Let's plan it deductively!

W. Bibel
Technical University Darmstadt, Germany

Abstract

The paper describes a transition logic, TL, and a deductive formalism for it. It shows how various important aspects (such as ramification, qualification, specificity, simultaneity, indeterminism etc.) involved in planning can be modelled in TL in a rather natural way. (The deductive formalism for) TL extends the linear connection method proposed earlier by the author by embedding the latter into classical logic, so that classical and resource-sensitiv reasoning coexist within TL. The attraction of a logical and deductive approach to planning is emphasized and the state of automated deduction briefly described.

1 Introduction

Artificial Intelligence (AI, or intellectics [Bibel, 1992a]) aims at creating artificial intelligence. Were there no natural intelligence, the sentence would be meaningless to us. Hence understanding natural intelligence by necessity has always been among the goals of intellectics (and is also the goal of cognitive science).

Different points of view for approaching the goal of creating artificial intelligence have been distinguished [Kushmerick, 1996]. Logicism [Nilsson, 1991], cognitivism [Laird *et al.*, 1987], and situated action [Agre, 1995] are three out of several such points of view. In a nutshell, the logicistic point of view argues that man can describe his creations (including an artificial intelligence) only by natural linguistic, hence logical means; thus any way towards artificial intelligence must in some sense be a logical one. This author is strongly committed to the logicistic approach. As a consequence he believes that any other approach is in fact a logicistic one in disguise.

Intelligence has many features. Clearly one of them is the ability to plan ahead in time. Intuitively, planning is logical reasoning of some kind. All the more one might expect that planning is the domain where logic and its deductive machinery excel. The fact is that it does not. There are many software systems in everyday use solving planning tasks, but to the author's best knowledge none of them is based on logic and has a deductive component. Does this imply that logic is irrelevant for planning and for artificial intelligence for that matter?

While intelligence implies the ability for planning, the converse has not necessarily to be true. It very much depends on what kind of planning is meant. In a fixed and relatively restricted domain (such as text layout) planning may well be realized in a purely functional way and with standard programming techniques. But functional (or procedural) programming has its limits as we enter more complex and unpredictable domains; in particular it will never be able to produce a behavior which rightly deserves to be named "intelligent" (surely as a user of computers you noticed the stupidity of text layout systems). Section 7, as well as numerous texts in the literature, give arguments for this statement. It also gives reasons which explain the resistance of the software industry to a bolder move into a logic technology for planning and for other applications. In other words, logic is essential for intelligent planning in the true sense of the term, but industry is not ready to build intelligent systems.

It is not the task of intellecticians to lament about this state of affairs but rather to prepare for the coming day when the market will be ripe for a broader use of a truly intelligent technology and to develop the best possible technological basis for it. In fact, if we are frank there is yet a lot to be developed before we can comfortably go out to industry and offer a coherent set of methods for dealing with the many facets of intelligence including planning.

In the present paper I review the state of the art in deductive planning with an emphasis on the contributions from research groups influenced by my own work. While much of the work in deductive planning has focused on representational issues we have always approached the problem with the necessary and available deductive techniques in mind. Since the methods and systems growing out of our work have finally achieved a leading position in the deduction community by winning the CADE-96 competition in automated theorem proving with the SETHEO system [Letz *et al.*, 1992; Moser *et al.*, 1997], we are perhaps also well placed to import the best possible techniques into the planning

community. In other words, the paper will focus on deductive planning as well as on the underlying deduction techniques. Since the author sees planning as just one among a number of aspects for achieving artificial intelligence, the case for deductive planning is presented in this paper in form of a paradigm case for achieving the grander goal of artificial intelligence. The paper will therefore not only point the way to intelligent planning but to some extent also the author's proposed way to artificial intelligence (the "it" in the title).

In the next section we introduce the logical language used in our approach and discuss the deductive aspects thereafter. The resulting computational logic is called *transition logic* (TL) which has classical as well as resource-sensitive features. Section 4 shows what TL has to do with planning and computation (or with temporal prediction or postdiction for that matter). Section 5 compares the logic with other known logics. Section 6 shows how the various aspects involved in reasoning about actions and causality can be taken into account within TL. Specifically, we discuss ramification, qualification, specificity, simultaneity, indeterminism, continuity, hierarchies etc. Finally, we briefly describe the tensions between the specialistic and logistic approaches in AI and explain it by outlining the underlying pattern. Given the impressive recent achievements in automated deduction we conclude with making a case for a logical path towards an artificial intelligence.

