have chosen to disallow role inclusion axioms of the form $T \sqsubseteq R$ (or $R \sqsubseteq T$) for R an abstract and T a concrete role, since each model of such an axiom would necessarily interpret T (or R) as the empty relation.

Restricting number restrictions to simple roles is required to yield a decidable logic [Horrocks *et al.*, 1999].

Next, negation of concepts and datatypes is relativised to both the abstract and the concrete domain.

As usual, subsumption and satisfiability can be reduced to each other, and $\mathcal{SHOQ}(\mathbf{D})$ has the expressive power to *internalise* general concept inclusion axioms [Horrocks *et al.*, 1999]. However, in the presence of nominals, we must also add $\exists O.o_1 \sqcap \dots \sqcap \exists O.o_\ell$ to the concept internalising the general concept inclusion axioms to make sure that the universal role O indeed reaches all nominals o_i occurring in the input concept and terminology.

Finally, we did not choose to make a *unique name assumption*, i.e., two nominals might refer to the same individual. However, the inference algorithm presented below can easily be adapted to the unique name case by a suitable initialisation of the inequality relation \neq .

3 A Tableau for $\mathcal{SHOQ}(\mathbf{D})$

For ease of presentation, we assume all concepts to be in *negation normal form* (NNF). Each concept can be transformed into an equivalent one in NNF by pushing negation inwards, making use of deMorgan's laws and the following equivalences:

$$\begin{aligned}
\neg \exists R.C &\equiv \forall R.\neg C & \neg \forall R.C &\equiv \exists R.\neg C \\
\neg \exists T.d &\equiv \forall T.\neg d & \neg \forall T.d &\equiv \exists T.\neg d \\
\neg(\leq nR.C) &\equiv (\geq (n+1)R.C) \\
\neg(\geq (n+1)R.C) &\equiv (\leq nR.C) \\
\neg(\geq 0R.C) &\equiv C \sqcap \neg C
\end{aligned}$$

We use $\sim C$ to denote the NNF of $\neg C$. Moreover, for a concept D , we use $\text{cl}(D)$ to denote the set of all subconcepts of D , the NNF of these subconcepts, and the (possibly negated) datatypes occurring in these (NNFs of) subconcepts.

Definition 2 If D is a $\mathcal{SHOQ}(\mathbf{D})$ -concept in NNF, \mathcal{R} a role box, and $\mathbf{R}_A^D, \mathbf{R}_D^D$ are the sets of abstract and concrete roles occurring in D or \mathcal{R} , a *tableau* T for D w.r.t. \mathcal{R} is defined to be a quadruple $(\mathbf{S}, \mathcal{L}, \mathcal{E}_A, \mathcal{E}_D)$ such that: \mathbf{S} is a set of individuals, $\mathcal{L} : \mathbf{S} \rightarrow 2^{\text{cl}(D)}$ maps each individual to a set of concepts which is a subset of $\text{cl}(D)$, $\mathcal{E}_A : \mathbf{R}_A^D \rightarrow 2^{\mathbf{S} \times \mathbf{S}}$ maps each abstract role in \mathbf{R}_A^D to a set of pairs of individuals, $\mathcal{E}_D : \mathbf{R}_D^D \rightarrow 2^{\mathbf{S} \times \Delta_D}$ maps each concrete role in \mathbf{R}_D^D to a set of pairs of individuals and concrete values, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s, t \in \mathbf{S}$, $C, C_1, C_2 \in \text{cl}(D)$, $R, S \in \mathbf{R}_A^D$, $T, T' \in \mathbf{R}_D^D$, and

$$S^T(s, C) := \{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}_A(S) \text{ and } C \in \mathcal{L}(t)\},$$

it holds that:

- (P1) if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,
- (P2) if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,
- (P3) if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,
- (P4) if $\langle s, t \rangle \in \mathcal{E}_A(R)$ and $R \sqsubseteq S$, then $\langle s, t \rangle \in \mathcal{E}_A(S)$,
if $\langle s, t \rangle \in \mathcal{E}_D(T)$ and $T \sqsubseteq T'$, then $\langle s, t \rangle \in \mathcal{E}_D(T')$
- (P5) if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}_A(R)$, then $C \in \mathcal{L}(t)$,
- (P6) if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ such that
 $\langle s, t \rangle \in \mathcal{E}_A(R)$ and $C \in \mathcal{L}(t)$,
- (P7) if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}_A(R)$ for some $R \sqsubseteq S$
with $\text{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,
- (P8) if $(\geq n.S.C) \in \mathcal{L}(s)$, then $\#S^T(s, C) \geq n$,
- (P9) if $(\leq n.S.C) \in \mathcal{L}(s)$, then $\#S^T(s, C) \leq n$, and
- (P10) if $\{(\leq n.S.C), (\geq n.S.C)\} \cap \mathcal{L}(s) \neq \emptyset$ and $\langle s, t \rangle \in \mathcal{E}_A(S)$, then $\{C, \sim C\} \cap \mathcal{L}(t) \neq \emptyset$,
- (P11) if $o \in \mathcal{L}(s) \cap \mathcal{L}(t)$, then $s = t$,
- (P12) if $\forall T.d \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}_D(T)$, then $t \in d^D$,
- (P13) if $\exists T.d \in \mathcal{L}(s)$, then there is some $t \in \Delta_D$ such that
 $\langle s, t \rangle \in \mathcal{E}_D(T)$ and $t \in d^D$.

Lemma 3 A $\mathcal{SHOQ}(\mathbf{D})$ -concept D in NNF is satisfiable w.r.t. a role box \mathcal{R} iff D has a tableau w.r.t. \mathcal{R} .

Proof: We concentrate on (P11) to (P13), which cover the new logical features, i.e., nominals and datatypes; the remainder is similar to the proof found in [Horrocks *et al.*, 1999]. Roughly speaking, we construct a model \mathcal{I} from a tableau by taking \mathbf{S} as its interpretation domain and adding the missing role-successorships for transitive roles. Then, by induction on the structure of formulae, we prove that, if $C \in \mathcal{L}(s)$, then $s \in C^{\mathcal{I}}$. (P11) ensures that nominals are indeed interpreted as singletons, and (P12) and (P13) make sure that concrete datatypes are interpreted correctly.

For the converse, each model is by definition of the semantics a tableau. \square

4 A tableau algorithm for $\mathcal{SHOQ}(\mathbf{D})$

From Lemma 3, an algorithm which constructs a tableau for a $\mathcal{SHOQ}(\mathbf{D})$ -concept D can be used as a decision procedure for the satisfiability of D with respect to a role box \mathcal{R} . Such an algorithm will now be described in detail. Please note that, due to the absence of inverse roles, *subset blocking* is sufficient (see also [Baader and Sattler, 2000]) to ensure termination and correctness.

Definition 4 Let \mathcal{R} be a role box, D a $\mathcal{SHOQ}(\mathbf{D})$ -concept in NNF, \mathbf{R}_A^D the set of abstract roles occurring in D or \mathcal{R} , and \mathbf{I}^D the set of nominals occurring in D . A *completion forest* for D with respect to \mathcal{R} is a set of trees F where each node x of the forest is labelled with a set

$$\mathcal{L}(x) \subseteq \text{cl}(D) \cup \{\uparrow(R, o) \mid R \in \mathbf{R}_A^D \text{ and } o \in \mathbf{I}^D\},$$

and each edge $\langle x, y \rangle$ is labelled with a set of role names $\mathcal{L}(\langle x, y \rangle)$ containing roles occurring in $\text{cl}(D)$ or \mathcal{R} . Additionally, we keep track of inequalities between nodes of the

forest with a symmetric binary relation \neq between the nodes of F . For each $o \in \mathbf{I}^D$ there is a *distinguished* node x_o in F such that $o \in \mathcal{L}(x)$. We use $\uparrow(R, o) \in \mathcal{L}(y)$ to represent an R labelled edge from y to x_o .

Given a completion forest, a node y is called an R -*successor* of a node x if, for some R' with $R' \sqsubseteq R$, either y is a successor of x and $R' \in \mathcal{L}(\langle x, y \rangle)$, or $\uparrow(R', o) \in \mathcal{L}(x)$ and $y = x_o$. Ancestors and roots are defined as usual.

For a role S and a node x in F we define $S^F(x, C)$ by

$$S^F(x, C) := \{y \mid y \text{ is an } S\text{-successor of } x \text{ and } C \in \mathcal{L}(y)\}.$$

A node x is *directly blocked* if none of its ancestors are blocked, and it has an ancestor x' that is not distinguished such that $\mathcal{L}(x) \subseteq \mathcal{L}(x')$. In this case we will say that x' blocks x . A node is *blocked* if is directly blocked or if its predecessor is blocked.

For a node x , $\mathcal{L}(x)$ is said to contain a *clash* if

1. for some concept name $A \in N_C$, $\{A, \neg A\} \subseteq \mathcal{L}(x)$,
2. for some role S , $(\leq n.S.C) \in \mathcal{L}(x)$ and there are $n + 1$ S -successors y_0, \dots, y_n of x with $C \in \mathcal{L}(y_i)$ for each $0 \leq i \leq n$ and $y_i \neq y_j$ for each $0 \leq i < j \leq n$,
3. $\mathcal{L}(x)$ contains (possibly negated) datatypes d_1, \dots, d_n such that $d_1^D \cap \dots \cap d_n^D$ is empty, or if
4. for some $o \in \mathcal{L}(x)$, $x \neq x_o$.

If o_1, \dots, o_ℓ are all individuals occurring in D , the algorithm initialises the completion forest F to contain $\ell + 1$ root nodes $x_0, x_{o_1}, \dots, x_{o_\ell}$ with $\mathcal{L}(x_0) = \{D\}$ and $\mathcal{L}(x_{o_i}) = \{o_i\}$. The inequality relation \neq is initialised with the empty relation. F is then expanded by repeatedly applying the rules from Figure 2, stopping if a clash occurs in one of its nodes.

The completion forest is *complete* when, for some node x , $\mathcal{L}(x)$ contains a clash, or when none of the rules is applicable. If the expansion rules can be applied in such a way that they yield a complete, clash-free completion forest, then the algorithm returns “ D is satisfiable w.r.t. \mathcal{R} ”, and “ D is *unsatisfiable* w.r.t. \mathcal{R} ” otherwise.

Lemma 5 When started with a $\mathcal{SHOQ}(\mathbf{D})$ concept D in NNF, the completion algorithm terminates.

Proof: Let $m = |\text{cl}(D)|$, $k = |\mathbf{R}_A^D|$, n the maximal number in atleast number restrictions, and $\ell = |\mathbf{I}^D|$. Termination is a consequence of the following properties of the expansion rules: (1) Each rule but the \leq - or the \mathbf{O} -rule strictly extends the completion forest, by extending node labels or adding nodes, while removing neither nodes nor elements from node. (2) New nodes are only generated by the \exists - or the \geq -rule as successors of a node x for concepts of the form $\exists R.C$ and $(\geq n.S.C)$ in $\mathcal{L}(x)$. For a node x , each of these concepts can trigger the generation of successors at most once—even though the node(s) generated was later removed by either the \leq - or the \mathbf{O} -rule. If a successor y of x was generated for a concept $\exists S.C \in \mathcal{L}(x)$, and y is removed later, then there will always be some S -successor z of x such that $C \in \mathcal{L}(z)$, and hence the \exists -rule cannot be applied again to x and $\exists S.C$.

For the \geq -rule, if y_1, \dots, y_n were generated by an application of the \geq -rule for a concept $(\geq n.S.C)$, then $y_i \neq y_j$ is added for each $i \neq j$. This implies that there will always

<p>\sqcap-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$, then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$</p> <p>$\sqcup$-rule: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$, then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$</p> <p>\exists-rule: if $\exists R.C \in \mathcal{L}(x)$, (or $\exists T.d \in \mathcal{L}(x)$) x is not blocked, and x has no R-successor y with $C \in \mathcal{L}(y)$ (resp. no T-successor y with $d \in \mathcal{L}(y)$), then create a new node y with $\mathcal{L}(\langle x, y \rangle) = \{R\}$ and $\mathcal{L}(y) = \{C\}$ (resp. with $\mathcal{L}(\langle x, y \rangle) = \{T\}$ and $\mathcal{L}(y) = \{d\}$)</p> <p>\forall-rule: if $\forall R.C \in \mathcal{L}(x)$ (or $\forall T.d \in \mathcal{L}(x)$), x is not blocked, and there is an R-successor y of x with $C \notin \mathcal{L}(y)$, (resp. a T-successor y of x with $d \notin \mathcal{L}(y)$), then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$ (resp. $\mathcal{L}(y) = \mathcal{L}(y) \cup \{d\}$)</p> <p>$\forall_+$-rule: if $\forall S.C \in \mathcal{L}(x)$, x is not blocked, and there is some R with $\text{Trans}(R)$ and $R \sqsubseteq S$, and an R-successor y of x with $\forall R.C \notin \mathcal{L}(y)$, then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$</p> <p>choose-rule: $\{(\geq nS.C), (\leq nS.C)\} \cap \mathcal{L}(x) \neq \emptyset$, x is not blocked, and y is an S-successor of x with $\{C, \sim C\} \cap \mathcal{L}(y) = \emptyset$, then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \sim C\}$</p> <p>\geq-rule: if $(\geq nS.C) \in \mathcal{L}(x)$, x is not blocked, and there are no n S-successors y_1, \dots, y_n of x with $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$, then create n new nodes y_1, \dots, y_n with $\mathcal{L}(\langle x, y_i \rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.</p> <p>\leq-rule: if $(\leq nS.C) \in \mathcal{L}(x)$, x is not blocked, and x has $n + 1$ S-successors y_0, \dots, y_n with $C \in \mathcal{L}(y_i)$ for each $0 \leq i \leq n$, and there exist $i \neq j$ s. t. not $y_i \neq y_j$ and, if only one of y_i, y_j is distinguished, then it is y_i, then 1. $\mathcal{L}(y_i) = \mathcal{L}(y_i) \cup \mathcal{L}(y_j)$ and add $y \neq y_i$ for each y with $y \neq y_j$, and if both y_i, y_j are not distinguished, then 2. $\mathcal{L}(\langle x, y_i \rangle) = \mathcal{L}(\langle x, y_i \rangle) \cup \mathcal{L}(\langle x, y_j \rangle)$ if y_i is and y_j is not distinguished, then 2. $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\uparrow(S, o) \mid S \in \mathcal{L}(\langle x, y_j \rangle)\}$ for some $o \in \mathcal{L}(y_i)$ and 3. remove y_j and all edges leading to y_j from the completion forest</p> <p>O-rule: if $o \in \mathcal{L}(x)$, x is neither blocked nor distinguished, and not $x \neq x_o$ then, for z distinguished with $o \in \mathcal{L}(z)$, do 1. $\mathcal{L}(z) = \mathcal{L}(z) \cup \mathcal{L}(x)$, and 2. if x has a predecessor x', then $\mathcal{L}(x') = \mathcal{L}(x') \cup \{\uparrow(R, o) \mid R \in \mathcal{L}(\langle x', x \rangle)\}$, 3. add $y \neq z$ for each y with $y \neq x$, and remove x and all edges leading to x from the completion forest</p>

Figure 2: The complete tableaux expansion rules for $\mathcal{SHOQ}(\mathbf{D})$

be n S -successors y'_1, \dots, y'_n of x since neither the \leq -rule nor the **O**-rule ever merges two nodes y'_i, y'_j with $y'_i \neq y'_j$, and, whenever the \leq - or the **O**-rule removes a successor of x , there will be some S -successor z of x that “inherits” all inequalities from y'_i .

Hence the out-degree of the forest is bounded by nm .

(3) Nodes are labelled with subsets of $\text{cl}(D) \cup \{\uparrow(R, o) \mid R \in \mathbf{R}_A^D \text{ and } o \in \mathbf{I}^D\}$, so there are at most $2^{m+k\ell}$ different node labellings. Therefore, if a path p is of length at least $2^{m+k\ell}$, then, from the blocking condition in Definition 4, there are two nodes x, y on p such that x is directly blocked by y . Hence paths are of length at most $2^{m+k\ell}$. \square

Lemma 6 If a $\mathcal{SHOQ}(\mathbf{D})$ concept D in NNF has a tableau w.r.t. \mathcal{R} , then the expansion rules can be applied to D and \mathcal{R} such that they yield a complete, clash-free completion forest.

Proof: Again, we concentrate on the new features nominals and datatypes and refer the reader to [Horrocks *et al.*, 1999] for the remainder. Given a tableau T for D w.r.t. \mathcal{R} , we can apply the non-deterministic rules, i.e., the \sqcup -, choose-, and \leq -rule, in such a way that we obtain a complete and clash-free tableau: inductively with the generation of new nodes, we define a mapping π from nodes of the completion forest to individuals in the tableau and concrete values in such a way that $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ for $\pi(x) \in \mathbf{S}$ and, for each pair of nodes x, y and each (abstract or concrete) role R , if y is an R -successor of x , then $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}_A(R)$ or $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}_D(R)$. Please note that the latter also holds

in the case that y is not a successor of x but a distinguished node (i.e., $\uparrow(R, o) \in \mathcal{L}(x)$ and $y = x_o$), and in the case that y is a concrete value (i.e., $\pi(y) \notin \mathbf{S}$). Due to (P12) and (P13), we do not encounter a clash of the form (3), and (P11) makes sure that the **O**-rule can be applied correctly. \square

Lemma 7 If the expansion rules can be applied to a $\mathcal{SHOQ}(\mathbf{D})$ concept D in NNF and a role box \mathcal{R} such that they yield a complete and clash-free completion forest, then D has a tableau w.r.t. \mathcal{R} .

Proof: From a complete and clash-free completion forest F , we can obtain a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E}_A, \mathcal{E}_D)$ by *unravelling* as usual. That is, each element of the tableau is a *path* in the completion forest that starts at one of the root nodes and that, instead of going to a blocked node, goes to the node that is blocking this node (we disregard nodes that have datatypes in their labels). \mathcal{E} -successorship for abstract roles is defined according to the labels of edges (i.e., if $R' \in \mathcal{L}(\langle x_n, x_{n+1} \rangle)$ in F with $R' \sqsubseteq R$, then $\langle x_0 \dots x_n, x_0 \dots x_n x_{n+1} \rangle \in \mathcal{E}_A(R)$ in T) and following labels $\uparrow(R, o)$ (i.e., if $\uparrow(R, o) \in \mathcal{L}(x_n)$ in F , then $\langle x_0 \dots x_n, x_o \rangle \in \mathcal{E}_A(R)$). \mathcal{E} -successorship for concrete roles is defined following the edges to those (disregarded) nodes with datatypes in their labels. Clash-freeness makes sure that this is possible.

To satisfy (P8) also in cases where two R -successors y_1, y_2 of a node x with $(\geq nR.C)$ are blocked by the same node z , we must distinguish between individuals that, instead of going to y_i , go to z . This can be easily done as in [Horrocks *et*

al., 1999], annotating points in the path accordingly. Finally, we set $\mathcal{L}'(x_0 \dots x_n) = \mathcal{L}(x_n)$.

It remains to prove that T satisfies each (Pi). (P1) to (P10) are similar to those in [Horrocks et al., 1999]. (P11) is due to completeness (otherwise, the O-rule was applicable), which implies that nominals can be found only in the labels of distinguished nodes (note that the definition of blocking is such that a distinguished node can never block another one). (P12) and (P13) are due to the fact that F has no clash of form (3), and that the \exists - and \forall -rule are not applicable. \square

As an immediate consequence of Lemmas 3, 5, 6, and 7, the completion algorithm always terminates, and answers with “ D is satisfiable w.r.t. \mathcal{R} ” iff D is satisfiable w.r.t. \mathcal{R} . Next, subsumption can be reduced to (un)satisfiability. Finally, as we mentioned in Section 2, $\mathcal{SHOQ}(\mathbf{D})$ can internalise general concept inclusion axioms, and we can thus decide these inference problems also w.r.t. terminologies.

Theorem 8 The completion algorithm presented in Definition 4 is a decision procedure for satisfiability and subsumption of $\mathcal{SHOQ}(\mathbf{D})$ concepts w.r.t. terminologies.

5 Conclusion

As we have seen, ontologies are set to play a key rôle in the Semantic Web, where they will provide a source of shared and precisely defined terms for use in descriptions of web resources. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes.

We have presented the DL $\mathcal{SHOQ}(\mathbf{D})$, along with a sound and complete decision procedure for concept satisfiability/subsumption. With its support for both nominals and concrete datatypes, $\mathcal{SHOQ}(\mathbf{D})$ is well suited to the provision of reasoning support for ontology languages in general, and web based ontology languages in particular. In addition, the $\mathcal{SHOQ}(\mathbf{D})$ decision procedure is similar to the \mathcal{SHIQ} decision procedure implemented in the highly successful FaCT system, and should be amenable to a similar range of performance enhancing optimisations.

The only feature of languages such as OIL and DAML (and \mathcal{SHIQ}) that is missing in $\mathcal{SHOQ}(\mathbf{D})$ is inverse roles. Its exclusion was motivated by the very high complexity of reasoning that results from the unconstrained interaction of inverse roles with nominals and datatypes. Future work will include a detailed study of this interaction with a view to providing (restricted) support for inverse roles without triggering the explosion in complexity. An implementation (based on the FaCT system) is also planned, and will be used to test empirical performance.

References

- [Arecas et al., 2000] C. Arecas, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 2000. To appear.
- [Baader and Hanschke, 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, pages 452–457, 1991.
- [Baader and Sattler, 2000] F. Baader and U. Sattler. Tableau algorithms for description logics. In *Proc. TABLEAUX 2000*, vol. 1847 of *LNAI*, pages 1–18, 2000.
- [Berners-Lee, 1999] T. Berners-Lee. *Weaving the Web*. Orion Business Books, 1999.
- [Biron and Malhorta, 2000] Xml schema part 2: Datatypes. W3C Candidate Recommendation, Oct 2000. <http://www.w3.org/TR/xmlschema-2/>.
- [Blackburn and Seligman, 1998] P. Blackburn and J. Seligman. What are hybrid languages? In *Advances in Modal Logic*, vol. 1, pages 41–62. CSLI Publications, 1998.
- [Corcho and Pérez, 2000] O. Corcho and A. Gómez Pérez. Evaluating knowledge representation and reasoning capabilities of ontology specification languages. In *Proc. of ECAI-00 Workshop on Applications of Ontologies and Problem-Solving Methods*, 2000.
- [De Giacomo, 1995] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Università degli Studi di Roma “La Sapienza”, 1995.
- [Decker et al., 2000] S. Decker et al. The semantic web — on the respective roles of XML and RDF. *IEEE Internet Computing*, 2000.
- [Fensel et al., 2000] D. Fensel et al. OIL in a nutshell. In *Proc. of EKAW-2000*, *LNAI*, 2000.
- [Hendler and McGuinness, 2001] J. Hendler and D. L. McGuinness. The DARPA agent markup language. *IEEE Intelligent Systems*, 2001.
- [Horrocks et al., 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, vol. 1705 of *LNAI*, 1999.
- [Horrocks, 2000] I. Horrocks. Benchmark analysis with FaCT. In *Proc. TABLEAUX 2000*, vol. 1847 of *LNAI*, 2000.
- [Lutz, 2000] C. Lutz. Nexttime-complete description logics with concrete domains. In *Proceedings of the ESSLLI-2000 Student Session*, 2000.
- [McGuinness, 1998] D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS-98*. IOS-press, 1998.
- [Schaerf, 1994] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.
- [Streett, 1982] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Computation*, 54:121–141, 1982.
- [Tobies, 2000] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *JAIR*, 12:199–217, 2000.
- [Uschold and Grüninger, 1996] M. Uschold and M. Grüninger. Ontologies: Principles, methods and applications. *Knowledge Eng. Review*, 11(2), 1996.
- [van Heijst et al., 1997] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in KBS development. *Int. J. of Human-Computer Studies*, 46(2/3), 1997.

The \mathcal{SG} Family: Extensions of Simple Conceptual Graphs

Jean-François Baget

LIRMM

161, rue Ada

34392 Montpellier, Cedex 5

France

<http://www.lirmm.fr/~baget/>

Marie-Laure Mugnier

LIRMM

161, rue Ada

34392 Montpellier, Cedex 5

France

<http://www.lirmm.fr/~mugnier/>

Abstract

We introduce the \mathcal{SG} family of graph-based knowledge representation and reasoning models, basically extensions of the *simple conceptual graphs* model. Objects of these models are *colored simple graphs* and are used to represent *facts*, *rules* and *constraints*. Reasonings are based on graph-theoretic mechanisms, mainly graph homomorphism. Models of this family are defined by the kind of objects composing a knowledge base. In this paper, we focus on the formal definitions of these models, including their operational semantics and relationships with FOL, and we study their decidability properties and computational complexity.

1 Introduction

Conceptual graphs (CGs) have been proposed in [Sowa, 1984] as a knowledge representation and reasoning model, mathematically founded on logics and graph theory. Though mainly studied as a graphical interface for logics or as a diagrammatic system of logics, their graph-theoretic foundations have been less investigated. Most works in this area are limited to *simple conceptual graphs* (*simple graphs* or SGs) [Sowa, 1984; Chein and Mugnier, 1992], corresponding to the positive, conjunctive and existential fragment of FOL. This model has three fundamental characteristics:

1. objects are bipartite *labelled graphs* (nodes represent *entities* and *relations* between these entities);
2. reasonings are based on graph-theoretic operations, mainly a graph homomorphism called *projection*;
3. it is logically founded, reasonings being sound and complete w.r.t. a FOL semantics called Φ .

Main extensions of the SG model, keeping projection-based operations and sound and complete semantics, are *inference rules* [Gosh and Wuwongse, 1995; Salvat, 1998] and *nested graphs* [Chein *et al.*, 1998]; for *general CGs*, [Kerdiles, 1997] introduces an original deduction system, combining analytic tableaux with projection.

We present here a family of extensions of the simple graphs model. The common ground for these extensions is that objects are *colored simple graphs* representing *facts*, *rules* or

constraints, and operations are based upon projection; the deduction problem asks, given \mathcal{K} a knowledge base and Q a simple graph (which may represent a query, a goal, ...), whether Q can be deduced from \mathcal{K} . According to the kinds of objects considered in \mathcal{K} , different reasoning models are obtained, composing the \mathcal{SG} family. In this paper, we focus on the formal definitions of these models, including their operational semantics and relationships with FOL, and we study their decidability properties and computational complexity.

In section 2 basic definitions and results about simple graphs are recalled. Section 3 presents an overview of the \mathcal{SG} family. In particular, we explain why we consider SGs graphical features as essential for knowledge modeling and point out that these properties are preserved in the \mathcal{SG} family. In next sections we study the different members of the family.

2 Basic Notions: the \mathcal{SG} Model

Basic ontological knowledge is encoded in a structure called a *support*. Factual knowledge is encoded into *simple graphs* (SGs), which are bipartite labelled multigraphs (there can be several edges between two nodes). Elementary reasonings are computed by a graph homomorphism called *projection*.

Definition 1 (Support) A support is a 4-tuple $\mathcal{S} = (T_C, T_R, \mathcal{I}, \tau)$. T_C and T_R are two partially ordered finite sets, respectively of concept types and relation types. Relation types may be of any arity greater or equal to 1. \mathcal{I} is the set of individual markers and τ is a mapping from \mathcal{I} to T_C . We denote by $*$ the generic marker, where $* \notin \mathcal{I}$. The partial order on $\mathcal{I} \cup \{*\}$ considers elements of \mathcal{I} as pairwise non comparable, and $*$ as its greatest element.

Definition 2 (Simple Graph) A simple graph, defined on a support \mathcal{S} , is a bipartite multigraph $G = (V, E, \lambda)$, where $V = (V_C, V_R)$. V_C and V_R are the sets of concept nodes and relation nodes, E is the set of edges. Edges incident on a relation node are numbered from 1 to the degree of the node. We denote by $G_i(r)$ the i^{th} neighbor of a relation node r in G . Each node has a label given by the mapping λ . A concept node c is labelled by a couple (type(c), marker(c)), where type(c) is an element of T_C , called its type, and marker(c) is an element of $\mathcal{I} \cup \{*\}$, called its marker. If marker(c) = m is an individual marker, then type(c) = $\tau(m)$. A relation node r is labelled by type(r), an element of T_R , called its type, and the degree of r must be equal to the arity of type(r).

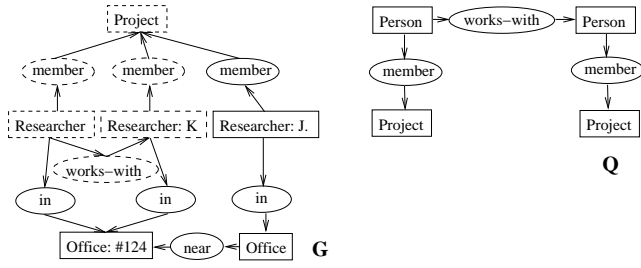


Figure 1: Simple graphs.

In the drawing of a SG, concept nodes are represented by rectangles and relation nodes by ovals. Generic markers are omitted. And since in our examples we use binary relations only, numbers on edges are replaced by directed edges: a relation node is incident to exactly one incoming and one outgoing edge. Fig. 1 shows two (connected) simple graphs assumed to be defined over the same support.

Simple graphs correspond to the existential conjunctive and positive fragment of FOL (by the semantics Φ). Types are mapped to predicates and individual markers to constants. A set of formulas $\Phi(S)$ is assigned to any support S , translating partial orders on types. Given any SG G , a formula $\Phi(G)$ is built as follows. A term is assigned to each concept node: a distinct variable for each generic node, and the constant corresponding to its marker otherwise. An atom $t(c)$ (resp. $t(c_1 \dots c_k)$) is associated to each concept node (resp. relation node r of arity k), where t is the type of the node, and c (resp. c_i) is the term assigned to this node (resp. assigned to $G_i(r)$). Let $\alpha(G)$ be the conjunction of these atoms. $\Phi(G)$ is the existential closure of $\alpha(G)$. E.g. the formula assigned to the dashed subgraph of G in Fig. 1 is $\exists x \exists y (Researcher(x) \wedge Project(y) \wedge Researcher(K) \wedge member(x, y) \wedge member(K, y) \wedge works-with(x, K))$.

Definition 3 (Projection) Let Q and G be two SGs defined on a support S , a projection from Q into G is a mapping π from $V_C(Q)$ to $V_C(G)$ and from $V_R(Q)$ to $V_R(G)$ that preserves edges and may decrease node labels:

1. $\forall c \in E(Q), \pi(c)\pi(r) \in E(G)$; and if $c = Q_i(r)$, then $\pi(c) = G_i(\pi(r))$;
2. $\forall x \in V(Q), \lambda(\pi(x)) \leq \lambda(x)$ (if x is a concept node, \leq is the product of the orders on T_C and $\mathcal{I} \cup \{*\}$, i.e. $type(\pi(x)) \leq type(x)$ and $marker(\pi(x)) \leq marker(x)$).

We note $Q \geq G$ (Q subsumes G) if there exists a projection from Q into G . Typically, Q represents a query, G a fact, and projections from Q to G define answers to Q . In Fig. 1, suppose $Researcher \leq Person$, then there is one projection from Q into G . The image of Q by this projection is the dashed subgraph of G . Projection is sound and complete w.r.t. the semantics Φ , up to a normality condition for completeness; the normal form of a SG G is the SG $nf(G)$ obtained by merging concept nodes having the same individual marker. This SG always exists (and is computable in linear time with a naive algorithm). Then: let Q and G be two SGs. $Q \geq nf(G) \Leftrightarrow \Phi(S), \Phi(G) \models \Phi(Q)$ [Chein and Mugnier, 1992], [Gosh and Wuwongse, 1995].

For the sake of brevity, we consider in what follows that SGs (and more complex constructs built upon SGs) are given in normal form, and put into normal form if needed after a modification. And, since a SG needs not be a connected graph, we confuse a set of SGs with the SG obtained by performing the disjoint union of its elements. In following definition for instance, the SG \mathcal{G} represents a set of SGs.

Definition 4 (SG Deduction) Let \mathcal{G} and Q be two SGs. Q can be deduced from \mathcal{G} if $Q \geq \mathcal{G}$.

The SG deduction problem is NP-complete [Chein and Mugnier, 1992]. Note that the subsumption relation induced by projection over SGs is a quasi-order. Two graphs are said to be *equivalent* if they project to each other. A SG is said to be *redundant* if it is equivalent to one of its strict subgraphs. Redundancy checking is an NP-complete problem. Each equivalence class admits a unique (up to isomorphism) non redundant graph [Chein and Mugnier, 1992].

3 Overview of the SG Family

From a modeling viewpoint, the simple graphs model has two essential properties. The objects, simple graphs, are easily understandable by an end-user (a knowledge engineer, or even an expert). And reasonings are easily understandable too, for two reasons: projection is a graph matching operation, thus easily interpretable and visualisable; and the same language is used at interface and operational levels. It follows that reasonings can be explained in a natural manner to the user, step by step, and directly on his own modelization (see for instance the knowledge engineering application described in [Bos *et al.*, 1997] or experiments in document retrieval done by [Genest, 2000], where in both cases the representation language is at the expert level). The SG family keeps these essential properties.

Let us now informally present the most general model of the family. Throughout this section, we will use examples inspired from a modelization of a knowledge acquisition case study, called Sysiphus-I: it describes a resource allocation problem, where the aim is to assign offices to persons of a research group while fulfilling constraints [Baget *et al.*, 1999].

Simple graphs are the basic constructs, upon which more complex constructs, rules and constraints, are defined, and operations are based on projection. A *rule* expresses a knowledge of form “if A holds then B can be added”. It is encoded into a simple graph provided with two colors, highlighting the hypothesis and the conclusion of the rule. In drawings, we represent the hypothesis by white nodes, and the conclusion by gray ones. Rules are used in the following way: if the hypothesis of a rule can be projected into a graph, then the rule is applicable to the graph, and its conclusion can be added to the graph according to the projection. The rule of Fig. 2 can be understood as “for all persons x and y , if x works with y , then y works with x ”. It can be applied to G of Fig. 1, adding a relation node (*work-with*) with predecessor [Researcher: K] and successor [Researcher].

A *constraint* can be a positive constraint or a negative constraint, expressing a knowledge of form “if A holds, so must B ”, or “if A holds, B must not”. It is also a bicolored simple graph: the first color defines the condition part, and the

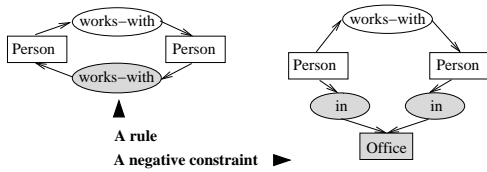


Figure 2: Colored SGs

second color the mandatory (or forbidden) part. A graph G satisfies a positive constraint C if *each* projection from the condition part of C into G can be extended as a projection of the whole C . And G satisfies a negative constraint if *no* projection of C into G can be extended as a projection of the whole C . Fig. 2 represents the negative constraint “two persons working together should not share an office”. The graph G of Fig 1 does not satisfy this constraint because “there is a researcher who works with researcher K” (projection of the condition part of C) “and they share office #124” (extension of the projection to a projection of the whole C).

When constraints are involved, we distinguish between two kinds of rules: *inference rules* of form “if A then add B ” and *evolution rules* of form “if A then add B , except if it brings inconsistency”. Now, a knowledge base contains four sets representing different kinds of knowledge: a set \mathcal{G} of *simple graphs* encoding factual knowledge, a set \mathcal{R} of *inference rules*, a set \mathcal{E} of *evolution rules* and a set \mathcal{C} of *constraints*.

Let us outline the deduction problem: the problem takes in input a knowledge base (KB) and a goal expressed as a SG, and asks whether there is a path of consistent worlds evolving from the initial one to a world satisfying the goal. Factual knowledge describes an initial world; inference rules represent implicit knowledge about worlds; evolution rules represent possible transitions from one world to other worlds; constraints define consistency of worlds; a successor of a consistent world is obtained by an evolution rule application; solving the problem consists in finding a path of consistent worlds evolving from the initial one to a world satisfying the goal.

In the particular case of the Sysiphus-I modelization, \mathcal{G} and \mathcal{R} describe initial information (office locations, persons and group organization), \mathcal{C} represents allocation constraints, \mathcal{E} consists of one evolution rule: “whenever there are a person and an office, try to assign this office to this person”. The goal represents a situation where each person of the group has an office. A solution to the problem is a world obtained from the initial one by a sequence of office assignments, where each person has an office, while satisfying allocation constraints.

Let us now specify definitions and notations concerning the \mathcal{SG} family. Rules and constraints are defined as colored SGs.

Definition 5 (colored SGs) A colored simple graph is a pair $K = (G, \rho)$ where G is a SG and ρ is a mapping from $V(G)$ into $\{0, 1\}$. The number associated to a node is called the color of the node. We denote by $K_{(i)}$ the subgraph of G induced by i -colored nodes. The subgraph $K_{(0)}$ must form a SG (i.e. the neighbors of a relation node of the hypothesis must also belong to the hypothesis).

A KB is denoted by $\mathcal{K} = (\mathcal{G}, \mathcal{R}, \mathcal{E}, \mathcal{C})$. Given a KB \mathcal{K} and a goal Q , the *deduction problem* asks whether Q can be de-

duced from \mathcal{K} (we note $Q \geq \mathcal{K}$). If we impose some of the sets \mathcal{R} , \mathcal{E} or \mathcal{C} to be empty, one obtains specific reasoning models. Note that in the absence of constraints ($\mathcal{C} = \emptyset$), inference and evolution rules have the same behavior, thus \mathcal{R} and \mathcal{E} can be confused. The \mathcal{SG} family is then composed of the six following models.

- the \mathcal{SG} model for $\mathcal{K} = (\mathcal{G}, \emptyset, \emptyset, \emptyset)$
- the \mathcal{SR} model for $\mathcal{K} = (\mathcal{G}, \mathcal{R}, \mathcal{E}, \emptyset)$
- the \mathcal{SGC} model for $\mathcal{K} = (\mathcal{G}, \emptyset, \emptyset, \mathcal{C})$
- the \mathcal{SRC} model for $\mathcal{K} = (\mathcal{G}, \mathcal{R}, \emptyset, \mathcal{C})$
- the \mathcal{SEC} model for $\mathcal{K} = (\mathcal{G}, \emptyset, \mathcal{E}, \mathcal{C})$
- the \mathcal{SREC} model for $\mathcal{K} = (\mathcal{G}, \mathcal{R}, \mathcal{E}, \mathcal{C})$

Since a fact has the same semantics as a rule with an empty hypothesis, the set \mathcal{G} is only used in models names when both rule sets \mathcal{R} and \mathcal{E} are empty. The hierarchy of these models is represented in Fig. 3. It highlights the decidability and complexity of the associated deduction problems.

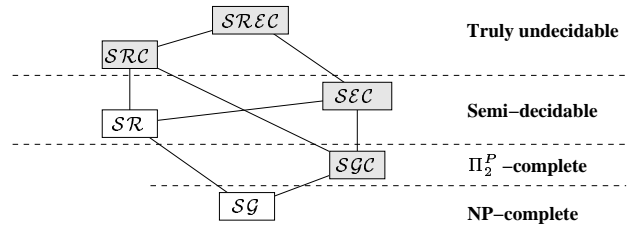


Figure 3: Reasoning models hierarchy for the \mathcal{SG} family

4 SGs and Rules: the \mathcal{SR} Model

A *simple graph rule* (SG rule) embeds knowledge of form “if A then B ”. The following definition is equivalent to the more traditional one (two SGs related with coreference links) used in [Gosh and Wuwongse, 1995; Salvat, 1998].

Definition 6 (SG Rules) A simple graph rule R is a colored SG. $R_{(0)}$ is called its hypothesis, and $R_{(1)}$ its conclusion.

Deduction depends on the notion of a *rule application*: it is a graph transformation based upon projection.

Definition 7 (Application of a SG Rule) Let G be a SG, and R be a SG rule. R is applicable to G if there exists a projection, say π , from $R_{(0)}$ (the hypothesis of R) into G . In that case, the result of the application of R on G according to π is the graph G' obtained by making the disjoint union of G and of a copy of $R_{(1)}$ (the conclusion of R), then, for every edge cr , where $c \in R_{(0)}$ and $r \in R_{(1)}$, adding an edge with the same number between $\pi(c)$ and the copy of r . G' is said to be an immediate R -derivation from G .

A derivation is naturally defined as a (possibly empty) sequence of rule applications:

Definition 8 (Derivation) Let \mathcal{R} be a set of SG rules, and G be a SG. We call \mathcal{R} -derivation from G to a SG G' a sequence of SGs $G = G_0, \dots, G_k = G'$ such that, for $1 \leq i \leq k$, G_i is an immediate R -derivation from G_{i-1} , where $R \in \mathcal{R}$.

To deduce a SG Q , we must be able to derive a SG into which Q can be projected, hence the following definition:

Definition 9 (Deduction in \mathcal{SR}) Let $\mathcal{K} = (\mathcal{G}, \mathcal{R})$ be a KB and let Q be a SG. Q can be deduced from \mathcal{K} (notation $Q \geq (\mathcal{G}, \mathcal{R})$) if there exists an \mathcal{R} -derivation from \mathcal{G} to a SG H such that $Q \geq H$.

The semantics Φ is extended to translate SG rules: let R_0 and R_1 be two SGs, s.t. $R_0 = R_{(0)}$ and R_1 is the SG obtained from $R_{(1)}$ by adding the neighbors of the relation nodes of $R_{(1)}$ which are concept nodes of $R_{(0)}$. Then $\Phi(R) = \forall x_1 \dots x_p (\alpha(R_0) \rightarrow \exists y_1 \dots y_q \alpha(R_1))$ where x_i are the variables of $\alpha(R_0)$ and y_j are the variables of $\alpha(R_1)$ that do not appear in $\alpha(R_0)$. The following soundness and completeness result is obtained: $Q \geq (\mathcal{G}, \mathcal{R}) \Leftrightarrow \Phi(\mathcal{S}), \Phi(\mathcal{G}), \Phi(\mathcal{R}) \models \Phi(Q)$ [Salvat, 1998]. The \mathcal{SR} deduction problem is semi-decidable [Coulondre and Salvat, 1998].

5 SGs and Constraints: the SGC Model

Let us now introduce *constraints*, which are used to validate knowledge. In presence of constraints, deduction is defined only on a *consistent* knowledge base.