2 A logical language

Any textbook on AI also contains some introduction to first-order logic so that we may assume the reader to be familiar with it. Only to communicate our notational conventions we mention that there are objects named by constants (a, b, c), (n-ary) functions named by function symbols (f, g, h) and (n-ary) relations named by predicate symbols (P, Q, R). Terms (r, s, t), built from variables (x, y, z, ranging over objects), constants and function symbols, again denote objects. Literals (K, L) are relations among objects or the negation ($\neg$) thereof. They correspond to simple factual sentences in natural language (such as "John is married to the mother of Bill").

For building more complicated sentences represented as formulas (F, G, H) we use the well-known classical (logical) operators $\wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$ as well as the resource-sensitive operators & (non-idempotent conjunction), | (non-idempotent exclusive disjunction), and $\Rightarrow$ (transition). The latter need explanation which follows.

The language of predicate logic has been designed to express natural language sentences formally and unambiguously. This was done in a biased way since many of those involved in the design (such as Frege [Frege, 1879]) had mainly sentences of a mathematical nature in mind. Sentences involving actions were not taken into serious consideration until the publication of the situation calculus in 1969 [McCarthy and Hayes, 1969] in which any n-ary relation P is extended to an $(n+1)$-ary one by an argument for determining the situation in which the relation is meant to hold (see Section 5.4).

Natural language apparently does not need such an extra vehicle. A (static) mathematical sentence (such as "if a number is greater than zero then it is positive") looks exactly like a (dynamic) one about actions (such as "if I take the book then it is mine"). In [Bibel, 1986a] the main idea for a logic has been outlined which resembles natural language more closely in this aspect of treating actions than does the situation calculus. The approach then was called *linear connection method* or shortly LCM; for the logic we introduce here on the basis of LCM we propose the name *transition logic* or TL. The idea underlying LCM spawned a great number of studies based on this idea such as [Fronhöfer, 1987; Bibel *et al.*, 1989; Hölldobler and Schneeberger, 1990; Große *et al.*, 1992; Brüning *et al.*, 1992; 1993; Hölldobler and Thielscher, 1993; 1995; Große *et al.*, 1996; Fronhöfer, 1996; Herrmann and Thielscher, 1996; Thielscher, 1996; Eder *et al.*, 1996; Thielscher, 1997b; Bornscheuer and Thielscher, 1997; Thielscher, 1997a] to mention several of them. Here facts may be treated as resources which may be consumed by actions. Two different formalisms are used to achieve this. One, TL, employs the additional set of resource-sensitive operators &, |, $\Rightarrow$ just introduced (the other achieves their effects on the term level of classical logic as we will see in Section 5.3).

A rule $K \Rightarrow L$, called an *action* (or *transition*) rule (or *effect axiom*), models an action which consumes K and produces L. For instance,

$$on_table(book) \Rightarrow in_hand(book)$$

can be seen as the equivalent in TL of the situation calculus rule

$$on_table(book, s) \rightarrow in_hand(book, result(take, s)).$$

In classical logic $L \wedge L$ is equivalent with L according to the rule of idempotence. In real-world scenarios it does matter, however, whether you have the same thing (say a dollar) once or twice. Similarly, it does matter whether you take your dollar or mine. That is why we need the two extra operators &, | which behave just like their classical counterparts $\wedge, \vee$ except for the rule of idempotence, which does *not* hold for them, and for | modelling an exclusive (rather than an inclusive) alternative. In consequence, we will not have the law of distributivity which allows | to be distributed over &.

Formulas built from literals by means of the quantifiers and the resource-sensitive operators only are called *r-formulas*. r-formulas without | are also called *conjunctive* r-formulas. General *formulas* of TL are r-formulas, and any expression built from those by means of the classical operators. For instance, $P \,\&\, (P \Rightarrow Q) \Rightarrow Q$ is an r-formula, hence a formula, $(P' \rightarrow P) \rightarrow [P' \,\&\, (P \Rightarrow Q) \Rightarrow Q]$ is a formula but not an r-formula, and $P \,\&\, (P \rightarrow Q) \Rightarrow Q$ is not a formula (nor an r-formula) since the definition does not allow classical operators (other than quantifiers) below a resource-sensitive one in the formula tree. An r-subformula which is not a proper

subformula of an r-subformula is also called an *r-part* in the given formula.

3 Basic deductive machinery

We will deal in this paper with a restricted class of r-formulas only which have the form $G_0 \,\&\, (F_1 \Rightarrow G_1) \,\&\, \ldots \,\&\, (F_n \Rightarrow G_n) \Rightarrow H$ whereby $\Rightarrow$ does not occur in F_i, G_j, H. Semantic entailment $\models$ for the resulting class of formulas will be introduced only informally. $T \rightarrow [K \,\&\, (K \Rightarrow L)] \models F$ holds if F is classically entailed by $T \rightarrow K$ or by $T \rightarrow L$, depending of the state reached by not performing or performing the transition $K \Rightarrow L$ in the r-part which, if executed, consumes K and produces L.

As we see two different states, say σ_0, σ_1, are to be distinguished in this example, the one before and the other after the transition. Semantic entailment is dependent on these states. For instance, we might write $T \rightarrow [K \,\&\, (K \Rightarrow L)]_{\sigma_0} \models T \rightarrow K$. As more transitions get involved we obtain more such states to be distinguished.[1] Validity, $\models F$, is then defined as usual. Section 5 will resume the discussion of the semantics of TL while in the present section we focus on its deductive aspects.

As the original name of our approach, *linear connection method* (LCM), suggests, the basic deductive machinery is based on the connection method [Bibel, 1993; 1987]. This deductive method is characterized by the fundamental theorem which in turn characterizes validity of a formula by the so-called spanning property explained shortly. Many different logical calculi can be based on this method.

In order to explain the spanning property we need the concepts of a path through a formula and of a connection. A *path* through a formula F is the set of literals of any conjunct of the conjunctive normal form of F. Paths can best be illustrated if formulas are displayed as matrices. Matrices (positively) represent disjunctions of clauses which in turn represent conjunctions of literals (or, in general, matrices). Consider the formula $P \wedge (P \rightarrow Q) \rightarrow Q$ (expressing the well-known logical rule of modus ponens). In negation normal form the same formula reads $\neg P \vee (P \wedge \neg Q) \vee Q$, which is a disjunction of three clauses. Hence as a matrix it looks as follows.

$$\begin{bmatrix} & \neg Q & \neg P \\ Q & P & \end{bmatrix}$$

As an aside we mention that a rotation of this matrix by 90° (counterclockwise) basically yields the corresponding PROLOG program except for the differences due to the negative representation used in PROLOG.

[1] [Thielscher, 1997a] gives a precise semantics which, however, needs adaption to TL and the view just outlined.

```
P .
Q   :-  P .
    ?-  Q
```

A path through such a matrix (or the formula it represents, or the corresponding PROLOG program) is now the set of literals obtained by selecting exactly one literal from each clause (or, in other words, traversing the matrix say from left to right). In the present example there are exactly two such paths, namely $\{Q, \neg Q, \neg P\}$ and $\{Q, P, \neg P\}$. The disjunction of the literals of these two paths are obviously the disjuncts of the conjunctive normal form of $\neg P \vee (P \wedge \neg Q) \vee Q$.

A *connection* is a subset of a path of the form $\{R, \neg R\}$. There are two connections in our present example illustrated as arcs in the following display.

$$\begin{bmatrix} & \neg Q & \neg P \\ Q & P & \end{bmatrix}$$

A set of connections (or *mating*) is called *spanning* if each path through the matrix contains at least one connection. This is the case for the two connections of our example, hence the formula is (of course) valid according to the fundamental theorem mentioned at the outset of the section. Recall that the matrix form is used just for illustration and is thus not essential for the connection method. The connections (and the spanning property) could as well have been identified in the original formula as follows.