Definition 10 (Constraints) A positive (resp. negative) constraint C is a colored SG. $C_{(0)}$ is called the trigger of the constraint, $C_{(1)}$ is called its obligation (resp. interdiction). A SG G π -violates a positive (resp. negative) constraint C if π is a projection of the trigger of C into the non redundant form of G (resp. into G) that cannot be extended (resp. that can be extended) to a projection of C as a whole. G violates C if it π -violates C for some projection π . Otherwise, G satisfies C .

Notice there may be two equivalent SGs, such that one satisfies a positive constraint and the other does not. E.g. given a constraint C , take G satisfying C , and the equivalent (redundant) graph H obtained by making the disjoint union of G and the trigger of C . H violates C . We have thus chosen to define constraint satisfaction w.r.t. the non redundant form of a SG. This problem does not occur with negative constraints.

Two constraints C_1 and C_2 are said *equivalent* if, for every graph G , G violates C_1 iff G violates C_2 . Any negative constraint C is equivalent to the negative constraint obtained from C by coloring all its nodes by 1. Furthermore, negative constraints are indeed a particular case of positive ones: consider the positive constraint C' obtained from a negative one C by coloring all nodes of C by 0, then adding a concept node typed `NotThere`, colored by 1, where `NotThere` is incomparable with all other types and does not appear anywhere excepted in \mathcal{C} . Then a graph G violates C if and only if it violates C' . Positive constraints strictly include negative constraints, in the sense that the associated consistency problems are not in the same complexity class (see theorem 3).

Let us relate our definitions to other definitions of constraints found in the CG literature. The constraints of [Mineau and Missaoui, 1997] correspond to a particular case of our constraints where the trigger and the obligation are not connected. In turn, our constraints are a particular case of the minimal descriptive constraints of [Dibie *et al.*, 1998], where the disjunctive part is restricted to one graph.

Definition 11 (Consistency/Deduction in SGC) A KB $\mathcal{K} = (\mathcal{G}, \mathcal{C})$ is consistent if \mathcal{G} satisfies all constraints of \mathcal{C} . A SG Q can be deduced from \mathcal{K} if \mathcal{K} is consistent and Q can be deduced from \mathcal{G} .

Note that a Q that violates a constraint of \mathcal{K} may still be deduced from \mathcal{K} . It does not matter since Q is a *partial* representation of knowledge deducible from \mathcal{K} . How can we translate the notion of consistency into FOL? For negative constraints, the correspondence is immediate, and relies on projection soundness and completeness.

Theorem 1 A SG G violates a negative constraint $C = (C', \rho)$ iff $\Phi(\mathcal{S}), \Phi(G) \models \Phi(C')$, where C' is the SG underlying C .

Consistency relative to positive constraints can be explained with FOL, translating “projection” into a notion of logical “substitution” between the formulas associated to graphs (see for instance the S-substitution of [Chein and Mugnier, 1992]). Another bridge can be built using rules. Indeed, a graph G satisfies a positive constraint C if and only if, considering C as a rule, all applications of C on G produce a graph equivalent to G . Or, more specifically:

Property 1 A SG G π -violates a positive constraint C iff, considering C as a rule, the application of C on G according to π produces a graph not equivalent to G .

Using soundness and completeness of the \mathcal{SR} deduction, and property 1, one obtains the following relation with FOL:

Theorem 2 A SG G violates a positive constraint C iff there is a SG G' such that $\Phi(\mathcal{S}), \Phi(G), \Phi(C) \models \Phi(G')$ and not $\Phi(\mathcal{S}), \Phi(G) \models \Phi(G')$, where $\Phi(C)$ is the translation of C considered as a rule.

The problem “does a given graph satisfy a given constraint?” is co-NP-complete if this constraint is a negative constraint (we must check the absence of projection), but becomes Π_2^P -complete for a positive one (Π_2^P is co-NP^{NP}).

Theorem 3 (Complexity in SGC) Consistency in SGC is Π_2^P -complete (but is co-NP-complete if all constraints are negative).

Sketch of proof: SGC -Consistency belongs to Π_2^P since it corresponds to the language $L = \{x | \forall y_1 \exists y_2 R(x, y_1, y_2)\}$, where x encodes an instance $G; C$ and $(x, y_1, y_2) \in R$ iff y_1 is a projection Π_0 from $C_{(0)}$ into G , y_2 is a projection Π from C into G s.t. $\Pi[C_{(0)}] = \Pi_0$. Now, let us consider the Π_2^P -complete problem B_2^c : given a boolean formula E , and a partition $\{X_1, X_2\}$ of its variables, is it true that for any truth assignment for the variables in X_1 there exists a truth assignment for the variables in X_2 s.t. E is true? We first show that the special case where E is an instance of 3-SAT remains Π_2^P -complete (let us call this problem 3-SAT₂^c). For that we use the polynomial transformation from any boolean formula to a set of clauses described in ([Papadimitriou, 1994], example 8.3), followed by a polynomial transformation from a clause to clauses with 3 literals. The “new” variables are put in the set X_2 . We then reduce 3-SAT₂^c to SGC -Consistency. E is mapped to a constraint C : briefly said, there is one concept node [t:*] for each variable and one ternary relation node

with type r_i for each clause c_i . $C_{(0)}$ is composed of all concept nodes coming from variables in X_1 . G is composed of two individual concept nodes [t:0] and [t:1] corresponding to the truth values and, for each clause c_i there is one relation node of type r_i for each triple of truth values giving the value true to c_i . Note that, since $C_{(0)}$ does not contain any relation node, any truth assignment for the variables of X_1 is a projection from $C_{(0)}$ to G , and reciprocally. \square

Corollary 1 *Deduction in SGC is Π_2^P -complete.*

Note that the theorem 3 can be used to show that consistency of minimal descriptive constraints in [Dibie *et al.*, 1998] is also Π_2^P -complete.

6 Rules and Constraints: SEC/SRC

The two kinds of rules, inference rules \mathcal{R} and evolution rules \mathcal{E} , define two alternative models. In SEC , \mathcal{G} is seen as the initial world, root of a potentially infinite tree of possible worlds, and \mathcal{E} describes the possible evolutions from one world to others. The deduction problem asks whether there is a path of consistent worlds from \mathcal{G} to a world satisfying Q .

Definition 12 (Deduction in SEC) *Let $\mathcal{K} = (\mathcal{G}, \mathcal{E}, \mathcal{C})$ be a KB, and let Q be a SG. Q can be deduced from \mathcal{K} if there is an \mathcal{E} -derivation $\mathcal{G} = G_0, \dots, G_k$ such that, for $0 \leq i \leq k$, (G_i, \mathcal{C}) is consistent and Q can be deduced from G_k .*

In SRC , \mathcal{G} provided with \mathcal{R} is a finite description of a potentially infinite world, that has to be consistent. Applying a rule to \mathcal{G} can create inconsistency, but a further application of a rule may restore consistency. Let us formalize this notion of *consistency restoration*. Suppose there is a π -violation of a positive constraint C in \mathcal{G} ; this violation (C, π) is said to be \mathcal{R} -restorable if there exists an \mathcal{R} -derivation from \mathcal{G} into G' such that the projection π of the trigger of C into G' can be extended to a projection of C as a whole. The violation of a negative constraint can never be restored. Note that the \mathcal{R} -restoration can create new violations, that must themselves be proven \mathcal{R} -restorable.

Definition 13 (Consistency/Deduction in SRC) *A KB $\mathcal{K} = (\mathcal{G}, \mathcal{R}, \mathcal{C})$ is consistent if, for any SG G' that can be \mathcal{R} -derived from \mathcal{G} , for every constraint $C \in \mathcal{C}$, for every π -violation of C in G' , (C, π) is \mathcal{R} -restorable. A SG Q can be deduced from \mathcal{K} if \mathcal{K} is consistent and Q can be deduced from $(\mathcal{G}, \mathcal{R})$.*

Consider for instance a KB containing the SG G in Fig. 4, expressing the existence of the number 0, a constraint and a rule, both represented by the colored SG K . The constraint asserts that for every integer n , there must be an integer n' , successor of n . If the rule is an evolution rule, G is seen as an inconsistent initial world (there is no successor of 0 in G) and nothing will be deduced from this KB. If the rule is an inference rule, its application immediately repairs the constraint violation, while creating a new integer, that has no successor, thus a new violation. Finally, every constraint violation could eventually be repaired by a rule application, and the KB should be proven consistent.

Let us point out that the SRC model is obtained from SRC or SEC when \mathcal{C} is empty, and SGC is obtained from SRC (resp. SEC) when \mathcal{R} (resp. \mathcal{E}) is empty.

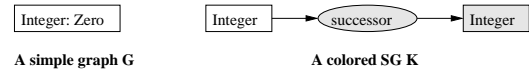


Figure 4: Consistency in SEC/SRC

Theorem 4 (Complexity in SEC/SRC) *Deduction in SEC is semi-decidable. Consistency and deduction in SRC are truly undecidable.*

Proof: SEC includes SRC thus SEC -deduction is not decidable. When Q is deducible from \mathcal{K} , a breadth-first search of the tree of all derivations from \mathcal{K} , each graph being checked for consistency, ensures that G_k is found in finite time. For SRC , we show that checking consistency is truly undecidable. Let \mathcal{K} be a KB where \mathcal{C} contains a positive constraint C^+ and a negative constraint C^- , both with an empty trigger. For proving consistency, one has to prove that $C^- \not\prec (\mathcal{G}, \mathcal{R})$, and the algorithm does not stop in this case (from semi-decidability of deduction in SRC). The same holds for the complementary problem (proving inconsistency) taking C^+ instead of C^- , hence the undecidability. \square

7 Combining Inference and Evolution

The $SRECE$ model combines both derivation schemes of the SRC and SEC models. Now, \mathcal{G} describes an initial world, inference rules of \mathcal{R} complete the description of any world, constraints of \mathcal{C} evaluate the consistency of a world, evolution rules of \mathcal{E} try to make evolve a consistent world into a new, consistent one. The deduction problem is: can \mathcal{G} evolve into a consistent world satisfying the goal?

Definition 14 (Deduction in $SRECE$) *A SG G' is an immediate \mathcal{RE} -evolution from a SG G if there exists an \mathcal{R} -derivation from G into G'' and an immediate \mathcal{E} -derivation from G'' into G' . An \mathcal{RE} -evolution from a SG G to a SG G' is a sequence of SGs $G = G_0, \dots, G_k = G'$ such that, for $1 \leq i \leq k$, G_i is an immediate \mathcal{RE} -evolution from G_{i-1} . Given a KB $\mathcal{K} = (\mathcal{G}, \mathcal{R}, \mathcal{E}, \mathcal{C})$, a SG Q can be deduced from \mathcal{K} if there is an \mathcal{RE} -evolution $\mathcal{G} = G_0, \dots, G_k$ where, for $0 \leq i \leq k$, $(G_i, \mathcal{R}, \mathcal{C})$ is consistent, and Q can be deduced from (G_k, \mathcal{R}) .*

When $\mathcal{E} = \emptyset$, one obtains the SRC model (there is only one world). When $\mathcal{R} = \emptyset$, one obtains SEC (information contained in a world is complete). As a generalization of SRC , deduction in $SRECE$ is truly undecidable. Let us now consider a decidable fragment of $SRECE$ (which in particular was sufficient for the Sysiphus-I modelization). First note a rule has only to be applied once according to a given projection: further applications with this projection obviously produce redundant information. Such applications are said to be *useless*. A SG G is said to be *closed* w.r.t. a given rule R if all applications of R on G are useless. Given a set of rules \mathcal{R} , we note, when it exists, $G_{\mathcal{R}}^*$ (the closure of G w.r.t. \mathcal{R}) the smallest graph derived from G that is closed w.r.t. every rule in \mathcal{R} . When it exists $G_{\mathcal{R}}^*$ is unique. \mathcal{R} is called a *finite expansion set* (f.e.s.) if, for every SG G , its closure $G_{\mathcal{R}}^*$ exists (and thus can be computed in finite time). If \mathcal{R} is a finite expansion set, the deduction problem in the SRC model becomes decidable (but it is not a necessary condition for decidability).

Property 2 (Finite expansion sets) Let $\mathcal{K} = (\mathcal{G}, \mathcal{R}, \mathcal{C})$ be a KB where \mathcal{R} is a finite expansion set. Then \mathcal{K} is consistent iff $(\{\mathcal{G}_{\mathcal{R}}^*\}, \mathcal{C})$ is consistent, and a SG Q can be deduced from $(\mathcal{G}, \mathcal{R})$ iff Q can be deduced from $(\{\mathcal{G}_{\mathcal{R}}^*\})$.

The following decidable case is based on property 2.

Property 3 (Decidable case) Deduction in $SR\mathcal{E}\mathcal{C}$ is semi-decidable if \mathcal{R} is a f.e.s. and is decidable if $\mathcal{R} \cup \mathcal{E}$ is a f.e.s.

Proof: Suppose \mathcal{R} is a f.e.s. When Q can be deduced, one obtains an answer in finite time; we proceed as for $\mathcal{S}\mathcal{E}\mathcal{C}$ (see proof of theorem 4) but consistency checks are done on the graph closure instead of the graph itself. Now, if $\mathcal{R} \cup \mathcal{E}$ is a f.e.s., $\mathcal{G}_{\mathcal{R} \cup \mathcal{E}}^*$ exists, thus the derivation tree is finite, and consistency checks may only cut some parts of this tree. \square

A rule is said to be *range restricted* (by analogy with the so-called rules in Datalog, where all variables of the head must appear in the conclusion) if no generic concept node belongs to its conclusion. Then:

Property 4 A set of range restricted rules is a f.e.s.

Proof: Since all graphs are put into normal form, an individual marker appears at most once in a graph. Then the closure of a SG G can be obtained with a derivation of length $\mathcal{O}(n^k)$, where n is the size of (G, \mathcal{R}) and k is the greatest arity of a relation type appearing in a rule conclusion. \square

8 Conclusion

We propose a family of models that can be seen as the basis of a generic modeling framework. Main features of this framework are the following: a clear distinction between different kinds of knowledge, that fit well with intuitive categories, a uniform graph-based language that keeps essential properties of the SG model, namely readability of objects as well as reasonings. We guess this later point is particularly important for the usability of any knowledge based system. In our framework, all kinds of knowledge are graphs easily interpreted, and reasonings can be graphically represented in a natural manner using the graphs themselves, thus explained to the user on its own modelization.

Technical contributions, w.r.t. previous works on conceptual graphs, can be summarized as follows:

- the representation of different kinds of knowledge as colored SGs: facts, inference rules, evolution rules and constraints.
- the integration of constraints into a reasoning model; more or less similar notions of a constraint had already been introduced in [Dibie *et al.*, 1998; Mineau and Missaoui, 1997] but were only used to check consistency of a simple graph (as in the $S\mathcal{G}\mathcal{C}$ model). The complexity of consistency checking was not known.
- a systematic study of the obtained family of models with a complexity classification of associated consistency/deduction problems.

We established links between consistency checking and FOL deduction. The operational semantics of models including constraints, namely $SR\mathcal{E}\mathcal{C}$, $SR\mathcal{C}$ and $\mathcal{S}\mathcal{E}\mathcal{C}$, is easy to

understand but there is an underlying non monotonic mechanism whose logical interpretation should require non standard logics. The definition of a logical semantics for these models is an open problem.

Acknowledgments

We are indebted to Geneviève Simonet for her careful reading and some error corrections on an earlier version of this paper, and to Michel Chein, as well as the anonymous reviewers, for their helpful comments.

References

- [Baget *et al.*, 1999] J.-F. Baget, D. Genest, and M.-L. Mugnier. Knowledge Acquisition with a Pure Graph-Based Knowledge Representation Model – Application to the Sisyphus-I Case Study. In *Proc. of KAW'99*, 1999.
- [Bos *et al.*, 1997] C. Bos, B. Botella, and P. Vanheeghe. Modeling and Simulating Human Behaviors with Conceptual Graphs. In *Proc. of ICCS'97, LNAI 1257*, pages 275–289. Springer, 1997.
- [Chein and Mugnier, 1992] M. Chein and M.-L. Mugnier. Conceptual Graphs: Fundamental Notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.
- [Chein *et al.*, 1998] M. Chein, M.-L. Mugnier, and G. Simonet. Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics. In *Proc. of KR'98*, pages 524–534. Morgan Kaufmann, 1998.
- [Coulondre and Salvat, 1998] S. Coulondre and E. Salvat. Piece Resolution: Towards Larger Perspectives. In *Proc. of ICCS'98, LNAI 1453*, pages 179–193. Springer, 1998.
- [Dibie *et al.*, 1998] J. Dibie, O. Haemmerlé, and S. Loiseau. A Semantic Validation of Conceptual Graphs. In *Proc. of ICCS'98, LNAI 1453*, pages 80–93. Springer, 1998.
- [Genest, 2000] D. Genest. *Extension du modèle des graphes conceptuels pour la recherche d'informations*. PhD thesis, Université Montpellier II, Dec. 2000.
- [Gosh and Wuwongse, 1995] B. C. Gosh and V. Wuwongse. A Direct Proof Procedure for Definite Conceptual Graphs Programs. In *Proc. of ICCS'95, LNAI 954*, pages 158–172. Springer, 1995.
- [Kerdiles, 1997] G. Kerdiles. Projection: A Unification Procedure for Tableaux in Conceptual Graphs. In *Proc. of TABLEAUX'97, LNAI 1227*, pages 216–230, 1997.
- [Mineau and Missaoui, 1997] G. W. Mineau and R. Missaoui. The Representation of Semantic Constraints in Conceptual Graphs Systems. In *Proc. of ICCS'97, LNAI 1257*, pages 138–152. Springer, 1997.
- [Papadimitriou, 1994] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [Salvat, 1998] E. Salvat. Theorem proving using graph operations in the conceptual graphs formalism. In *Proc. of ECAI'98*, 1998.
- [Sowa, 1984] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

KNOWLEDGE REPRESENTATION AND REASONING

DESCRIPTION LOGICS AND
FORMAL CONCEPT ANALYSIS

Matching under Side Conditions in Description Logics *

Franz Baader and Sebastian Brandt

Theoretical Computer Science, RWTH Aachen

email: {baader,sbrandt}@cs.rwth-aachen.de

Ralf Küsters

Institute for Computer Science and Applied Mathematics, University of Kiel

email: kuesters@ti.informatik.uni-kiel.de

Abstract

Whereas matching in Description Logics is now relatively well-investigated, there are only very few formal results on matching under additional side conditions, though these side conditions were already present in the original paper by Borgida and McGuinness introducing matching in DLs. The present paper closes this gap for the DL \mathcal{ALN} and its sublanguages: matching under subsumption conditions remains polynomial, while strict subsumption conditions increase the complexity to NP.

1 Introduction

The traditional inference problems (like subsumption) in description logics (DLs) are now well-investigated, which means that there exist complexity results and algorithms for a great variety of DLs of differing expressive power [Donini *et al.*, 1996] as well as optimized implementations of the algorithms for expressive DLs [Horrocks, 1998]. In contrast, matching concepts against patterns is a relatively new inference problem in DLs, which has originally been introduced in [Borgida and McGuinness, 1996; McGuinness, 1996] to help filter out the unimportant aspects of large concepts appearing in knowledge bases of the CLASSIC system [Brachmann *et al.*, 1991]. More recently, matching (as well as the more general problem of unification) has been proposed as a tool for detecting redundancies in knowledge bases [Baader and Narendran, 1998] and to support the integration of knowledge bases by prompting possible interschema assertions [Borgida and Küsters, 2000].

All three applications have in common that one wants to search a large knowledge base for concepts having a certain (not completely specified) form. This “form” can be expressed with the help of so-called *concept patterns*, i.e., concept descriptions containing variables. For example, the pattern $D := X \sqcap \forall \text{child}.(Y \sqcap \text{Female})$ looks for concepts that restrict the *child* role to fillers that are *Female*, such as the concept $C := (\geq 1 \text{ child}) \sqcap \forall \text{child}.(Female \sqcap Rich)$. In fact, applying the substitution $\sigma := \{X \mapsto (\geq 1 \text{ child}), Y \mapsto$

$Rich\}$ to the pattern D yields a concept equivalent to C , i.e., σ is a solution (matcher) of the matching problem $C \equiv^? D$.

This type of matching problems has been investigated in detail for sublanguages of the DLs \mathcal{ALN} and \mathcal{ALC} in [Baader *et al.*, 1999] and [Baader and Küsters, 2000], respectively. In particular, it was shown that, for sublanguages of \mathcal{ALN} , solvable matching problems always have a least matcher (w.r.t. subsumption), which can be computed in polynomial time. For sublanguages of \mathcal{ALC} , deciding solvability of matching problems modulo equivalence is already NP-complete.

In [Borgida and McGuinness, 1996; McGuinness, 1996], the expressivity of matching problems was further enhanced by allowing for additional *side conditions* on the variables (through the **as**-construct): a (strict) subsumption condition is of the form $X \sqsubseteq^? E$ ($X \sqsubset^? E$) where X is a variable and E a pattern, and it restricts the matchers to substitutions satisfying $\sigma(X) \sqsubseteq \sigma(E)$ ($\sigma(X) \sqsubset \sigma(E)$). Using a subsumption condition, the matching problem of the above example can be written more intuitively as $X \sqcap \forall \text{child}.Z \equiv^? (\geq 1 \text{ child}) \sqcap \forall \text{child}.(Female \sqcap Rich)$ under the subsumption condition $Z \sqsubseteq^? Female$. One result of this paper is that also more complex sets of subsumption conditions do not extend the expressive power of matching problems (see below). However, they are often more convenient to state. In contrast, strict subsumption conditions cannot always be simulated by pure matching problems. They can, e.g., be used to avoid trivial matches. For example, the pattern $D' := X \sqcap \forall \text{child}.Y$ matches every concept since $\forall \text{child}.\top \equiv \top$ (where the top concept \top stands for the set of all individuals). The additional strict subsumption condition $Y \sqsubset^? \top$ ensures that we can only match concepts with a real restriction on *child*.

The first (rather restricted) formal results on matching under side conditions were given in [Baader *et al.*, 1999]: it was shown that matching under strict subsumption conditions in the small DL \mathcal{FL}_0 is already NP-hard, and that matching under so-called acyclic subsumption conditions can be reduced to matching without side conditions. However, [Baader *et al.*, 1999] does not give a complexity upper bound for matching under strict subsumption conditions and the reduction for acyclic subsumption conditions given there is exponential.

This paper investigates in detail matching under side conditions in sublanguages of \mathcal{ALN} . We will show that matching under subsumption conditions can be reduced in polynomial time to matching without side conditions. In particular,

*This work has been partially supported by the Deutsche Forschungsgemeinschaft, DFG Project BA 1122/4-1.

this implies that solvable matching problems under subsumption conditions in sublanguages of \mathcal{ALN} always have a least matcher, which can be computed in polynomial time. For acyclic strict subsumption conditions, matching is shown to be NP-complete in the sublanguages \mathcal{FL}_\perp and \mathcal{FL}_\neg of \mathcal{ALN} .

2 Description logics

Concept descriptions are inductively defined with the help of a set of concept *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. In this paper, we consider concept descriptions built from the constructors shown in Table 1. In the description logic \mathcal{FL}_0 , concept descriptions are formed using the constructors top-concept (\top), conjunction ($C \sqcap D$), and value restriction ($\forall r.C$). The description logic \mathcal{FL}_\perp additionally provides us with the bottom concept (\perp), and \mathcal{FL}_\neg also allows for primitive negation ($\neg P$). Finally, \mathcal{ALN} extends \mathcal{FL}_\neg with number restrictions ($\geq n r$) and ($\leq n r$) (see Table 1).

As usual, the semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of \mathcal{I} is a non-empty set and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $P \in N_C$ to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is defined inductively, as shown in the second column of Table 1.

One of the most important traditional inference services provided by DL systems is computing the subsumption hierarchy. The concept description C is *subsumed* by the description D ($C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations \mathcal{I} ; C and D are *equivalent* ($C \equiv D$) iff they subsume each other; C is *strictly subsumed* by D ($C \sqsubset D$) iff $C \sqsubseteq D$ and $C \not\equiv D$. For all DLs listed in Table 1, subsumption can be decided in polynomial time using a structural subsumption algorithm [Borgida and Patel-Schneider, 1994].

Matching in description logics

In order to define concept patterns, we additionally need a set N_X of concept variables, which we assume to be disjoint from $N_C \cup N_R$. Informally, an \mathcal{ALN} -concept pattern is an \mathcal{ALN} -concept description over the concept names $N_C \cup N_X$ and the role names N_R , with the only exception that primitive negation must not be applied to variables. More formally, *concept patterns* (denoted D, D') are defined using the following syntax rules:

$$D, D' \longrightarrow X \mid C \mid D \sqcap D' \mid \forall r.D,$$

where $X \in N_X$, $r \in N_R$, and C is an \mathcal{ALN} -concept description. For example, if X, Y are concept variables, r a role name, and A, B concept names, then $D := A \sqcap X \sqcap \forall r.(B \sqcap Y)$ is an \mathcal{ALN} -concept pattern, but $\neg X$ is *not*. The notion of a pattern (and also the notions “substitution” and “matching problem” introduced below) can be restricted to sublanguages of \mathcal{ALN} in the obvious way.

A *substitution* σ is a mapping from N_X into the set of all \mathcal{ALN} -concept descriptions. This mapping is extended to concept patterns in the usual way by replacing the occurrences of the variables X in the pattern by the corresponding concept description $\sigma(X)$. For example, if we apply the substitution $\sigma := \{X \mapsto A \sqcap B, Y \mapsto A\}$ to the pattern D from above,

we obtain the description $\sigma(D) = A \sqcap A \sqcap B \sqcap \forall r.(B \sqcap A)$. The result of applying a substitution to an \mathcal{ALN} -concept pattern is always an \mathcal{ALN} -concept description. Note that this would no longer be the case if negation were allowed in front of concept variables.

Subsumption can be extended to substitutions as follows: the substitution σ is subsumed by the substitution τ ($\sigma \sqsubseteq \tau$) iff $\sigma(X) \sqsubseteq \tau(X)$ for all variables $X \in N_X$.

Definition 1 *Let C be an \mathcal{ALN} -concept description and D an \mathcal{ALN} -concept pattern. Then, $C \equiv^? D$ is an \mathcal{ALN} -matching problem. The substitution σ is a solution (matcher) of $C \equiv^? D$ iff $C \equiv \sigma(D)$.*

In the following, we will abbreviate a matching problem of the form $C \equiv^? C \sqcap D$ as $C \sqsubseteq^? D$. This notation is justified by the fact that σ solves $C \equiv^? C \sqcap D$ iff $C \sqsubseteq \sigma(D)$.

A matching problem can either be viewed as a *decision problem*, where one asks whether the problem is solvable, or as a *computation problem*, where one asks for actual matchers of this problem (if any). Although the computation problem is usually the more interesting one, the decision problem can serve as a starting point for the complexity analysis. In general, matching problems may have several (even an infinite number of) solutions, and thus the question arises which matcher to compute. Following [Borgida and McGuinness, 1996; Baader *et al.*, 1999] we will here concentrate on the problem of computing a least matcher (w.r.t. the ordering \sqsubseteq on substitutions).

Instead of a single matching problem, we may also consider finite systems $\{C_1 \equiv^? D_1, \dots, C_m \equiv^? D_m\}$ of such problems, which must be solved simultaneously. As shown in [Baader *et al.*, 1999], solving such a system can, however, be reduced to solving the single matching problem

$$\forall r_1.C_1 \sqcap \dots \sqcap \forall r_m.C_m \equiv^? \forall r_1.D_1 \sqcap \dots \sqcap \forall r_m.D_m$$

where the r_i are pairwise distinct role names.

The following theorem summarizes the results obtained in [Baader and Narendran, 1998; Baader *et al.*, 1999] for matching in (sublanguages of) \mathcal{ALN} :

Theorem 2 *Let $\mathcal{L} \in \{\mathcal{FL}_0, \mathcal{FL}_\perp, \mathcal{FL}_\neg, \mathcal{ALN}\}$. Then there exists a polynomial time matching algorithm that computes the least matcher of a given system of \mathcal{L} -matching problems, if this system has a solution, and returns “fail” otherwise.*

Obviously, this implies that the existence of a solution can be decided in polynomial time. In the sequel, let $\text{MATCH}_{\mathcal{L}}$ denote an algorithm according to Theorem 2.

Matching under side conditions

In this paper, we focus on more general matching problems, those that allow for additional *side conditions*.

Definition 3 *A subsumption condition is of the form $X \sqsubseteq^? E$ where X is a concept variable and E is a pattern; a strict subsumption condition is of the form $X \sqsubset E$ where X and E are as above. A side condition is either a subsumption condition or a strict subsumption condition. The substitution σ satisfies the side condition $X \rho E$ for $\rho \in \{\sqsubseteq, \sqsubset\}$ iff $\sigma(X) \rho \sigma(E)$.*

A matching problem under side conditions is a tuple $M := \langle C \equiv^? D, S \rangle$, where $C \equiv^? D$ is a matching problem and S

Syntax	Semantics	\mathcal{FL}_0	\mathcal{FL}_\perp	\mathcal{FL}_\neg	\mathcal{ACN}
\top	$\Delta^{\mathcal{I}}$	x	x	x	x
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	x	x	x	x
$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y: (x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	x	x	x	x
\perp	\emptyset		x	x	x
$\neg P, P \in N_C$	$\Delta^{\mathcal{I}} \setminus P^{\mathcal{I}}$			x	x
$(\geq nr), n \in \mathbb{N}$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x, y) \in r^{\mathcal{I}}\} \geq n\}$				x
$(\leq nr), n \in \mathbb{N}$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x, y) \in r^{\mathcal{I}}\} \leq n\}$				x

Table 1: Syntax and semantics of concept descriptions.

is a finite set of side conditions. If the set S contains only subsumption conditions, then M is called matching problem under subsumption conditions. The substitution σ is a solution (matcher) of M iff it is a matcher of $C \equiv^? D$ that satisfies every side condition in S .

In the next section, we will restrict the attention to matching problems under subsumption conditions. Section 4 then treats matching problems under acyclic side conditions.

In order to define matching problems under acyclic side conditions, we say that a variable X directly depends on a variable Y in S iff S contains a side condition $X \rho E$ such that Y occurs in E . If there are $n \geq 1$ variables X_1, \dots, X_n such that X_i directly depends on X_{i+1} in S ($1 \leq i \leq n-1$), then we say that X_1 depends on X_n in S . The set of side conditions S is cyclic iff there is a variable X that depends on itself in S ; otherwise, S is acyclic.

3 Matching under subsumption conditions

Let \mathcal{L} be one of the DLs $\mathcal{FL}_\perp, \mathcal{FL}_\neg, \mathcal{ACN}$. We present a polynomial time algorithm that, given an \mathcal{L} -matching problem under subsumption conditions, returns a least matcher (w.r.t. the ordering \sqsubseteq on substitutions) if the problem is solvable, and “fail” otherwise. In principle, the algorithm iterates the application of $\text{MATCH}_{\mathcal{L}}$ until a fixpoint is reached. However, the matcher computed in one step is used to modify the matching problem to be solved in the next step. Given an \mathcal{L} -matching problem under subsumption conditions $M := \langle C \equiv^? D, S \rangle$ and a substitution σ , we define

$$M_\sigma := \{C \equiv^? D\} \cup \{\sigma(X) \sqsubseteq^? E \mid X \sqsubseteq^? E \in S\}.$$

Recall that $\sigma(X) \sqsubseteq^? E$ abbreviates the matching problem $\sigma(X) \equiv^? \sigma(X) \sqcap E$. Thus M_σ is a system of \mathcal{L} -matching problems without side conditions, to which $\text{MATCH}_{\mathcal{L}}$ can be applied.

Algorithm 4 Let $M := \langle C \equiv^? D, S \rangle$ be an \mathcal{L} -matching problem under subsumption conditions. Then, the algorithm $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(M)$ works as follows:

1. $\sigma(X) := \perp$ for all variables X ;
2. If $\text{MATCH}_{\mathcal{L}}(M_\sigma)$ returns “fail”, then return “fail”; else if $\sigma \equiv \text{MATCH}_{\mathcal{L}}(M_\sigma)$, then return σ ; else $\sigma := \text{MATCH}_{\mathcal{L}}(M_\sigma)$; continue with 2.

Let σ_0 denote the substitution defined in step 1 of the algorithm, and σ_t ($t \geq 1$) the matcher computed in the t -th iteration of Step 2. Note that σ_t is undefined if $\text{MATCH}_{\mathcal{L}}$ returns

“fail” in the t -th iteration or if the algorithm has stopped before the t -th iteration.

To show that the algorithm is correct, we must show soundness, completeness, and termination, i.e., i) if the algorithm terminates and returns a substitution, then this substitution in fact solves the problem; ii) if the algorithm terminates and returns “fail”, then there indeed is no solution; and iii) the algorithm halts on every input. The following lemma proves soundness and completeness of the algorithm. The first two items establish a loop invariant.

Lemma 5 Let $M := \langle C \equiv^? D, S \rangle$ be an \mathcal{L} -matching problem under subsumption conditions.

1. If σ_t is defined and τ is a solution of M , then $\sigma_t \sqsubseteq \tau$.
2. If σ_t, σ_{t+1} are defined, then $\sigma_t \sqsubseteq \sigma_{t+1}$.
3. If $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(M)$ returns the substitution σ , then σ solves M (soundness).
4. If $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(M)$ returns “fail”, then M has no solution (completeness).

PROOF. 1. Obviously, the claim is true for σ_0 . Assume that $\sigma_t \sqsubseteq \tau$, and that σ_{t+1} is defined. To prove $\sigma_{t+1} \sqsubseteq \tau$, it is sufficient to show that τ solves M_{σ_t} since σ_{t+1} is the least solution of M_{σ_t} . Since τ solves M , we know that it solves $C \equiv^? D$ and that $\tau(X) \sqsubseteq \tau(E)$ for all $X \sqsubseteq^? E \in S$. The induction assumption $\sigma_t \sqsubseteq \tau$ implies $\sigma_t(X) \sqsubseteq \tau(X)$, and thus $\sigma_t(X) \sqsubseteq \tau(E)$, which shows that τ solves M_{σ_t} .

2. Obviously, $\sigma_0 \sqsubseteq \sigma_1$. Now assume that $\sigma_{t-1} \sqsubseteq \sigma_t$. Together with the fact that σ_t solves $M_{\sigma_{t-1}}$, this implies that σ_{t+1} solves the system $M_{\sigma_{t-1}}$. Since σ_t is the least solution of $M_{\sigma_{t-1}}$, we can conclude $\sigma_t \sqsubseteq \sigma_{t+1}$.
3. Assume that $\sigma = \sigma_t$. By definition of $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}$, $C \equiv \sigma_t(D)$. It remains to show that σ_t solves the side conditions. We know that $\sigma_t \equiv \sigma_{t+1}$ and σ_{t+1} solves M_{σ_t} . Thus, $\sigma_t(X) \sqsubseteq \sigma_{t+1}(E) \equiv \sigma_t(E)$ for every $X \sqsubseteq^? E \in S$.
4. Assume that $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(M)$ returns “fail,” and that σ_t is the last substitution computed by the algorithm. Now assume that τ solves M . As in the proof of 1. we can show that τ solves M_{σ_t} . Consequently, M_{σ_t} is solvable, and thus $\text{MATCH}(M_{\sigma_t})$ returns the least matcher of this system, in contradiction to the assumption that $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(M)$ returns “fail” in this step of the iteration. \blacksquare

Proving termination of Algorithm 4 is more involved, and the exact argument depends on the DL \mathcal{L} under consideration. Due to the space limitations, in this paper we sketch the proof for the DL \mathcal{FL}_\perp (see [Baader *et al.*, 2000] for complete proofs). The proof depends on the so-called reduced normal form of \mathcal{FL}_\perp -concept descriptions, to be introduced next.

It is easy to see that any \mathcal{FL}_\perp -concept description can be transformed into an equivalent description that is either \top or a (nonempty) conjunction of descriptions of the form $\forall r_1 \dots \forall r_m.A$, where r_1, \dots, r_m are $m \geq 0$ (not necessarily distinct) roles, and A is the bottom concept \perp or a concept name. We abbreviate $\forall r_1 \dots \forall r_m.A$ by $\forall r_1 \dots r_m.A$, where $r_1 \dots r_m$ is viewed as a word over the alphabet N_R of all role names. If $m = 0$, then this is the empty word ε , and thus $\forall \varepsilon.A$ is our “abbreviation” for A . In addition, instead of $\forall w_1.A \sqcap \dots \sqcap \forall w_\ell.A$ we write $\forall L.A$ where $L := \{w_1, \dots, w_\ell\}$ is a finite set of words over N_R ; we define $\forall \emptyset.A \equiv \top$. Using these abbreviations, any \mathcal{FL}_\perp -concept description C containing only concept names in the finite set $\mathcal{C} \subseteq N_C$ can be written as

$$C \equiv \forall U_\perp.\perp \sqcap \prod_{A \in \mathcal{C}} \forall U_A.A,$$

where U_H for $H \in \mathcal{C} \cup \{\perp\}$ are finite sets of words over N_R (called *role languages*). This representation of C will subsequently be called its *U-labeled normal form*.

As an example, consider the \mathcal{FL}_\perp -concept description $C_{ex} := \forall r.(\perp \sqcap \forall r.\perp) \sqcap \forall r.\forall s.A \sqcap \forall s.A$. Its \mathcal{FL}_0 -normal form C'_{ex} is $\forall \{r, rr\}.\perp \sqcap \forall \{rs, s\}.A$.

The role languages may contain redundant words, i.e., words that, when removed, yield equivalent concept descriptions: i) since $\forall w.\perp \sqsubseteq \forall wv.\perp$ for every $w, v \in N_R^*$, we can require U_\perp to be prefix-free, i.e., $w, wv \in U_\perp$ implies $v = \varepsilon$; and ii) since $\forall w.\perp \sqsubseteq \forall wv.A$, we can require $U_A \cap (U_\perp \cdot N_R^*) = \emptyset$. A normal form satisfying these conditions is called *reduced normal form*.

Obviously, any \mathcal{FL}_\perp -concept description can be turned into such a reduced normal form (in polynomial time). In our example, the reduced normal form of C_{ex} is $\forall \{r\}.\perp \sqcap \forall \{s\}.A$, which is obtained from C'_{ex} by removing rr from $U_\perp = \{r, rr\}$ and rs from $U_A = \{rs, s\}$. Reduced normal forms can be used to characterize equivalence of \mathcal{FL}_\perp -concept descriptions:

Lemma 6 *Assume that the \mathcal{FL}_\perp -concept descriptions C, D are given in their U- and V-labeled reduced normal forms, respectively. Then $C \equiv D$ iff $U_H = V_H$ for all $H \in \mathcal{C} \cup \{\perp\}$.*

Termination of Algorithm 4 for $\mathcal{L} = \mathcal{FL}_\perp$

Let the substitutions σ_t be defined as above. We assume that every $\sigma_t(X)$ is given in $U^{t,X}$ -labeled reduced normal form, and that C (as defined in Algorithm 4) is in U-labeled reduced normal form. Then, termination follows from the fact that every solvable matching problem under subsumption conditions has a matcher that only uses concept names already contained in the matching problem M , denoted by the set $\mathcal{C} \subseteq N_C$, and the following three properties of the languages $U^{t,X}$ and U_H for $H \in \mathcal{C} \cup \{\perp\}$. In the formulation of these properties we implicitly assume that the substitution σ_t is defined whenever we talk about one of the languages $U^{t,X}$.

1. For every variable X and every $H \in \mathcal{C} \cup \{\perp\}$, the set $U^{t,X}_H$ contains only suffixes of U_H .
2. For every word w , if $w \in U^{t,X}_H \setminus U^{t+1,X}_H$, then $w \notin U^{t',X}_H$ for any $t' > t$.
3. If σ_t and σ_{t+1} are defined and $\sigma_t \not\equiv \sigma_{t+1}$, then there exists an $H \in \mathcal{C} \cup \{\perp\}$, a variable X , and a word w such that $w \in U^{t,X}_H \setminus U^{t+1,X}_H$.

A complete proof of these properties can be found in [Baader *et al.*, 2000]. It also depends on details of the algorithm $\text{MATCH}_{\mathcal{L}}$, which we have here introduced only as a black box. Note that these properties would not hold if we did not use *reduced* normal forms.

Given these properties, it is now easy to show that the algorithm halts after a polynomial number of steps. In fact, Property 1 yields a polynomial upper bound on the size of the role languages $U^{t,X}_H$. Property 3 shows that in every step of the iteration at least one word is removed from one of these languages, and Property 2 ensures that words that have been removed cannot reappear.

To sum up, we have shown the following theorem.

Theorem 7 *Let $\mathcal{L} \in \{\mathcal{FL}_\perp, \mathcal{FL}_-, \mathcal{ALN}\}$. The algorithm $\text{MATCH}_{\mathcal{L}}^{\square}$ is a polynomial time algorithm that, given an \mathcal{L} -matching problem with subsumption conditions, returns a least matcher of this problem if it is solvable, and “fail” otherwise.*

It should be noted that the algorithm $\text{MATCH}_{\mathcal{L}}^{\square}$ does not work for $\mathcal{L} = \mathcal{FL}_0$ since this language does not allow for the bottom concept, and thus the initialization step (Step 1) is not possible. However, instead of starting with $\sigma(X) := \perp$, the algorithm can also start from the least matcher of $C \equiv^? D$. In case the side conditions do not introduce new variables (i.e., variables not contained in D), this modification works and yields a polynomial time matching algorithm. In contrast, if new variables are introduced, then we can show [Baader *et al.*, 2000] that the size of the least matcher may grow exponentially in the size of the matching problem, and that there exists an exponential time algorithm for computing such matchers. Nevertheless, the size of the substitutions for variables in D can still be bounded polynomially, and if one is only interested in substitutions for these variables, then these can still be computed in polynomial time.

4 Matching under acyclic side conditions

Matching under acyclic side conditions (i.e., strict and non-strict acyclic subsumption conditions) is more complex than matching under subsumption conditions for two reasons.

First, as already shown in [Baader *et al.*, 1999], deciding the solvability of an \mathcal{FL}_0 -matching problem under strict (and acyclic) subsumption conditions is NP-hard. It is easy to see that the same reduction works for the DLs \mathcal{FL}_\perp , \mathcal{FL}_- , and \mathcal{ALN} . Thus, assuming that $P \neq NP$, there cannot exist a polynomial time algorithm computing matchers of matching problems under general side conditions.

Second, as shown by the following example, solvable matching problems under strict subsumption conditions no

longer need to have a *least* matcher (but rather finitely many *minimal* matchers).