$$P \;\wedge\; (P \;\rightarrow\; Q) \;\rightarrow\; Q$$

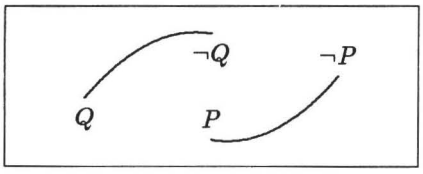
A chain of two (or more) connections like the two displayed in the matrix may thus be regarded as an encoding of one (or more) applications of modus ponens. This illustration also demonstrates that it is connections which lie at the heart of deductive reasoning (more so than rules like the $P \rightarrow Q$ in the example). In some sense a connection may also be seen as an encoding of an application of the well-known resolution rule. So far connections have been illustrated for propositional examples. They apply to first-order formulas in the obvious way, connecting literals with opposite signs and unifiable terms. An example is the following matrix.

$$\begin{bmatrix} & \neg Qx & \neg Pa \\ Qa & Px & \end{bmatrix}$$

Here validity is established by the two spanning connections along with the substitution $x \backslash a$, which makes the connected literals complementary.

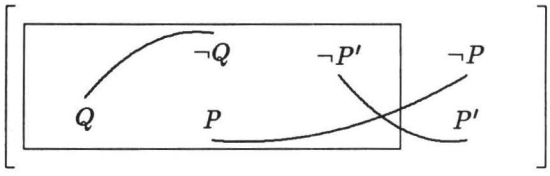
Up to this point we have restricted our discussion to deduction for purely classical formulas. The characterizing spanning property carries over to the case of general

formulas in our logical language with one minor modification to be explained shortly. In fact, if we take the r-formula $P \,\&\, (P \Rightarrow Q) \Rightarrow Q$ as our first example then we may use exactly the same matrix as the one before to represent the formula in a two-dimensional way. In fact, in spite of the modification in the formula exactly the same proof for validity is obtained.



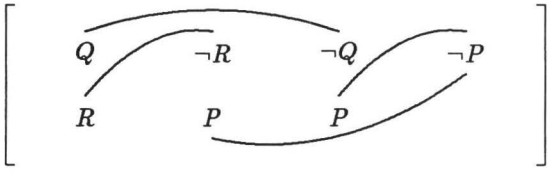
In order to distinguish it from the classical matrix we use a box rather than brackets. One should note, however, that the semantics of the operations represented by the structural arrangement this time is quite a different one. As in the classical case the matrix representation is more of an illustrative relevance, since the connections also here could as well have been placed in the original formula.

Let us now consider a general formula like $(P' \rightarrow P) \rightarrow [P' \,\&\, (P \Rightarrow Q) \Rightarrow Q]$ which in its classical part expresses that P' is just another name for P and which is represented by the following matrix.

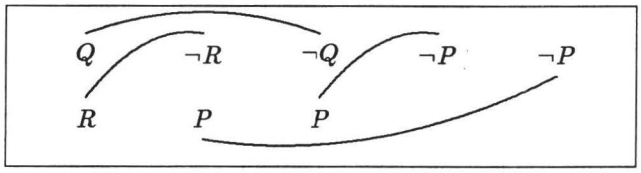


The r-submatrix is boxed.[2] The three connections are obviously spanning. Hence the formula is again valid.

The need for a modification becomes clear if we compare the classically valid formula $P \wedge (P \rightarrow Q) \wedge (P \rightarrow R) \rightarrow Q \wedge R$ whose proof is



with the analogical r-formula $P \,\&\, (P \Rightarrow Q) \,\&\, (P \Rightarrow R) \Rightarrow Q \,\&\, R$. While the validity of the first formula is clear from the spanning property obviously satisfied for the matrix, the second formula should intuitively *not* be valid. Namely, if a dollar (P) buys a coffee (Q), and if a dollar buys a tea (R), and if I have just one dollar (as the formula suggests) then clearly I cannot buy both coffee and tea since I would rather need *two* dollars for that purpose. Since the matrix and its connections would be exactly the same for the second formula as for the first one, we are lead to conclude that the spanning property (characterizing validity in the classical case) needs some modification. The kind of modification becomes clear if we add another P to the present r-formula, ie. $P \,\&\, P \,\&\, (P \Rightarrow Q) \,\&\, (P \Rightarrow R) \Rightarrow Q \,\&\, R$, which intuitively is valid as just illustrated and compare its proof



with the previous one. In the latter matrix *each literal is contained in at most one connection* while in the former this *linearity restriction in its original form* [Bibel, 1986a]) is not satisfied because the literal $\neg P$ is contained in more than one, namely in two connections. To cover the general case considered in the present paper this linearity restriction needs a more general definition.