Example 8 Consider the \mathcal{FL}_\perp -matching problem $A_1 \sqcap \dots \sqcap A_n \equiv^? X_1 \sqcap \dots \sqcap X_n$ under the strict acyclic subsumption conditions $\{X_{i+1} \sqsubseteq^? X_i \mid 1 \leq i \leq n-1\} \cup \{X_1 \sqsubseteq^? \top\}$.

The pure matching problem enforces that each X_i must be replaced by a (possibly empty) conjunction of concept names from $\{A_1, \dots, A_n\}$. Thus, the strict subsumption conditions can only be satisfied if X_1 is replaced by one of these names, X_2 by a conjunction of this name with an additional one, etc. From this it is easy to derive that the matchers of the problem are of the following form: given a permutation $P := (p_1, \dots, p_n)$ of $(1, \dots, n)$, the substitution σ^P is defined by $\sigma^P(X_i) := A_{p_1} \sqcap \dots \sqcap A_{p_i}$ ($1 \leq i \leq n$). Thus, there are $n!$ non-equivalent matchers, and it is easy to see that each of them is minimal.

The new contribution of this section is a (non-deterministic) algorithm, $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}$, that computes matchers of \mathcal{L} -matching problems under acyclic side conditions for $\mathcal{L} \in \{\mathcal{FL}_\perp, \mathcal{FL}_-\}$. This non-deterministic algorithm matches the lower complexity bound (NP-hard) for the decision problem in the following sense. The length of every computation path of this algorithm is polynomially bounded in the size of the given matching problem. In case the problem is not solvable, every computation returns “fail”. Otherwise, the successful computation paths yield all minimal matchers.

The algorithm handling acyclic side conditions

In the following, let $M = \langle C \equiv^? D, S \rangle$ be an \mathcal{L} -matching problem ($\mathcal{L} \in \{\mathcal{FL}_\perp, \mathcal{FL}_-\}$) under acyclic side conditions. We assume that $S = \{X_1 \rho_1^? E_1, \dots, X_\ell \rho_\ell^? E_\ell\}$ for distinct variables X_1, \dots, X_ℓ and patterns E_1, \dots, E_ℓ such that E_i does not contain the variables X_i, \dots, X_ℓ . (The case where not all the left-hand side variables are distinct can be treated similarly.) We denote by S_{\sqsubseteq} the set of side conditions obtained from S by replacing every ρ_i by \sqsubseteq .

Applied to input M , the algorithm $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}$ first calls $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(\langle C \equiv^? D, S_{\sqsubseteq} \rangle)$. If this yields “fail”, then M is also unsolvable. Otherwise, the computed substitution σ solves $C \equiv^? D$, but may still violate some of the strict subsumption conditions. Starting with the violated side condition with the largest index, the algorithm tries to modify σ such that this side condition is satisfied.

Assume that $X_k \sqsubseteq^? E_k$ is this side condition. Since σ solves $X_k \sqsubseteq^? E_k$, we know that $\sigma(X_k) \equiv \sigma(E_k)$. Thus, we must either make $\sigma(X_k)$ more specific or $\sigma(E_k)$ more general. Since $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}$ computes the least solution, the first option cannot lead to a solution of the overall system. Hence, we must try the second one. The idea (which will be explained in more detail later) is that we consider the reduced normal form of $\sigma(E_k)$. We try to make $\sigma(E_k)$ more general by (non-deterministically) choosing one word from one of its role languages and by removing this word by appropriately modifying the role languages of the variables occurring in E_k . Since we want to compute minimal matchers, we make as little changes as possible in order to keep the substitution as specific as possible.

The new substitution σ' obtained this way solves $X_k \sqsubseteq^? E_k$, and since we only modified variables occurring in E_k , the side conditions with larger index are still satisfied. However, the side conditions with smaller index (even the non-strict ones) as well as the matching problem need no longer be solved by σ' . To overcome this problem, $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}$ is used to compute the least substitution that (i) solves $\langle C \equiv^? D, S_{\sqsubseteq} \rangle$, and (ii) subsumes σ' . It can be shown that the second condition (which can be expressed by a system of matching problems) makes sure that the computed substitution still solves the strict subsumption conditions from index k to ℓ . We can now continue the modification process with this substitution.

Algorithm 9 Let $M = \langle C \equiv^? D, S \rangle$ be an \mathcal{L} -matching problem under acyclic side conditions. Then, $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}$ works as follows:

1. If $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(\langle C \equiv^? D, S_{\sqsubseteq} \rangle)$ returns “fail”, then return “fail”;
2. $k := \ell$; $\sigma := \text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(\langle C \equiv^? D \rangle, S_{\sqsubseteq})$;
3. If $k = 0$, then return σ ;
If $\sigma(X_k) \rho_k \sigma(E_k)$, then continue with 5.
4. Guess modification σ' of σ for $X_k \sqsubseteq^? E_k$;
If $\sigma'(E_k) \equiv \sigma(E_k)$, then return “fail”;
 $M' := \langle \{C \equiv^? D\} \cup \{\sigma'(X_j) \sqsubseteq^? X_j \mid 1 \leq j \leq \ell\}, S_{\sqsubseteq} \rangle$;
If $\text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(M')$ returns “fail”, then return “fail”;
 $\sigma := \text{MATCH}_{\mathcal{L}}^{\sqsubseteq}(M')$
5. $k := k - 1$; continue with 3.

How to guess modifications

Here we sketch the modifications for \mathcal{FL}_\perp . (A formal definition for \mathcal{FL}_\perp and \mathcal{FL}_- can be found in [Baader *et al.*, 2000].) Recall that the goal is to make $\sigma(E_k)$ more general by (non-deterministically) choosing one word w from one of its role languages and by removing this word by appropriately modifying the role languages of the variables occurring in E_k .

We call this a *C-modification* if w is picked from a role language corresponding to some atomic concept A . In this case, removing certain words from role languages of the variables in E suffices to obtain a minimal modification.

In case of a *\perp -modification*, where w is picked from the role language corresponding to the \perp -concept, the removal of some word v in the role language of a variable implicitly removes every continuation vv' of v . To correct this effect, every word in $\{v\} \cdot N_R$ is put back whenever some v is removed. In addition, since v is also implicitly removed from role languages corresponding to atomic concepts, it is also transferred to such role languages. This ensures that the computed substitution is as specific as possible. This is vital both for the proof of correctness and to obtain all minimal solutions.

The following example illustrates in more detail how the modifications work.

Example 10 Consider the \mathcal{FL}_\perp -matching problem $A \sqcap \forall\{r, s\}.\perp \equiv^? X_1 \sqcap \forall r.X_2 \sqcap \forall r.X_3$ under the strict subsumption conditions $X_2 \sqsubseteq^? X_1$, $X_3 \sqsubseteq^? X_2$.

Executing the above algorithm, we obtain in Step 2 as initial solution σ the following substitution:

$$\{X_1 \mapsto \forall\{r, s\}.\perp \sqcap \forall\{\varepsilon\}.A, X_2 \mapsto \forall\{\varepsilon\}.\perp, X_3 \mapsto \forall\{\varepsilon\}.\perp\}.$$

The iteration begins in Step 3 by checking the second side condition, which is violated. Choosing a \perp -modification in Step 4, we must choose a word from the role language $\{\varepsilon\}$ corresponding to \perp in $\sigma(X_2) = \sigma(X_3)$. In this case, we can only pick ε . To keep the change minimal, we do not simply remove it, but rather replace it by $\{r, s\}$ in the role language corresponding to \perp in $\sigma(X_2)$. In addition, we transfer ε to the role language corresponding to A . This yields $\sigma'(X_2) = \forall\{r, s\}.\perp \sqcap \forall\{\varepsilon\}.A$. The other variables remain unchanged.

In this case, the substitution σ' itself solves the matching problem M' considered in Step 4, and thus $\text{MATCH}_{\mathcal{FL}_\perp}^{\square}(M')$ returns σ' .

In the second iteration, we find in Step 3 that the first side condition $X_2 \sqsubseteq^? X_1$ no longer holds. In Step 4, we again choose a \perp -modification, and choose the word r from the role language $\{r, s\}$ corresponding to \perp in $\sigma(X_1)$. The modification replaces r by rr, rs and adds r to the role language corresponding to A . This yields $\sigma'(X_1) := \forall\{rr, rs, s\}.\perp \sqcap \forall\{\varepsilon, r\}.A$. Again, this substitution solves M' , and thus the new value of σ is σ' .

In the next iteration we have $k = 0$, ending the iteration in Step 3. The algorithm finally returns the substitution $\{X_1 \mapsto \forall\{rr, rs, s\}.\perp \sqcap \forall\{\varepsilon, r\}.A, X_2 \mapsto \forall\{r, s\}.\perp \sqcap \forall\{\varepsilon\}.A, X_3 \mapsto \forall\{\varepsilon\}.\perp\}$.

Note that, in the first iteration step, it was not possible to apply a \mathcal{C} -modification since the role language corresponding to A was empty. In the second step, we could have applied a \mathcal{C} -modification, removing ε from the role language corresponding to A in $\sigma(X_1)$. Then, however, the system M' obtained this way would not have been solvable. In fact, it is easy to see that the two matching problems $A \sqcap \forall\{r, s\}.\perp \equiv^? X_1 \sqcap \forall r.X_2 \sqcap \forall r.X_3$ and $\forall\{r, s\}.\perp \sqsubseteq^? X_1$ occurring in M' cannot be solved simultaneously.

Soundness and completeness for \mathcal{FL}_\perp and \mathcal{FL}_- is proved in [Baader *et al.*, 2000]. It is easy to see that the length of each computation branch of the nondeterministic algorithm $\text{MATCH}_{\mathcal{L}}^{\square}$ is polynomially bounded. Because matching under strict acyclic subsumption conditions in \mathcal{FL}_\perp and \mathcal{FL}_- is known to be NP-hard, we obtain the following theorem.

Theorem 11 *Let $\mathcal{L} \in \{\mathcal{FL}_\perp, \mathcal{FL}_-\}$. Deciding the solvability of \mathcal{L} -matching problems under acyclic side conditions is an NP-complete problem.*

The algorithm described above not only decides solvability of matching problems under acyclic side conditions. It also computes matchers in case the problem is solvable. To be more precise, all minimal matchers (w.r.t. subsumption of substitutions) are computed.

5 Future work

In contrast to the algorithm for matching under subsumption conditions, the algorithm $\text{MATCH}_{\mathcal{L}}^{\square}$ presently is restricted to (i) $\mathcal{L} \in \{\mathcal{FL}_\perp, \mathcal{FL}_-\}$ and (ii) acyclic side conditions. Naturally, the next step is to overcome the first restriction by extending the result for acyclic side conditions to \mathcal{ALN} . We

strongly conjecture that an algorithm similar to $\text{MATCH}_{\mathcal{L}}^{\square}$ can be devised for this purpose.

In [Baader *et al.*, 1999], \mathcal{FL}_0 -matching problems with cyclic subsumption conditions are transformed into equivalent ones with acyclic subsumption conditions. Unfortunately, this approach cannot be extended to \mathcal{FL}_\perp in a straightforward way. Instead, in order to overcome the second restriction, one might try to adapt the general idea of the algorithm $\text{MATCH}_{\mathcal{L}}^{\square}$ to cyclic side conditions. Hence, one would first compute the solution to the analogous non-strict matching problem and then proceed by iteratively modifying the solution to satisfy the strict side conditions. Proving termination of this approach appears to be the main problem.

Apart from \mathcal{ALN} and its sublanguages, it also seems promising to consider matching problems under side conditions in \mathcal{ALE} , which extends \mathcal{FL}_- by existential restrictions. Since the algorithm $\text{MATCH}_{\mathcal{L}}^{\square}$ proposed here is defined on top of an existing algorithm for pure matching problems (with certain properties), one might similarly take advantage of the matching algorithm proposed in [Baader and Küsters, 2000] for \mathcal{ALE} -matching problems.

References

- [Baader *et al.*, 2000] F. Baader, S. Brandt, and R. Küsters. Matching under Side Conditions in Description Logics. LTCS-Report 01-02, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 2000. See <http://www-iti.informatik.rwth-aachen.de/Forschung/Reports.html>.
- [Baader and Küsters, 2000] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In *Proc. of KR2000*, pp. 261–272, Morgan Kaufmann Publishers, 2000.
- [Baader *et al.*, 1999] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.
- [Baader and Narendran, 1998] F. Baader and P. Narendran. Unification of concept terms in description logics. In *Proc. of ECAI-98*, pp. 331–335, John Wiley & Sons Ltd., 1998.
- [Borgida and Küsters, 2000] A. Borgida and R. Küsters. What's not in a name: some properties of a purely structural approach to integrating large DL knowledge bases. In *Proc. of DL2000*, number 33 in *Proc. of CEUR-WS*, 2000.
- [Borgida and McGuinness, 1996] A. Borgida and D. L. McGuinness. Asking queries about frames. In *Proc. of KR'96*, pp. 340–349, Morgan Kaufmann Publishers, 1996.
- [Borgida and Patel-Schneider, 1994] A. Borgida and P. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
- [Brachmann *et al.*, 1991] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. *Principles of Semantic Networks*, pp. 401–456. Morgan Kaufmann Publishers, 1991.
- [Donini *et al.*, 1996] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp. 193–238. CSLI Publications, 1996.
- [Horrocks, 1998] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR'98*, pp. 636–647, Morgan Kaufmann Publishers, 1998.
- [McGuinness, 1996] D. L. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Department of Computer Science, Rutgers University, October, 1996.

Computing Least Common Subsumers in $\mathcal{AL}\mathcal{EN}$

Ralf Küsters

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Germany
kuesters@ti.informatik.uni-kiel.de

Ralf Molitor*

Swiss Life
IT Research and Development Group
Switzerland
ralf.molitor@swisslife.ch

Abstract

Computing the least common subsumer (lcs) has proved to be useful in a variety of different applications. Previous work on the lcs has concentrated on description logics that either allow for number restrictions or for existential restrictions. Many applications, however, require to combine these constructors. In this work, we present an lcs algorithm for the description logic $\mathcal{AL}\mathcal{EN}$, which allows for both constructors, thereby correcting previous algorithms proposed in the literature.

1 Introduction

Computing the least common subsumer (lcs) in description logics (DLs) is an inference task first introduced by Cohen, Borgida, and Hirsh [Cohen *et al.*, 1992] for sublanguages of CLASSIC. Since then it has found several applications: as a key operation in inductive learning algorithms [Cohen and Hirsh, 1994], as a means to measure the similarity of concepts for information retrieval [Möller *et al.*, 1998], and as an operation to support the bottom-up construction of DL-knowledge bases [Baader and Küsters, 1998; Baader *et al.*, 1999]. Roughly speaking, the lcs of a set of concepts is the most specific concept description (among a possibly infinite number of concept descriptions) that subsumes all of the input descriptions, and as such allows to extract the commonalities from given concept descriptions, a task essential for all the mentioned applications.

The first lcs algorithms proposed in the literature were applicable to sublanguages of CLASSIC, more precisely, DLs that in particular allow for number restrictions [Cohen *et al.*, 1992; Cohen and Hirsh, 1994]. More recently, motivated by the bottom-up construction of knowledge bases in a chemical engineering application [Sattler, 1998; von Wedel and Marquardt, 2000], the lcs has been investigated for the DL $\mathcal{AL}\mathcal{E}$ [Baader *et al.*, 1999], which allows for existential restrictions instead of number restrictions. Although first empirical results are encouraging [Baader and Molitor, 2000], they also show that this application asks for a more expressive DL, one

*This work was carried out while the author was still at the LuFG Theoretical Computer Science, RWTH Aachen.

that allows to combine number restrictions and existential restrictions. Such a logic can, for example, be used to describe a reactor with cooling jacket and exactly two inlet valves by $\text{Reactor} \sqcap \exists \text{is-coupled-to.Cooling-Jacket} \sqcap (=2 \text{ has-inlet}) \sqcap \forall \text{has-inlet.Valve}$.

In this work, we propose an algorithm for computing the lcs of $\mathcal{AL}\mathcal{EN}$ -concept descriptions. The DL $\mathcal{AL}\mathcal{EN}$ allows for conjunction, a restricted form of negation, value restrictions, existential restrictions, and number restrictions. Similar to previous approaches [Cohen *et al.*, 1992; Baader and Küsters, 1998; Baader *et al.*, 1999], our lcs algorithm builds on a structural characterization of subsumption.

Typically, such a characterization works in two steps. First, concept descriptions are turned into a structural normal form, which makes all facts implicitly represented in the description explicit. Second, the subsumer and the subsumee, given in structural normal form, are compared syntactically. A sound and complete characterization then ensures that the structural normal form indeed contains all implied facts.

Now, given that the structural normal form of concept descriptions can be computed effectively, the lcs of concept descriptions can be obtained by first computing their structural normal forms and then extracting the “common facts” present in these normal forms.

For $\mathcal{AL}\mathcal{EN}$, however, computing the structural normal form already requires to (inductively) compute the lcs (see our running example in Section 3). Consequently, the proof of correctness of the lcs algorithm needs to be interleaved with the proof of soundness and completeness of the characterization of subsumption, making the proofs quite involved. In [Mantay, 1999], this approach is in fact pursued. However, in an attempt to avoid these interleaving proofs, Mantay made unproved assumptions concerning the existence and other properties of the lcs. Moreover, the lcs algorithm presented there is incorrect in that the computed concept description not necessarily subsumes the input descriptions.

In this work, we devise a more relaxed notion of structural normal form, which does not involve the lcs computation and therefore allows to decouple the characterization of subsumption from the lcs computation. Instead of a single $\mathcal{AL}\mathcal{EN}$ -concept description, our normal form consists of a set of $\mathcal{AL}\mathcal{EN}$ -concept descriptions, where some of the implicit facts are not made explicit.

The outline of our paper is as follows: In Section 2, we

Construct name	Syntax	Semantics
top-concept	\top	$\Delta_{\mathcal{I}}$
bottom-concept	\perp	\emptyset
concept name $P \in N_C$	P	$P^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}}$
primitive negation, $P \in N_C$	$\neg P$	$\Delta_{\mathcal{I}} \setminus P^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restrictions for $r \in N_R$	$\exists r.C$	$\{x \in \Delta_{\mathcal{I}} \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restrictions for $r \in N_R$	$\forall r.C$	$\{x \in \Delta_{\mathcal{I}} \mid \forall y : (x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
number restrictions	$(\geq nr)$	$\{x \in \Delta_{\mathcal{I}} \mid \#\{y : (x, y) \in r^{\mathcal{I}}\} \geq n\}$
for $r \in N_R, n \in \mathbb{N}$	$(\leq nr)$	$\{x \in \Delta_{\mathcal{I}} \mid \#\{y : (x, y) \in r^{\mathcal{I}}\} \leq n\}$

Table 1: Syntax and semantics of $\mathcal{AL}\mathcal{EN}$ -concept descriptions.

formally introduce the language $\mathcal{AL}\mathcal{EN}$ and the lcs operation. Section 3 contains a running example and a discussion of the main difficulties that occur when computing the lcs. In subsequent sections, this example is used to illustrate the notions introduced. Section 4 then covers the characterization of subsumption and Section 5 the lcs algorithm. Finally, in Section 6, we briefly discuss the results obtained. Due to space limitations, we cannot give all technical details. These details as well as complete proofs can be found in [Küstners and Molitor, 2000].

2 Preliminaries

Concept descriptions are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. In this work, we consider the DL $\mathcal{AL}\mathcal{EN}$, i.e., concept descriptions built from the constructors shown in Table 1, subsequently called *$\mathcal{AL}\mathcal{EN}$ -concept descriptions*.

The semantics of a concept description is defined in terms of an *interpretation* $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta_{\mathcal{I}}$ of \mathcal{I} is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $P \in N_C$ to a set $P^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

One of the most important traditional inference services provided by DL systems is computing the subsumption hierarchy. The concept description C is *subsumed* by the description D ($C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations \mathcal{I} . The concept descriptions C and D are *equivalent* ($C \equiv D$) iff they subsume each other.

In this paper, we are interested in the computation of least common subsumers.

Definition 1 *Given $\mathcal{AL}\mathcal{EN}$ -concept descriptions C_1, \dots, C_n , $n \geq 2$, the $\mathcal{AL}\mathcal{EN}$ -concept description C is the least common subsumer (lcs) of C_1, \dots, C_n ($C = \text{lcs}(C_1, \dots, C_n)$ for short) iff (i) $C_i \sqsubseteq C$ for all $1 \leq i \leq n$, and (ii) C is the least concept description with this property, i.e., if C' satisfies $C_i \sqsubseteq C'$ for all $1 \leq i \leq n$, then $C \sqsubseteq C'$.*

Depending on the DL under consideration, the lcs of two or more descriptions need not always exist, but if it exists, then it is unique up to equivalence. The main contribution of this

paper is to show that in $\mathcal{AL}\mathcal{EN}$ the lcs always exists and that it can be computed effectively.

For the sake of simplicity, we consider $\mathcal{AL}\mathcal{EN}$ -concept descriptions over a set N_C of concept names and assume N_R to be the singleton $\{r\}$. However, all definitions and results can easily be generalized to arbitrary sets of role names. Furthermore, w.l.o.g., we assume all $\mathcal{AL}\mathcal{EN}$ -concept descriptions to be in the following *normal form*: Each conjunction in an $\mathcal{AL}\mathcal{EN}$ -concept description contains

1. at most one number restriction of the form $(\geq nr)$ (this is w.l.o.g. due to $(\geq m r) \sqcap (\geq n r) \equiv (\geq n r)$ if $n \geq m$);
2. at most one number restriction of the form $(\leq nr)$ (this is w.l.o.g. due to $(\leq m r) \sqcap (\leq n r) \equiv (\leq n r)$ if $n \leq m$);
3. at most one value restriction of the form $\forall r.C$ (this is w.l.o.g. due to $\forall r.C \sqcap \forall r.D \equiv \forall r.(C \sqcap D)$).