For that purpose we inductively introduce the concept of the *directionality*[3] 0 (for consumption) and 1 (for resource) of the nodes in the formula tree of an r-formula. The root has directionality 0. If a node with directionality d is labelled by & or by | then its successor nodes have the same directionality d; if it is labelled $\Rightarrow$ then the directionality of the left successor node is $(d+1) \bmod 2$ while that of the right successor node is d. The directionality partitions the occurrences of literals in an r-formula (or r-matrix) into *resource literals* if their directionality in the formula is 1, and *consumption literals* if it is 0. We attach this directionality to a literal if needed as an upper index. In all our matrix examples the directionality is 1 for a negated literal and 0 for an unnegated one.

A *chain of connections* in a matrix M is a sequence $(c_1, \ldots, c_n)$, $n \geq 1$, of directed connections $(\overline{L}_i, L_i)$, such that for any i, $1 \leq i \leq n-1$, the *end literal* of c_i, ie. L_i, and the *start literal* of c_{i+1}, ie. $\overline{L}_{i+1}$, are literals in the same (top level) clause of M but do not occur in one and the same path through M, ie. they are *vertically related* [Fronhöfer, 1996]. A *cycle* is a chain such that also its *start literal* $\overline{L}_1$, ie. the start literal of c_1, and its *end literal* L_n are vertically related in this sense. A chain is called an *r-chain* if, viewed as a list $(K_1, \ldots, K_{2n})$ of literals K_j, $1 \leq j \leq 2n$, $n \geq 1$, K_1 is of directionality 1, K_1 and K_{2n} are in the same r-part M_0 of M while no other K_k, $k \neq 1, 2n$ is in M_0, and K_{2i} and K_{2i+1} are in different r-parts. Two r-chains $(K_1, \ldots, K_{k-1}, K_k, \ldots)$ and $(K_1, \ldots, K_{k-1}, K'_k, \ldots)$ are called *c-distinct* if K_k and K'_k occur in different r-parts of M, or $k-1$ is odd and K_k and K'_k are different literal occurrences. A set of connections in a matrix M *satisfies the linearity restriction* if each directionality-1 literal in an r-part of M is start literal of no more than one c-distinct r-chains.

These definitions go a bit beyond what is intended with this paper simply because they are novel and have not been published before. In fact it could well be that they need further adjustment once the structure of formulas and their semantics are finally settled. We just mention here that the definitions aim at yielding the

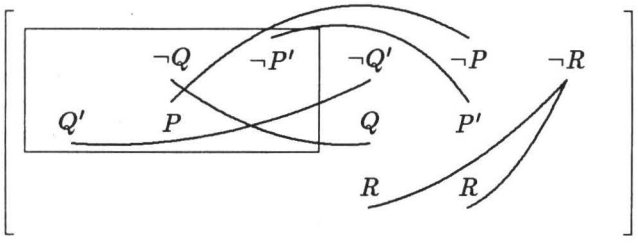
[2] For simplicity we do not box the literals, which formally are r-parts, in the classical part.

[3] Note that directionality is not the same as polarity.

fundamental theorem according to which a formula F is valid if (some compound instance F' of) F has a spanning and unifiable mating which satisfies the linearity restriction and has no (regular [Fronhöfer, 1996]) cycles within any of M's r-parts. Since none of our examples will need compound instances we refer to the ATP literature for the respective details (eg. see [Bibel, 1987]).

None of our examples has cycles. Therefore we ignore this technically intricate issue and refer the interested reader to [Fronhöfer, 1996] for the details. All matrices considered so far satisfy also the linearity restriction except the one just discussed in which the directionality-1 literal $\neg P$ is the start of two different r-chains (each with one connection) if the matrix is regarded as an r-matrix. The restriction is also satisfied in the matrix associated with the general formula $(P' \rightarrow P) \rightarrow [P' \,\&\, (P \Rightarrow Q) \Rightarrow Q]$ shown above since each of the two directionality-1 literals is start literal of exactly one r-chain, namely $\neg Q$ of the r-chain $((\neg Q, Q))$ and $\neg P'$ of $((\neg P', P'), (\neg P, P))$. r-chains with more than one connection may be regarded as theory connections [Bibel, 1993] while *within* an r-matrix any r-chain has exactly one connection.