3 Running example

In order to highlight the main problems to be solved in the structural characterization of subsumption and the computation of the lcs in $\mathcal{AL}\mathcal{EN}$, we will use the following $\mathcal{AL}\mathcal{EN}$ -concept descriptions:

$$C_{ex} := \exists r.(P \sqcap A_1) \sqcap \exists r.(P \sqcap A_2) \sqcap \exists r.(\neg P \sqcap A_1) \sqcap \exists r.(Q \sqcap A_3) \sqcap \exists r.(\neg Q \sqcap A_3) \sqcap (\leq 2 r), \text{ and}$$

$$D_{ex} := (\geq 3 r) \sqcap \forall r.(A_1 \sqcap A_2 \sqcap A_3).$$

The key point in the characterization of subsumption and the lcs computation is to describe the “non-trivial” concept descriptions, say C' , subsuming a given concept description, say C , where “non-trivial” means that C' does not occur as a conjunct on the top-level of C . Subsequently, these concept descriptions are called *induced*. It suffices to only consider induced concept descriptions that are minimal w.r.t. subsumption.

In what follows, we describe the concept descriptions induced by C_{ex} and D_{ex} . It turns out that some of the concept descriptions induced by C_{ex} correspond to the lcs of certain subdescriptions in C_{ex} . As we will see, given the induced concept descriptions of C_{ex} and D_{ex} , it is easy to determine the lcs of C_{ex} and D_{ex} . We start with the concept descriptions induced by C_{ex} .

Number restrictions: Because of the existential restrictions on the top-level of C_{ex} , e.g. $\exists r.(P \sqcap A_1)$ and $\exists r.(\neg P \sqcap A_1)$, we know $C_{ex} \sqsubseteq (\geq 2r)$, i.e., $(\geq 2r)$ is induced by C_{ex} . Conversely, there is no induced \leq -restriction, i.e., the most specific \leq -restriction subsuming C_{ex} is the \leq -restriction explicitly present on the top-level of C_{ex} .

Existential restrictions: Due to the \leq -restriction ($\leq 2r$) on the top-level of C_{ex} , each instance of C_{ex} has at most two r -successors. Consequently, some existential restrictions have to be “merged” to a single existential restriction, where “merging” means conjoining the concept descriptions occurring in the existential restrictions. For C_{ex} , there are several ways to merge the five existential restrictions on the top-level of C_{ex} into two existential restrictions. The merging process gives rise to new (derived) concept descriptions, where the only consistent ones are:

$$\begin{aligned} C_{ex}^1 &:= \exists r.(P \sqcap Q \sqcap A_1 \sqcap A_2 \sqcap A_3) \sqcap \\ &\quad \exists r.(\neg P \sqcap \neg Q \sqcap A_1 \sqcap A_3) \sqcap (\leq 2r), \text{ and} \\ C_{ex}^2 &:= \exists r.(P \sqcap \neg Q \sqcap A_1 \sqcap A_2 \sqcap A_3) \sqcap \\ &\quad \exists r.(\neg P \sqcap Q \sqcap A_1 \sqcap A_3) \sqcap (\leq 2r). \end{aligned}$$

It is clear that $C_{ex} \equiv C_{ex}^1 \sqcup C_{ex}^2$. The existential restrictions $\exists r.(P \sqcap A_1 \sqcap A_2 \sqcap A_3)$, $\exists r.(\neg P \sqcap A_1 \sqcap A_3)$, $\exists r.(Q \sqcap A_1 \sqcap A_3)$, and $\exists r.(\neg Q \sqcap A_1 \sqcap A_3)$ subsume both C_{ex}^1 and C_{ex}^2 . From this it can be concluded that these restrictions are induced by C_{ex} .

As we will see, the induced existential restrictions can be obtained by picking one existential restriction from each of the (consistent) derived concept descriptions and applying the lcs operation to them. In our example, we have, for instance, $P \sqcap A_1 \sqcap A_2 \sqcap A_3 \equiv lcs(P \sqcap Q \sqcap A_1 \sqcap A_2 \sqcap A_3, P \sqcap \neg Q \sqcap A_1 \sqcap A_2 \sqcap A_3)$. However, as explained below, our characterization of subsumption avoids to explicitly use the lcs by employing the fact that

$$lcs(C_1, \dots, C_n) \sqsubseteq D \text{ iff } C_i \sqsubseteq D \text{ for all } 1 \leq i \leq n. \quad (1)$$

Value restrictions: In view of $C_{ex} \equiv C_{ex}^1 \sqcup C_{ex}^2$, it not only follows that every instance of C_{ex} has exactly two r -successors but that these r -successors must satisfy the existential restrictions given in C_{ex}^1 and C_{ex}^2 . In either case, all r -successors belong to $A_1 \sqcap A_3$, and thus, the value restriction $\forall r.(A_1 \sqcap A_3)$ is induced by C_{ex} .

There are two things that should be pointed out here: First, note that there is an induced value restriction only if the number of successors induced by existential restrictions coincides with the number in the \leq -restriction, because only in this case, we have “full” information about *all* r -successors of an instance of C . For example, if we consider the concept description $E_{ex} := (\leq 2r) \sqcap \exists r.(A_1 \sqcap A_2) \sqcap \exists r.(A_1 \sqcap A_3)$, no value restriction is induced.¹

Second, if $\forall r.C'$ is the most specific value restriction induced by C , then C' corresponds to the lcs of all concept descriptions occurring in the merged existential restrictions. In

¹The structural subsumption algorithm introduced in [Mantay, 1999], however, computes $\forall r.A_1$ as a value restriction induced by E_{ex} and thus, is incorrect. For the same reason, the lcs-algorithm presented in [Mantay, 1999] is incorrect: although the lcs of E_{ex} and $\forall r.A_1$ is \top , the algorithm returns $\forall r.A_1$.

the example, $A_1 \sqcap A_3 \equiv lcs(P \sqcap Q \sqcap A_1 \sqcap A_2 \sqcap A_3, \neg P \sqcap \neg Q \sqcap A_1 \sqcap A_2 \sqcap A_3, P \sqcap \neg Q \sqcap A_1 \sqcap A_2 \sqcap A_3, \neg P \sqcap Q \sqcap A_1 \sqcap A_2 \sqcap A_3)$. Again, using the equivalence (1), we avoid to explicitly compute the lcs in the characterization of subsumption.

It is easy to see that the only (minimal) concept description induced by D_{ex} is $\exists r.(A_1 \sqcap A_2 \sqcap A_3)$.

Now, given the concept descriptions induced by C_{ex} and D_{ex} it is not hard to verify that the lcs of C_{ex} and D_{ex} can be stated as

$$(\geq 2r) \sqcap \forall r.(A_1 \sqcap A_3) \sqcap \exists r.(A_1 \sqcap A_2 \sqcap A_3).$$

4 Structural characterization of subsumption

In what follows, let C, D be $\mathcal{AL}\mathcal{EN}$ -concept descriptions in the normal form introduced in Section 2. Since both $C \equiv \perp$ and $D \equiv \top$ trivially imply $C \sqsubseteq D$, our characterization of subsumption explicitly checks these equivalences. Otherwise, roughly speaking, each conjunct in D , i.e., each (negated) concept name, number restriction, existential restriction, and value restriction occurring on the top-level of D , is compared with the corresponding conjuncts in C . For the existential restrictions and value restrictions, however, it will be necessary to resort to the concept descriptions derived from C by merging existential restrictions (C_{ex}^1 and C_{ex}^2 in the example). If all the comparisons have succeeded, it follows $C \sqsubseteq D$. In the remainder of this section, the structural comparison between C and D is further explained. Finally, Theorem 2 establishes the complete characterization of subsumption.

We first need some notation to access the different parts of the concept descriptions:

- $\text{prim}(C)$ denotes the set of all (negated) concept names occurring on the top-level of C ;
- $\text{min}_r(C) := \max\{k \mid C \sqsubseteq (\geq k r)\}$;
- $\text{max}_r(C) := \min\{k \mid C \sqsubseteq (\leq k r)\}$;
- if there exists a value restriction of the form $\forall r.C'$ on the top-level of C , then $\text{val}_r(C) := C'$; otherwise, $\text{val}_r(C) := \top$;
- $\text{exr}_r(C) := \{C' \mid \text{there exists } \exists r.C' \text{ on the top-level of } C\}$.

Note that $\text{min}_r(C)$ is always finite, whereas $\text{max}_r(C)$ may be ∞ , namely if there exists no positive integer k such that $C \sqsubseteq (\leq k r)$. Although these values need not be explicitly present in number restrictions of C , they can be computed in polynomial time in the size of C using an oracle for subsumption of $\mathcal{AL}\mathcal{EN}$ -concept descriptions [Küsters and Molitor, 2000]. In our example, we get $\text{min}_r(C_{ex}) = 2$, $\text{max}_r(C_{ex}) = 2$, $\text{min}_r(D_{ex}) = 3$, and $\text{max}_r(D_{ex}) = \infty$.

The structural comparison between the different parts of C and D can now be stated as follows (assuming $C \not\equiv \perp$ and $D \not\equiv \top$):

(Negated) concept names and number restrictions: (cf. Theorem 2, 1.–3.) In order for $C \sqsubseteq D$ to hold, it is obvious that the following conditions need to be satisfied: $\text{prim}(D) \subseteq \text{prim}(C)$, $\text{max}_r(C) \leq \text{max}_r(D)$, $\text{min}_r(C) \geq \text{min}_r(D)$. Otherwise, it is easy to construct a counter-model for $C \sqsubseteq D$.

Existential restrictions: (cf. Theorem 2, 4.) Given $D' \in \text{exr}_r(D)$, then for $C \sqsubseteq D$ to hold one at first might expect that there need to exist a $C' \in \text{exr}_r(C)$ with $\text{val}_r(C) \sqcap C' \sqsubseteq D'$. Although this works for $\mathcal{AL}\mathcal{E}\mathcal{N}$ -concept descriptions, it fails for $\mathcal{AL}\mathcal{E}\mathcal{N}$, as $C_{ex} \sqsubseteq \exists r.(P \sqcap A_1 \sqcap A_2 \sqcap A_3)$ shows (see Section 3). The reason is that in $\mathcal{AL}\mathcal{E}\mathcal{N}$ \leq -restrictions may require to merge existential restrictions, yielding new (implicit) existential restrictions. To deal with this phenomenon, sets of derived concept descriptions are considered with certain existential restrictions merged (see C_{ex}^1 and C_{ex}^2 in our running example).

The result of merging existential restrictions is described by so-called *existential mappings*

$$\alpha : \{1, \dots, n\} \longrightarrow 2^{\{1, \dots, m\}},$$

where $n := \min\{\max_r(C), |\text{exr}_r(C)|\}$ and $m := |\text{exr}_r(C)|$. We require α to obey the following conditions:

1. $\alpha(i) \neq \emptyset$ for all $1 \leq i \leq n$;
2. $\bigcup_{1 \leq i \leq n} \alpha(i) = \{1, \dots, m\}$ and $\alpha(i) \cap \alpha(j) = \emptyset$ for all $1 \leq i < j \leq n$.
3. $\bigcap_{j \in \alpha(i)} C_j \sqcap \text{val}_r(C) \not\sqsubseteq \perp$ for all $1 \leq i \leq n$.

Although the first two conditions are not essential for soundness and completeness of the characterization of subsumption, they reduce the number of existential mappings that need to be considered.

Given $\text{exr}_r(C) = \{C_1, \dots, C_m\}$, α yields an $\mathcal{AL}\mathcal{E}\mathcal{N}$ -concept description C^α obtained from C by substituting all existential restrictions on the top-level of C with

$$\bigcap_{1 \leq i \leq n} \exists r. \bigcap_{j \in \alpha(i)} C_j.$$

The set of all existential mappings on C satisfying the conditions (1)–(3) is denoted by $\Gamma_r(C)$, where $\Gamma_r(C) := \emptyset$ if $\text{exr}_r(C) = \emptyset$. It is in fact sufficient to consider $\Gamma_r(C)$ modulo permutations, i.e., modulo the equivalence

$$\alpha \cong \alpha' \quad \text{iff} \quad \text{there exists a permutation } \pi \text{ on } \{1, \dots, n\} \\ \text{s.t. } \alpha(i) = \alpha'(\pi(i)) \text{ for all } 1 \leq i \leq n.$$

In the sequel, for the sake of simplicity, we stay with $\alpha \in \Gamma_r(C)$ instead of $[\alpha]_{\cong} \in \Gamma_r(C)/_{\cong}$, though.

In our running example, let $\text{exr}_r(C_{ex}) = \{C_{ex,1}, \dots, C_{ex,5}\}$ with $C_{ex,1} = P \sqcap A_1$, $C_{ex,2} = P \sqcap A_2$, etc. Then, $\Gamma_r(C_{ex})$ consists of the two mappings

$$\begin{aligned} \alpha_1 &= \{1 \mapsto \{1, 2, 4\}, 2 \mapsto \{3, 5\}\}; \\ \alpha_2 &= \{1 \mapsto \{1, 2, 5\}, 2 \mapsto \{3, 4\}\}, \end{aligned}$$

and it is $C_{ex}^{\alpha_i} = C_{ex}^i$, $i = 1, 2$ (see Section 3).

For the characterization of subsumption, we will use the following notation:

$$\text{exr}_r(C)^\alpha := \{ \bigcap_{j \in \alpha(i)} C_j \mid 1 \leq i \leq n \}.$$

Now, in case $\text{exr}_r(C) \neq \emptyset$, $C \sqsubseteq D$ implies that for each $D' \in \text{exr}_r(D)$ the following holds: for each $\alpha \in \Gamma_r(C)$, there exists $C'_\alpha \in \text{exr}_r(C)^\alpha$ such that $C'_\alpha \sqcap \text{val}_r(C) \sqsubseteq D'$. This is what is stated in Theorem 2, 4. Note that, using the equivalence (1), and provided that the lcs of $\mathcal{AL}\mathcal{E}\mathcal{N}$ -concept

descriptions always exists, this condition could be rewritten as: there exists a set $M := \{C'_\alpha \in \text{exr}_r(C)^\alpha \mid \alpha \in \Gamma_r(C)\}$ with $\text{lcs}(\{C' \sqcap \text{val}_r(C) \mid C' \in M\}) \sqsubseteq D'$. However, since the existence of the lcs is not guaranteed a priori, the lcs cannot be used in the structural characterization of subsumption, unless the characterization and the lcs computation are interleaved.

If $\text{exr}_r(C) = \emptyset$, then for all $D' \in \text{exr}_r(D)$ it must hold, $\min_r(C) \geq 1$ and $\text{val}_r(C) \sqsubseteq D'$.

In our running example, $C_{ex} \sqsubseteq \exists r.(P \sqcap A_1 \sqcap A_2 \sqcap A_3)$ illustrates the case where $\text{exr}_r(C) \neq \emptyset$ and $D_{ex} \sqsubseteq \exists r.(A_1 \sqcap A_2 \sqcap A_3)$ illustrates $\text{exr}_r(C) = \emptyset$.

Value restrictions: (cf. Theorem 2, 5.) Value restrictions can only be induced for two reasons. First, if $\max_r(C) = 0$, then $C \sqsubseteq \forall r.\perp$, and thus, $C \sqsubseteq \forall r.C'$ for all concept descriptions C' .

Second, the merging of existential restrictions may induce value restrictions. In contrast to induced existential restrictions, however, one further needs to take care of \geq -restrictions induced by “incompatible” existential restrictions:

$$\kappa_r(C) := \min_r(\forall r.\text{val}_r(C) \sqcap \bigcap_{C' \in \text{exr}_r(C)} \exists r.C').$$

If $\text{exr}_r(C) = \emptyset$, we define $\kappa_r(C) := 0$. In our example, $\kappa_r(C_{ex}) = 2$, $\kappa_r(D_{ex}) = 0$, and $\kappa_r(E_{ex}) = 1$.

Now, only if $\kappa_r(C) = \max_r(C)$, value restrictions can be induced, since only then we “know” all the r -successors of instances of C . In our example, $\kappa_r(C_{ex}) = \max_r(C_{ex}) = 2$, which accounts for $C_{ex} \sqsubseteq \forall r.(A_1 \sqcap A_3)$. Conversely, $E_{ex} \not\sqsubseteq \forall r.A_1$ since $1 = \kappa_r(E_{ex}) < \max_r(E_{ex}) = 2$.

To formally state the comparison between value restrictions of C and D , we use the following notation:

$$\text{exr}_r(C)^* := \bigcup_{\alpha \in \Gamma_r(C)} \text{exr}_r(C)^\alpha.$$

One can show that $C \sqsubseteq D$ implies $\text{val}_r(C) \sqsubseteq \text{val}_r(D)$ provided that $\kappa_r(C) < \max_r(C)$. In case $0 < \kappa_r(C) = \max_r(C)$, however, it suffices if the value restriction of D satisfies $\text{val}_r(C) \sqcap C' \sqsubseteq \text{val}_r(D)$ for all $C' \in \text{exr}_r(C)^*$. Again, using the equivalence (1), and provided that the lcs of $\mathcal{AL}\mathcal{E}\mathcal{N}$ -concept descriptions always exists, this condition can be restated as $\text{val}_r(C) \sqcap \text{lcs}(\text{exr}_r(C)^*) \sqsubseteq D'$. For reasons already mentioned, we have not employed this variant.

We are now ready to state the structural characterization of subsumption in $\mathcal{AL}\mathcal{E}\mathcal{N}$.

Theorem 2 [Küstners and Molitor, 2000] *Let C, D be two $\mathcal{AL}\mathcal{E}\mathcal{N}$ -concept descriptions with $\text{exr}_r(C) = \{C_1, \dots, C_m\}$. Then $C \sqsubseteq D$ iff $C \equiv \perp$, $D \equiv \top$, or the following holds:*

1. $\text{prim}(D) \subseteq \text{prim}(C)$;
2. $\max_r(C) \leq \max_r(D)$;
3. $\min_r(C) \geq \min_r(D)$;
4. for all $D' \in \text{exr}_r(D)$ it holds that
 - (a) $\text{exr}_r(C) = \emptyset$, $\min_r(C) \geq 1$, and $\text{val}_r(C) \sqsubseteq D'$; or
 - (b) $\text{exr}_r(C) \neq \emptyset$ and for each $\alpha \in \Gamma_r(C)$, there exists $C' \in \text{exr}_r(C)^\alpha$ such that $C' \sqcap \text{val}_r(C) \sqsubseteq D'$; and

Let C, D be two $\mathcal{AL}\mathcal{EN}$ -concept descriptions.

If $C \sqsubseteq D$, then $\mathbf{c-lcs}(C, D) := D$, and if $D \sqsubseteq C$, then $\mathbf{c-lcs}(C, D) := C$.

Otherwise, let $\Gamma_r(C) = \{\alpha_1^C, \dots, \alpha_{n(C)}^C\}$ and $\Gamma_r(D) = \{\alpha_1^D, \dots, \alpha_{n(D)}^D\}$, and define for $E \in \{C, D\}$

$$\mathcal{C}_r(E) := \begin{cases} \{\mathbf{c-lcs}(\{E_1 \sqcap \text{val}_r(E), \dots, E_{n(E)} \sqcap \text{val}_r(E)\}) \mid E_i \in \text{exr}_r(E)^{\alpha_i^E}, 1 \leq i \leq n(E)\}, & \Gamma_r(E) \neq \emptyset, \\ \{\text{val}_r(E)\}, & \Gamma_r(E) = \emptyset \wedge \min_r(E) \geq 1, \\ \emptyset, & \min_r(E) = 0; \end{cases}$$

$$E_r^* := \begin{cases} \text{val}_r(E), & 0 \leq \kappa_r(E) < \max_r(E), \\ \perp, & \max_r(E) = 0, \\ \mathbf{c-lcs}(\{\text{val}_r(E) \sqcap E' \mid E' \in \text{exr}_r(E)^*\}), & 0 < \kappa_r(E) = \max_r(E). \end{cases}$$

Define

$$\mathbf{c-lcs}(C, D) := \bigcap_{Q \in \text{prim}(C) \cap \text{prim}(D)} Q \sqcap (\leq \max\{\max_r(C), \max_r(D)\} r) \sqcap (\geq \min\{\min_r(C), \min_r(D)\} r) \sqcap \bigcap_{C' \in \mathcal{C}_r(C), D' \in \mathcal{C}_r(D)} \exists r. \mathbf{c-lcs}(C', D') \sqcap \forall r. \mathbf{c-lcs}(C_r^*, D_r^*),$$

where

- $(\leq \max\{\max_r(C), \max_r(D)\} r)$ is omitted if $\max_r(C) = \infty$ or $\max_r(D) = \infty$, and
- $\bigcap_{C' \in \mathcal{C}_r(C), D' \in \mathcal{C}_r(D)} \exists r. \mathbf{c-lcs}(C', D') := \top$ if $\mathcal{C}_r(C) = \emptyset$ or $\mathcal{C}_r(D) = \emptyset$.

Figure 1: The recursive computation of the lcs in $\mathcal{AL}\mathcal{EN}$.

5. if $\text{val}_r(D) \neq \top$, then

- (a) $\max_r(C) = 0$; or
- (b) $\kappa_r(C) < \max_r(C)$ and $\text{val}_r(C) \sqsubseteq \text{val}_r(D)$; or
- (c) $0 < \kappa_r(C) = \max_r(C)$ and $\text{val}_r(C) \sqcap C' \sqsubseteq \text{val}_r(D)$ for all $C' \in \text{exr}_r(C)^*$.

As known from the literature, [Hemaspaandra, 1999], testing subsumption of $\mathcal{AL}\mathcal{EN}$ -concept descriptions is PSPACE-complete. However, our characterization is not intended to serve as a basis for a polynomial-space subsumption algorithm. It rather yields the formal basis for the lcs algorithm presented in the next section.

5 Computing the lcs in $\mathcal{AL}\mathcal{EN}$

The recursive algorithm computing the lcs of two $\mathcal{AL}\mathcal{EN}$ -concept descriptions C and D is depicted in Figure 1. For a set \mathcal{C} of $\mathcal{AL}\mathcal{EN}$ -concept descriptions, $\mathbf{c-lcs}(\mathcal{C})$ is computed by iteratively applying the binary $\mathbf{c-lcs}$ -operation, i.e., $\mathbf{c-lcs}(\{C_1, \dots, C_n\}) := \mathbf{c-lcs}(C_1, \mathbf{c-lcs}(C_2, \dots, \mathbf{c-lcs}(C_{n-1}, C_n) \dots))$ for $n > 2$.

Since the role depth of C and D is finite, the recursive computation of $\mathbf{c-lcs}(C, D)$ always terminates. The following theorem states correctness of the lcs algorithm depicted in Figure 1. As an immediate consequence, we obtain that the lcs of two $\mathcal{AL}\mathcal{EN}$ -concept descriptions always exists. The theorem is proved by induction on the role depth of $\mathbf{c-lcs}(C, D)$ and makes heavy use of the structural characterization of subsumption given in Theorem 2.

Theorem 3 [Küsters and Molitor, 2000] *Let C, D be $\mathcal{AL}\mathcal{EN}$ -concept descriptions. Then, $\mathbf{c-lcs}(C, D) \equiv \text{lcs}(C, D)$.*

In the remainder of this section, we illustrate the definition of $\mathbf{c-lcs}(C, D)$ in case that $C \not\sqsubseteq D$ and $D \not\sqsubseteq C$ (since the special cases $C \sqsubseteq D$, $D \sqsubseteq C$ are trivial), using our running example introduced in Section 3. The conjuncts occurring on the top-level of $\mathbf{c-lcs}(C, D)$ can, as before, be divided into three parts, namely (1) (negated) concept names and number restrictions, (2) the existential restrictions, and (3) the value restriction. These conjuncts are defined in such a way that

- (a) the conditions 1.–5. in Theorem 2 for $C \sqsubseteq \mathbf{c-lcs}(C, D)$ and $D \sqsubseteq \mathbf{c-lcs}(C, D)$ are satisfied, and
- (b) $\mathbf{c-lcs}(C, D)$ is the least concept description (w.r.t. \sqsubseteq) satisfying (a).

For the conditions 1.–3., this is quite obvious, since, for $E \in \{C, D\}$, we obtain

- $\text{prim}(\mathbf{c-lcs}(C, D)) = \text{prim}(C) \cap \text{prim}(D) \subseteq \text{prim}(E)$,
- $\min_r(\mathbf{c-lcs}(C, D)) = \min\{\min_r(C), \min_r(D)\} \leq \min_r(E)$, and
- $\max_r(\mathbf{c-lcs}(C, D)) = \max\{\max_r(C), \max_r(D)\} \geq \max_r(E)$.

In our running example, $\text{prim}(\mathbf{c-lcs}(C_{ex}, D_{ex})) = \emptyset$, $\min_r(\mathbf{c-lcs}(C_{ex}, D_{ex})) = 2$, and $\max_r(\mathbf{c-lcs}(C_{ex}, D_{ex})) = \infty$.

For existential and value restrictions things are more complicated. Let us first consider the definition of $\text{exr}_r(\mathbf{c-lcs}(C, D))$, i.e., the existential restrictions obtained from the sets $\mathcal{C}_r(C)$ and $\mathcal{C}_r(D)$. Roughly speaking, if $\Gamma_r(E) \neq \emptyset$, then $\mathcal{C}_r(E)$ contains all (minimal) concept descriptions occurring in an existential restriction induced by E for $E \in \{C, D\}$. Each such concept description is obtained

as the recursively computed lcs of a set of concept descriptions consisting of one concept description from $\text{ex}_r(E)^\alpha$ (conjoined with $\text{val}_r(E)$) for each $\alpha \in \Gamma_r(E)$. In our running example, each pair of concept descriptions occurring in the existential restrictions on the top-level of C_{ex}^1 and C_{ex}^2 , respectively, yields such an lcs, and thus $\mathcal{C}_r(C_{ex}) = \{P \sqcap A_1 \sqcap A_2 \sqcap A_3, Q \sqcap A_1 \sqcap A_3, \neg Q \sqcap A_1 \sqcap A_3, \neg P \sqcap A_1 \sqcap A_3\}$ (see Section 3).

If $\min_r(E) = 1$ and $\Gamma_r(E) = \emptyset$, we set $\mathcal{C}_r(E) = \{\text{val}_r(E)\}$ since then $\text{ex}_r(E) = \emptyset$, and $\exists r.\text{val}_r(E)$ is the unique minimal existential restriction induced by E . This case is illustrated by D_{ex} in our running example, where $\min_r(D_{ex}) = 3$, $\text{ex}_r(D_{ex}) = \emptyset$, and $\mathcal{C}_r(D_{ex}) = \{A_1 \sqcap A_2 \sqcap A_3\}$.

Finally, if $\min_r(E) = 0$, i.e., there exists no existential restriction subsuming E , then obviously, no existential restriction can occur on the top-level of a common subsumer of C and D . Therefore, we set $\mathcal{C}_r(E) := \emptyset$.

Given $\mathcal{C}_r(C)$ and $\mathcal{C}_r(D)$, the lcs of each pair $C' \in \mathcal{C}_r(C)$ and $D' \in \mathcal{C}_r(D)$ gives rise to an existential restriction on the top-level of the lcs of C and D . In our example, we obtain $\text{ex}_r(\mathbf{c}\text{-lcs}(C_{ex}, D_{ex})) = \{A_1 \sqcap A_2 \sqcap A_3, A_1 \sqcap A_3\}$, giving rise to the existential restrictions $\exists r.(A_1 \sqcap A_2 \sqcap A_3)$ and $\exists r.(A_1 \sqcap A_3)$ (where the latter one can be omitted).

It remains to comment on $\forall r.\mathbf{c}\text{-lcs}(C_r^*, D_r^*)$. Intuitively, $\forall r.E_r^*$ is the most specific value restriction subsuming E for $E \in \{C, D\}$. (Thus, $\forall r.\mathbf{c}\text{-lcs}(C_r^*, D_r^*)$ is the most specific value restriction subsuming both C and D .) If E has no induced value restriction, E_r^* coincides with $\text{val}_r(E)$. This is the case for D_{ex} , where $(D_{ex})_r^* = \text{val}_r(D_{ex}) = A_1 \sqcap A_2 \sqcap A_3$. If $\max_r(E) = 0$, the fact that $E \sqsubseteq \forall r.\perp$ is made explicit by defining $E_r^* := \perp$. Finally, if $0 < \kappa_r(E) = \max_r(E)$, then the induced value restriction is again made explicit by recursively computing the lcs of the merged existential restrictions (each conjoined with $\text{val}_r(E)$). In our running example, this case is illustrated by C_{ex} : there, $\text{val}_r(C_{ex}) = \top$, but since $\kappa_r(C_{ex}) = \max_r(C_{ex}) = 2$, we obtain $(C_{ex})_r^* = A_1 \sqcap A_3$ which is the recursively computed lcs of $\text{ex}_r(C_{ex})^* = \text{ex}_r(C_{ex}^1) \cup \text{ex}_r(C_{ex}^2)$ (see Section 3).

6 Conclusion and future work

We have presented an algorithm for computing the lcs in $\mathcal{AL}\mathcal{EN}$. Its proof of correctness is based on the structural characterization of subsumption introduced in Section 4. In this characterization, we avoided to explicitly use the lcs in order to be able to decouple the characterization of subsumption and the lcs computation. Interleaving these two tasks caused previous work on the lcs in $\mathcal{AL}\mathcal{EN}$ to be incorrect.

Due to the interaction between number restrictions and existential restrictions the characterization of subsumption and the lcs computation became much more involved than for the sublanguages $\mathcal{AL}\mathcal{E}$ [Baader *et al.*, 1999] and $\mathcal{AL}\mathcal{N}$ [Cohen and Hirsh, 1994; Baader and Küsters, 1998]. As an immediate consequence of results shown in [Baader *et al.*, 1999] for $\mathcal{AL}\mathcal{E}$, we obtain an exponential lower bound for the size of the lcs of two $\mathcal{AL}\mathcal{EN}$ -concept descriptions. It is, however, not known whether this bound is tight. The lcs algorithm presented in Section 5 takes double exponential time [Küsters

and Molitor, 2000]. Nevertheless, since a prototype implementation of the exponential-time lcs algorithm for $\mathcal{AL}\mathcal{E}$ behaves quite well in the chemical process engineering application [Baader and Molitor, 2000], the hope is that the lcs algorithm for $\mathcal{AL}\mathcal{EN}$ proposed here will also work in realistic application situations. For evaluation, continuing the work on $\mathcal{AL}\mathcal{E}$, a prototype of the algorithm for $\mathcal{AL}\mathcal{EN}$ will be implemented using the DL-system FaCT [Horrocks, 1998] for deciding subsumption in $\mathcal{AL}\mathcal{EN}$, which then is to be applied to the chemical engineering knowledge base.

References

- [Baader and Küsters, 1998] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{AL}\mathcal{N}$ -concept descriptions. In *Proc. of KI'98*, LNAI 1504. 1998.
- [Baader and Molitor, 2000] F. Baader and R. Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In *Proc. of ICCS2000*, LNAI 1867. 2000.
- [Baader *et al.*, 1999] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of IJCAI'99*. Morgan Kaufmann, 1999.
- [Cohen and Hirsh, 1994] W.W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *Proc. of KR'94*. Morgan Kaufmann, 1994.
- [Cohen *et al.*, 1992] W.W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In *Proc. of AAAI'92*. MIT Press, 1992.
- [Hemaspaandra, 1999] E. Hemaspaandra. The complexity of pure man's logic. In *Essays dedicated to Johan van Benthem on the occasion of his 50th birthday*, 1999.
- [Horrocks, 1998] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR'98*. Morgan Kaufmann, 1998.
- [Küsters and Molitor, 2000] Computing least common subsumers in $\mathcal{AL}\mathcal{EN}$. LTCS-Report 00-07, RWTH Aachen, Germany, 2000. See <http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html>.
- [Mantay, 1999] T. Mantay. Computing least common subsumers in expressive description logics. In *Proc. of AI'99*, LNAI 1747. 1999.
- [Möller *et al.*, 1998] R. Möller, V. Haaslev, and B. Neumann. Semantics-based information retrieval. In *Proc. of IT&KNOWS*. 1998.
- [Sattler, 1998] U. Sattler. *Terminological knowledge representation systems in a process engineering application*. PhD thesis, RWTH Aachen, 1998.
- [von Wedel and Marquardt, 2000] L. von Wedel and W. Marquardt. ROME: A repository to support the integration of models over the lifecycle of model-based engineering processes. In *Proc. of ESCAPE-10*, 2000.

FCA-MERGE: Bottom-Up Merging of Ontologies

Gerd Stumme

Institute for Applied Computer Science and
Formal Description Methods (AIFB)
University of Karlsruhe
D-76128 Karlsruhe, Germany
www.aifb.uni-karlsruhe.de/WBS/gst

Alexander Maedche

FZI Research Center
for Information Technologies
Haid-und-Neu-Strasse 10-14
D-76131 Karlsruhe, Germany
www.fzi.de/wim

Abstract

Ontologies have been established for knowledge sharing and are widely used as a means for conceptually structuring domains of interest. With the growing usage of ontologies, the problem of overlapping knowledge in a common domain becomes critical. We propose the new method FCA-MERGE for merging ontologies following a bottom-up approach which offers a structural description of the merging process. The method is guided by application-specific instances of the given source ontologies, that are to be merged. We apply techniques from natural language processing and formal concept analysis to derive a lattice of concepts as a structural result of FCA-MERGE. The generated result is then explored and transformed into the merged ontology with human interaction.

1 Introduction

Ontologies have been established for knowledge sharing and are widely used as a means for conceptually structuring domains of interest. With the growing usage of ontologies, the problem of overlapping knowledge in a common domain occurs more often and becomes critical. Domain-specific ontologies are modeled by multiple authors in multiple settings. These ontologies lay the foundation for building new domain-specific ontologies in similar domains by assembling and extending multiple ontologies from repositories.

The process of *ontology merging* takes as input two (or more) source ontologies and returns a merged ontology based on the given source ontologies. Manual ontology merging using conventional editing tools without support is difficult, labor intensive and error prone. Therefore, several systems and frameworks for supporting the knowledge engineer in the ontology merging task have recently been proposed [Hovy, 1998; Chalupsky, 2000; Noy and Musen, 2000; McGuinness *et al.*, 2000]. The approaches rely on syntactic and semantic matching heuristics which are derived from the behavior of ontology engineers when confronted with the task of merging ontologies, i. e. human behaviour is simulated. Although some of them locally use different kinds of logics

for comparisons, these approaches do not offer a structural description of the global merging process.

We propose the new method FCA-MERGE for merging ontologies following a bottom-up approach which offers a global structural description of the merging process. For the source ontologies, it extracts instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances we apply mathematically founded techniques taken from *Formal Concept Analysis* [Wille, 1982; Ganter and Wille, 1999] to derive a lattice of concepts as a structural result of FCA-MERGE. The produced result is explored and transformed to the merged ontology by the ontology engineer. The extraction of instances from text documents circumvents the problem that in most applications there are no objects which are simultaneously instances of the source ontologies, and which could be used as a basis for identifying similar concepts.

The remainder of the paper is as follows. We briefly introduce some basic definitions concentrating on a formal definition of what an ontology is and recall the basics of Formal Concept Analysis in Section 2. Before we present our generic method for ontology merging in Section 4, we give an overview over existing and related work in Section 3. Section 5 provides a detailed description of FCA-MERGE. Section 6 summarizes the paper and concludes with an outlook on future work.

2 Ontologies and Formal Concept Analysis

In this section, we briefly introduce some basic definitions. We thereby concentrate on a formal definition of what an ontology is and recall the basics of Formal Concept Analysis.

2.1 Ontologies

There is no common formal definition of what an ontology is. However, most approaches share a few core items: concepts, a hierarchical IS-A-relation, and further relations. For sake of generality, we do not discuss more specific features like constraints, functions, or axioms here. We formalize the core in the following way.

Definition: A (*core*) ontology is a tuple $\mathcal{O} := (\mathcal{C}, \text{is_a}, \mathcal{R}, \sigma)$, where \mathcal{C} is a set whose elements are called *concepts*, is_a is a partial order on \mathcal{C} (i. e., a binary relation $\text{is_a} \subseteq \mathcal{C} \times \mathcal{C}$ which is reflexive, transitive, and anti-

symmetric), \mathcal{R} is a set whose elements are called *relation names* (or *relations* for short), and $\sigma: \mathcal{R} \rightarrow \mathcal{C}^+$ is a function which assigns to each relation name its arity.

As said above, the definition considers the core elements of most languages for ontology representation only. It is possible to map the definition to most types of ontology representation languages. Our implementation, for instance, is based on Frame Logic [Kifer *et al.*, 1995]. Frame Logic has a well-founded semantics, but we do not refer to it in this paper.

2.2 Formal Concept Analysis

We recall the basics of Formal Concept Analysis (FCA) as far as they are needed for this paper. A more extensive overview is given in [Ganter and Wille, 1999]. To allow a mathematical description of concepts as being composed of extensions and intensions, FCA starts with a *formal context* defined as a triple $\mathbb{K} := (G, M, I)$, where G is a set of *objects*, M is a set of *attributes*, and I is a binary relation between G and M (i. e. $I \subseteq G \times M$). $(g, m) \in I$ is read “*object g has attribute m*”.

Definition: For $A \subseteq G$, we define $A' := \{m \in M \mid \forall g \in A: (g, m) \in I\}$ and, for $B \subseteq M$, we define $B' := \{g \in G \mid \forall m \in B: (g, m) \in I\}$.

A *formal concept* of a formal context (G, M, I) is defined as a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The sets A and B are called the *extent* and the *intent* of the formal concept (A, B) . The *subconcept–superconcept relation* is formalized by $(A_1, B_1) \leq (A_2, B_2) : \Leftrightarrow A_1 \subseteq A_2$ ($\Leftrightarrow B_1 \supseteq B_2$). The set of all formal concepts of a context \mathbb{K} together with the partial order \leq is always a complete lattice,¹ called the *concept lattice* of \mathbb{K} and denoted by $\mathfrak{B}(\mathbb{K})$.

A possible confusion might arise from the double use of the word ‘concept’ in FCA and in ontologies. This comes from the fact that FCA and ontologies are two models for the concept of ‘concept’ which arose independently. In order to distinguish both notions, *we will always refer to the FCA concepts as ‘formal concepts’*. *The concepts in ontologies are referred to just as ‘concepts’ or as ‘ontology concepts’*. There is no direct counter-part of formal concepts in ontologies. Ontology concepts are best compared to FCA attributes, as both can be considered as unary predicates on the set of objects.

3 Related Work

A first approach for supporting the merging of ontologies is described in [Hovy, 1998]. There, several heuristics are described for identifying corresponding concepts in different ontologies, e.g. comparing the names and the natural language definitions of two concepts, and checking the closeness of two concepts in the concept hierarchy.

The OntoMorph system [Chalupsky, 2000] offers two kinds of mechanisms for translating and merging ontologies: syntactic rewriting supports the translation between two different knowledge representation languages, semantic rewriting offers means for inference-based transformations. It ex-

¹I. e., for each set of formal concepts, there is always a greatest common subconcept and a least common superconcept.

PLICITLY allows to violate the preservation of semantics in trade-off for a more flexible transformation mechanism.

In [McGuinness *et al.*, 2000] the Chimaera system is described. It provides support for merging of ontological terms from different sources, for checking the coverage and correctness of ontologies and for maintaining ontologies over time. Chimaera offers a broad collection of functions, but the underlying assumptions about structural properties of the ontologies at hand are not made explicit.

Prompt [Noy and Musen, 2000] is an algorithm for ontology merging and alignment embedded in Protégé 2000. It starts with the identification of matching class names. Based on this initial step an iterative approach is carried out for performing automatic updates, finding resulting conflicts, and making suggestions to remove these conflicts.

The tools described above offer extensive merging functionalities, most of them based on syntactic and semantic matching heuristics, which are derived from the behaviour of ontology engineers when confronted with the task of merging ontologies. OntoMorph and Chimarea use a description logics based approach that influences the merging process locally, e.g. checking subsumption relationships between terms. None of these approaches offers a structural description of the global merging process. FCA–MERGE can be regarded as complementary to existing work, offering a structural description of the overall merging process with an underlying mathematical framework.

There is also much related work in the database community, especially in the area of federated database systems. The work closest to our approach is described in [Schmitt and Saake, 1998]. They apply Formal Concept Analysis to a related problem, namely database schema integration. As in our approach, a knowledge engineer has to interpret the results in order to make modeling decisions. Our technique differs in two points: There is no need of knowledge acquisition from a domain expert in the preprocessing phase; and it additionally suggests new concepts and relations for the target ontology.

4 Bottom-Up Ontology Merging

As said above, we propose a bottom-up approach for ontology merging. Our mechanism is based on application-specific instances of the two given ontologies \mathcal{O}_1 and \mathcal{O}_2 that are to be merged. The overall process of merging two ontologies is depicted in Figure 1 and consists of three steps, namely (i) instance extraction and computing of two formal contexts \mathbb{K}_1 and \mathbb{K}_2 , (ii) the FCA–MERGE core algorithm that derives a common context and computes a concept lattice, and (iii) the generation of the final merged ontology based on the concept lattice.

Our method takes as input data the two ontologies and a set D of natural language documents. The documents have to be relevant to both ontologies, so that the documents are described by the concepts contained in the ontology. The documents may be taken from the target application which requires the final merged ontology. From the documents in D , we *extract instances*. The mechanism for instance extraction is further described in Subsection 5.1. This automatic knowledge acquisition step returns, for each ontology, a formal con-

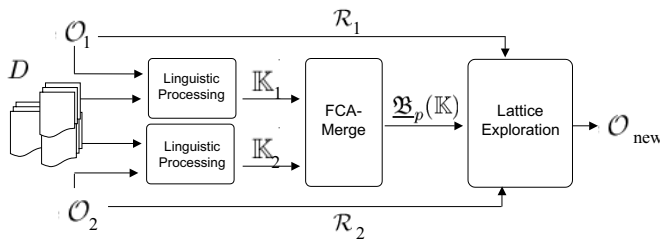


Figure 1: Ontology Merging Method

text indicating which ontology concepts appear in which documents.

The extraction of the instances from documents is necessary because there are usually no instances which are already classified by both ontologies. However, if this situation is given, one can skip the first step and use the classification of the instances directly as input for the two formal contexts.

The second step of our ontology merging approach comprises the FCA-MERGE core algorithm. The core algorithm merges the two contexts and computes a concept lattice from the merged context using FCA techniques. More precisely, it computes a *pruned concept lattice* which has the same degree of detail as the two source ontologies. The techniques applied for generating the pruned concept lattice are described in Subsection 5.2 in more detail.

Instance extraction and the FCA-MERGE core algorithm are fully automatic. The final step of *deriving the merged ontology* from the concept lattice requires human interaction. Based on the pruned concept lattice and the sets of relation names \mathcal{R}_1 and \mathcal{R}_2 , the ontology engineer creates the concepts and relations of the target ontology. We offer graphical means of the ontology engineering environment OntoEdit for supporting this process.

For obtaining good results, a few assumptions have to be met by the input data: Firstly, the documents have to be relevant to each of the source ontologies. A document from which no instance is extracted for each source ontology can be neglected for our task. Secondly, the documents have to cover all concepts from the source ontologies. Concepts which are not covered have to be treated manually after our merging procedure (or the set of documents has to be expanded). And last but not least, the documents must separate the concepts well enough. If two concepts which are considered as different always appear in the same documents, FCA-MERGE will map them to the same concept in the target ontology (unless this decision is overruled by the knowledge engineer). When this situation appears too often, the knowledge engineer might want to add more documents which further separate the concepts.

5 The FCA-MERGE Method

In this section, we discuss the three steps of FCA-MERGE in more detail. We illustrate FCA-MERGE with a small example taken from the tourism domain, where we have built several specific ontology-based information systems. Our general experiments are based on tourism ontologies that have been modeled in an ontology engineering seminar. Different ontologies have been modeled for a given text corpus on

the web, which is provided by a WWW provider for tourist information.² The corpus describes actual objects, like locations, accommodations, furnishings of accommodations, administrative information, and cultural events. For the scenario described here, we have selected two ontologies: The first ontology contains 67 concepts and 31 relations, and the second ontology contains 51 concepts and 22 relations. The underlying text corpus consists of 233 natural language documents taken from the WWW provider described above. For demonstration purposes, we restrict ourselves first to two very small subsets \mathcal{O}_1 and \mathcal{O}_2 of the two ontologies described above; and to 14 out of the 233 documents. These examples will be translated in English. In Subsection 5.3, we provide some examples from the merging of the larger ontologies.

5.1 Linguistic Analysis and Context Generation

The aim of this first step is to generate, for each ontology $\mathcal{O}_i, i \in \{1, 2\}$, a formal context $\mathbb{K}_i := (G_i, M_i, I_i)$. The set of documents D is taken as object set ($G_i := D$), and the set of concepts is taken as attribute set ($M_i := \mathcal{C}_i$). While these sets come for free, the difficult step is generating the binary relation I_i . The relation $(g, m) \in I_i$ shall hold whenever document g contains an instance of m .

The computation uses linguistic techniques as described in the sequel. We conceive an information extraction-based approach for ontology-based extraction, which has been implemented on top of SMES (Saarbrücken Message Extraction System), a shallow text processor for German (cf. [Neumann *et al.*, 1997]). The architecture of SMES comprises a *tokenizer* based on regular expressions, a *lexical analysis* component including a *word and a domain lexicon*, and a *chunk parser*. The tokenizer scans the text in order to identify boundaries of words and complex expressions like “\$20.00” or “Mecklenburg-Vorpommern”,³ and to expand abbreviations.

The lexicon contains more than 120,000 stem entries and more than 12,000 subcategorization frames describing information used for lexical analysis and chunk parsing. Furthermore, the domain-specific part of the lexicon contains lexical entries that express natural language representations of concepts and relations. Lexical entries may refer to several concepts or relations, and one concept or relation may be referred to by several lexical entries.

Lexical analysis uses the lexicon to perform (1) morphological analysis, i. e. the identification of the canonical common stem of a set of related word forms and the analysis of compounds, (2) recognition of named entities, (3) part-of-speech tagging, and (4) retrieval of domain-specific information. While steps (1), (2), and (3) can be viewed as standard for information extraction approaches, step (4) is of specific interest for our instance extraction mechanism. This step associates single words or complex expressions with a concept from the ontology if a corresponding entry in the domain-specific part of the lexicon exists. For instance, the expression “Hotel Schwarzer Adler” is associated with the concept `Hotel`. If the concept `Hotel` is in ontology \mathcal{O}_1 and document

²URL: <http://www.all-in-all.com>

³a region in the north east of Germany

I_1	Vacation	Hotel	Event	Concert	Root
doc1	x	x	x	x	x
doc2	x	x	x	x	x
doc3	x	x	x	x	x
doc4	x	x	x	x	x
doc5			x	x	x
doc6			x	x	x
doc7			x		x
doc8	x	x	x	x	x
doc9	x	x	x		x
doc10	x	x	x		x
doc11	x	x	x	x	x
doc12		x			x
doc13		x	x	x	x
doc14	x	x	x		x

I_2	Hotel	Accommodation	Musical	Root
doc1	x	x	x	x
doc2	x	x	x	x
doc3	x	x	x	x
doc4	x	x	x	x
doc5		x	x	x
doc6		x	x	x
doc7		x	x	x
doc8	x	x	x	x
doc9	x	x		x
doc10	x	x		x
doc11	x	x	x	x
doc12	x	x		x
doc13	x	x	x	x
doc14	x	x		x

Figure 2: The contexts \mathbb{K}_1 and \mathbb{K}_2 as result of the first step

g contains the expression “Hotel Schwarzer Adler”, then the relation $(g, \text{Hotel}) \in I_1$ holds.

Finally, the transitivity of the is_a -relation is compiled into the formal context, i.e. $(g, m) \in I$ and $m \text{ is_a } n$ implies $(g, n) \in I$. This means that if $(g, \text{Hotel}) \in I_1$ holds and $\text{Hotel is_a Accommodation}$, then the document also describes an instance of the concept Accommodation : $(g, \text{Accommodation}) \in I_1$.

Figure 2 depicts the contexts \mathbb{K}_1 and \mathbb{K}_2 that have been generated from the documents for the small example ontologies. E.g., document doc5 contains instances of the concepts Event , Concert , and Root of ontology \mathcal{O}_1 , and Musical and Root of ontology \mathcal{O}_2 . All other documents contain some information on hotels, as they contain instances of the concept Hotel both in \mathcal{O}_1 and in \mathcal{O}_2 .

5.2 Generating the Pruned Concept Lattice

The second step takes as input the two formal contexts \mathbb{K}_1 and \mathbb{K}_2 which were generated in the last step, and returns a *pruned concept lattice* (see below), which will be used as input in the next step.

First we merge the two formal contexts into a new formal context \mathbb{K} , from which we will derive the pruned concept lattice. Before merging the two formal contexts, we have to disambiguate the attribute sets, since \mathcal{C}_1 and \mathcal{C}_2 may contain the same concepts: Let $\widetilde{M}_i := \{(m, i) \mid m \in M_i\}$, for $i \in \{1, 2\}$. The indexation of the concepts allows the possibility that the same concept exists in both ontologies, but is treated differently. For instance, a Campground may be considered as an Accommodation in the first ontology, but not in the second one. Then the merged formal context is obtained by $\mathbb{K} := (G, M, I)$ with $G := D$, $M := \widetilde{M}_1 \cup \widetilde{M}_2$, and $(g, (m, i)) \in I \Leftrightarrow (g, m) \in I_i$.

We will not compute the whole concept lattice of \mathbb{K} , as it would provide too many too specific concepts. We restrict the computation to those formal concepts which are above at least one formal concept generated by an (ontology) concept of the source ontologies. This assures that we remain within the range of specificity of the source ontologies. More precisely, the *pruned concept lattice* is given by $\mathfrak{B}_p(\mathbb{K}) := \{(A, B) \in \mathfrak{B}(\mathbb{K}) \mid \exists m \in M: (\{m\}', \{m\}'') \leq (A, B)\}$ (with \cdot' as defined in Section 2.2).

For our example, the pruned concept lattice is shown in

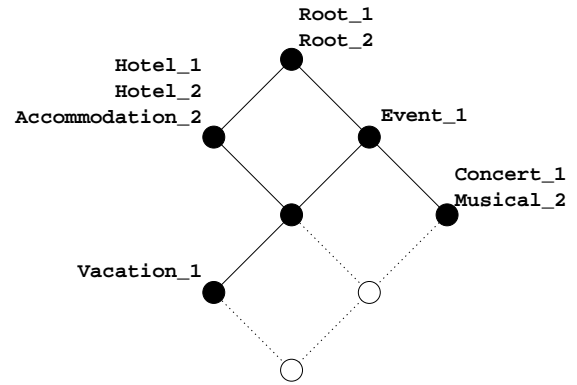


Figure 3: The pruned concept lattice

Figure 3. It consists of six formal concepts. Two formal concepts of the total concept lattice are pruned since they are too specific compared to the two source ontologies. In the diagram, each formal concept is represented by a node. The empty nodes are the pruned concepts and are usually hidden from the user. A concept is a subconcept of another one if and only if it can be reached by a descending path. The intent of a formal concept consists of all attributes (i.e., in our application, the ontology concepts) which are attached to the formal concept or to one of its superconcepts. As we are not interested in the document names, the extents of the contexts are not visualized in this diagram.

The computation of the pruned concept lattice is done with the algorithm TITANIC [Stumme *et al*, 2000]. It is slightly modified to allow the pruning. Compared to other algorithms for computing concept lattices, TITANIC has — for our purpose — the advantage that it computes the formal concepts via their *key sets* (or *minimal generators*). A key set is a minimal description of a formal concept: $K \subseteq M$ is a *key set* for the formal concept (A, B) if and only if $(K', K'') = (A, B)$ and $(X', X'') \neq (A, B)$ for all $X \subseteq K$ with $X \neq K$. In other words: K generates the formal concept (A, B) .

In our application, key sets serve two purposes. Firstly, they indicate if the generated formal concept gives rise to a new concept in the target ontology or not. A concept is new if and only if it has no key sets of cardinality one. Secondly, the key sets of cardinality two or more can be used as generic names for new concepts and they indicate the arity of new relations.

5.3 Generating the new Ontology from the Concept Lattice

While the previous steps (instance extraction, context derivation, context merging, and TITANIC) are fully automatic, the derivation of the merged ontology from the concept lattice requires human interaction, since it heavily relies on background knowledge of the domain expert.

The result from the last step is a pruned concept lattice. From it we have to derive the target ontology. Each of the formal concepts of the pruned concept lattice is a candidate for a concept, a relation, or a new subsumption in the target ontology. There is a number of queries which may be used to focus on the most relevant parts of the pruned concept lattice. We discuss these queries after the description of the general

strategy — which follows now. Of course, most of the technical details are hidden from the user.

As the documents are not needed for the generation of the target ontology, we restrict our attention to the intents of the formal concepts, which are sets of (ontology) concepts of the source ontologies. For each formal concept of the pruned concept lattice, we analyze the related key sets. For each formal concept, the following cases can be distinguished:

1. It has exactly one key set of cardinality 1.
2. It has two or more key sets of cardinality 1.
3. It has no key sets of cardinality 0 or 1.
4. It has the empty set as key set.⁴

The generation of the target ontology starts with all concepts being in one of the two first situations. The first case is the easiest: The formal concept is generated by exactly one ontology concept from one of the source ontologies. It can be included in the target ontology without interaction of the knowledge engineer. In our example, these are the two formal concepts labeled by `Vacation_1` and by `Event_1`.

In the second case, two or more concepts of the source ontologies generate the same formal concept. This indicates that the concepts should be merged into one concept in the target ontology. The user is asked which of the names to retain. In the example, this is the case for two formal concepts: The key sets `{Concert_1}` and `{Musical_2}` generate the same formal concept, and are thus suggested to be merged; and the key sets `{Hotel_1}`, `{Hotel_2}`, and `{Accommodation_2}` also generate the same formal concept.⁵ The latter case is interesting, since it includes two concepts of the same ontology. This means that the set of documents does not provide enough details to separate these two concepts. Either the knowledge engineer decides to merge the concepts (for instance because he observes that the distinction is of no importance in the target application), or he adds them as separate concepts to the target ontology. If there are too many suggestions to merge concepts which should be distinguished, this is an indication that the set of documents was not large enough. In such a case, the user might want to re-launch FCA-MERGE with a larger set of documents.

When all formal concepts in the first two cases are dealt with, then all concepts from the source ontologies are included in the target ontology. Now, all relations from the two source ontologies are copied into the target ontology. Possible conflicts and duplicates have to be resolved by the ontology engineer.

In the next step, we deal with all formal concepts covered by the third case. They are all generated by at least two concepts from the source ontologies, and are candidates for new ontology concepts or relations in the target ontology. The decision whether to add a concept or a relation to the target ontology (or to discard the suggestion) is a modeling decision, and is left to the user. The key sets provide suggestions either for the name of the new concept, or for the concepts which should be linked with the new relation. Only those key sets

⁴This implies (by the definition of key sets) that the formal concept does not have another key set.

⁵`{Root_1}` and `{Root_2}` are no key sets, as each of them has a subset (namely the empty set) generating the same formal concept.

with minimal cardinality are considered, as they provide the shortest names for new concepts and minimal arities for new relations, resp.

For instance, the formal concept in the middle of Figure 3 has `{Hotel_2, Event_1}`, `{Hotel_1, Event_1}`, and `{Accommodation_2, Event_1}` as key sets. The user can now decide if to create a new concept with the default name `HotelEvent` (which is unlikely in this situation), or to create a new relation with arity `(Hotel, Event)`, e. g., the relation `organizesEvent`.

Key sets of cardinality 2 serve yet another purpose: `{m1, m2}` being a key set implies that neither `m1 is_a m2` nor `m2 is_a m1` currently hold. Thus when the user does not use a key set of cardinality 2 for generating a new concept or relation, she should check if it is reasonable to add one of the two subsumptions to the target ontology. This case does not show up in our small example. An example from the large ontologies is given at the end of the section.

There is exactly one formal concept in the fourth case (as the empty set is always a key set). This formal concept gives rise to a new largest concept in the target ontology, the `Root` concept. It is up to the knowledge engineer to accept or to reject this concept. Many ontology tools require the existence of such a largest concept. In our example, this is the formal concept labeled by `Root_1` and `Root_2`.

Finally, the `is_a` order on the concepts of the target ontology can be derived automatically from the pruned concept lattice: If the concepts `c1` and `c2` are derived from the formal concepts `(A1, B1)` and `(A2, B2)`, resp., then `c1 is_a c2` if and only if `B1 ⊇ B2` (or if explicitly modeled by the user based on a key set of cardinality 2).

Querying the pruned concept lattice. In order to support the knowledge engineer in the different steps, there is a number of queries for focusing his attention to the significant parts of the pruned concept lattice.

Two queries support the handling of the second case (in which different ontology concepts generate the same formal concept). The first is a list of all pairs $(m_1, m_2) \in C_1 \times C_2$ with $\{m_1\}' = \{m_2\}'$. It indicates which concepts from the different source ontologies should be merged.

In our small example, this list contains for instance the pair `(Concert_1, Musical_2)`. In the larger application (which is based on the German language), pairs like `(Zoo_1, Tierpark_2)` and `(Zoo_1, Tiergarten_2)` are listed. We decided to merge `Zoo` [engl.: zoo] and `Tierpark` [zoo], but not `Zoo` and `Tiergarten` [zoological garden].

The second query returns, for ontology \mathcal{O}_i with $i \in \{1, 2\}$, the list of pairs $(m_i, n_i) \in C_i \times C_i$ with $\{m_i\}' = \{n_i\}'$. It helps checking which concepts out of a single ontology might be subject to merge. The user might either conclude that some of these concept pairs can be merged because their differentiation is not necessary in the target application; or he might decide that the set of documents must be extended because it does not differentiate the concepts enough.

In the small example, the list for \mathcal{O}_1 contains only the pair `(Hotel_1, Accommodation_1)`. In the larger application, we had additionally pairs like `(Räumliches, Gebiet)` and `(Auto, Fortbewegungsmittel)`. For the target applica-

tion, we merged Räumliches [spatial thing] and Gebiet [region], but not Auto [car] and Fortbewegungsmittel [means of travel].

The number of suggestions provided for the third situation can be quite high. There are three queries which present only the most significant formal concepts out of the pruned concepts. These queries can also be combined.

Firstly, one can fix an upper bound for the cardinality of the key sets. The lower the bound is, the fewer new concepts are presented. A typical value is 2, which allows to retain all concepts from the two source ontologies (as they are generated by key sets of cardinality 1), and to discover new binary relations between concepts from the different source ontologies, but no relations of higher arity. If one is interested in having exactly the old concepts and relations in the target ontology, and no suggestions for new concepts and relations, then the upper bound for the key set size is set to 1.

Secondly, one can fix a minimum support. This prunes all formal concepts where the cardinality of the extent is too low (compared to the overall number of documents). The default is no pruning, i. e., with a minimum support of 0%. It is also possible to fix different minimum supports for different cardinalities of the key sets. The typical case is to set the minimum support to 0% for key sets of cardinality 1, and to a higher percentage for key sets of higher cardinality. This way we retain all concepts from the source ontologies, and generate new concepts and relations only if they have a certain (statistical) significance.

Thirdly, one can consider only those key sets of cardinality 2 in which the two concepts come from one ontology each. This way, only those formal concepts are presented which give rise to concepts or relations linking the two source ontologies. This restriction is useful whenever the quality of each source ontology *per se* is known to be high, i. e., when there is no need to extend each of the source ontologies alone.

In the small example, there are no key sets with cardinality 3 or higher. The three key sets with cardinality 2 (as given above) all have a support of $\frac{11}{14} \approx 78.6\%$. In the larger application, we fixed 2 as upper bound for the cardinality of the key sets. We obtained key sets like (Telefon_1 [telephone], Öffentliche_Einrichtung_2 [public institution]) (support = 24.5%), (Unterkunft_1 [accommodation], Fortbewegungsmittel_2 [means of travel]) (1.7%), (Schloß_1 [castle], Bauwerk_2 [building]) (2.1%), and (Zimmer_1 [room], Bibliothek_2 [library]) (2.1%). The first gave rise to a new concept Telefonzelle [public phone], the second to a new binary relation hatVerkehrsansbindung [hasPublicTransportConnection], the third to a new subsumption Schloß ist_a Bauwerk, and the fourth was discarded as meaningless.

6 Conclusion and Future Work

FCA-MERGE is a bottom-up technique for merging ontologies based on a set of documents. In this paper, we described the three steps of the technique: the linguistic analysis of the texts which returns two formal contexts; the merging of the two contexts and the computation of the pruned concept lattice; and the semi-automatic ontology creation phase which

supports the user in modeling the target ontology. The paper described the underlying assumptions and discussed the methodology.

Future work includes the closer integration of the FCA-MERGE method in the ontology engineering environment ONTOEDIT. In particular, we will offer views on the pruned concept lattice based on the queries described in Subsection 5.3. It is also planned to further refine our information-extraction based mechanism for extracting instances.

The evaluation of ontology merging is an open issue [Noy and Musen, 2000]. We plan to use FCA-MERGE to generate independently a set of merged ontologies (based on two given source ontologies). Comparing these merged ontologies using the standard information retrieval measures as proposed in [Noy and Musen, 2000] will allow us to evaluate the performance of FCA-MERGE.

On the theoretical side, an interesting open question is the extension of the formalism to features of specific ontology languages, like for instance functions or axioms. The question is (i) how they can be exploited for the merging process, and (ii) how new functions and axioms describing the interplay between the source ontologies can be generated for the target ontology.

Acknowledgements

This research was partially supported by DFG and BMBF.

References

- [Chalupsky, 2000] H. Chalupsky: OntoMorph: A translation system for symbolic knowledge. *Proc. KR '00*, Breckenridge, CO, USA, 471–482.
- [Ganter and Wille, 1999] B. Ganter, R. Wille: *Formal Concept Analysis: mathematical foundations*. Springer.
- [Hovy, 1998] E. Hovy: Combining and standardizing large-scale, practical ontologies for machine translation and other uses. *Proc. 1st Intl. Conf. on Language Resources and Evaluation*, Granada.
- [Kifer *et al.*, 1995] M. Kifer, G. Lausen, J. Wu: Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* 42(4), 741–843.
- [McGuinness *et al.*, 2000] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder: An environment for merging and testing large Ontologies. *Proc. KR '00*, 483–493.
- [Neumann *et al.*, 1997] G. Neumann, R. Backofen, J. Baur, M. Becker, C. Braun: An information extraction core system for real world German text processing. *Proc. ANLP-97*, Washington.
- [Noy and Musen, 2000] N. Fridman Noy, M. A. Musen: PROMPT: algorithm and tool for automated ontology merging and alignment. *Proc. AAAI '00*, 450–455.
- [Schmitt and Saake, 1998] I. Schmitt, G. Saake: Merging inheritance hierarchies for database integration. *Proc. CoopIS'98*, IEEE Computer Science Press, 322–331.
- [Stumme *et al.*, 2000] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal: Fast computation of concept lattices using data mining techniques. *Proc. KRDB '00*, <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/>, 129–139.
- [Wille, 1982] R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets*. Reidel, Dordrecht, 445–470.