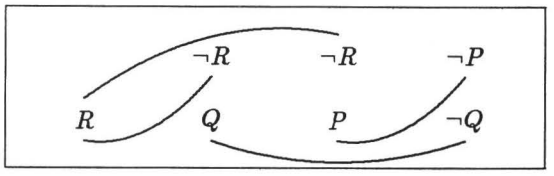
A more complex example derived from the previous one, namely $R \wedge (R \rightarrow (P' \rightarrow P)) \wedge (R \rightarrow (Q' \rightarrow Q) \rightarrow [P' \,\&\, (P \Rightarrow Q) \Rightarrow Q']$ yields the following proof.



As in the example before the linearity restriction is fulfilled in its spanning mating for the two resource literals of its r-part. The same is true for the formulas $(P \,\&\, Q \rightarrow R \,\&\, S) \rightarrow (P \,\&\, Q \Rightarrow R \,\&\, S)$ and $[R \rightarrow (P \Rightarrow Q)] \rightarrow (P \,\&\, R \Rightarrow Q)$. On the other hand, the formulas $(P \rightarrow P \wedge P) \rightarrow (P \Rightarrow P \,\&\, P)$ and $(P \rightarrow Q) \rightarrow (P \Rightarrow Q \,\&\, Q)$ have spanning matings, but in line with intuition none of them satisfies the linearity restriction.

The solution presented here and illustrated with these examples informally was already given in [Bibel, 1986a] where it reads: "the linearity restriction applies to connections with one element in a transition rule only, not to those possibly required for additional 'static' reasoning in the usual way". The generalized definition of the linearity restriction just presented also covers the case of disjunction $|$, which has been the topic of the publications [Große *et al.*, 1992; Brüning *et al.*, 1992; 1993] where a different solution based on the restriction in its original form has been offered.[4] We illustrate our solution for the formula $(P \mid Q) \,\&\, (P \Rightarrow R) \,\&\, (Q \Rightarrow R) \Rightarrow R$ with the following matrix.

[4] In contrast to these papers we introduced disjunction $|$ as a non-idempotent operation here.



Each of the four resource literals is the start literal of exactly one r-chain, each with exactly one connection. So the linearity restriction is satisfied in this spanning mating although the literal R is contained in two different connections thus violating the restriction in its original form.

Just for simplicity we kept all our examples at the propositional level. In general, formulas may include quantifiers which require the additional property of unifiability of connected literals. Otherwise everything goes as described. In the presence of quantifiers we may also consider more than one instance of a given action rule each of which is to be treated as a separate rule just like the ones in our examples so far.

4 Planning, temporal projection, and postdiction

Planning is the main topic of this paper. So what has the formalism, TL, introduced in the last two sections to do with planning? Well, given an r-formula description of an initial state (say P), of a goal state (Q), and of the possible actions in terms of r-implicational formulas ($P \Rightarrow Q$), then all we need to do is to activate a theorem prover for TL. Any proof it finds represents a plan which satisfies the description. The actions determined by the plan are those action rules used in the proof. Their order of execution is determined by the chains of connections establishing the proof from the initial to the goal situation.

For instance, consider the formula

$$P_0 \,\&\, Q_0 \,\&\, (P_0 \Rightarrow P_1) \,\&\, (P_1 \Rightarrow P_2) \,\&\, (Q_0 \Rightarrow Q_1) \,\&\, (Q_1 \Rightarrow Q_2) \Rightarrow P_2 \,\&\, Q_2$$

It may be interpreted as dressing one's left (P) and right (Q) foot (index 0) with a sock (1) and a shoe (2). There are no alternate choices for the 8 connections which establish the proof for this formula, so the reader may easily identify them. As these connections prescribe, the P_0-rule must precede the P_1-rule and similarly for the Q-rules. On the other hand the P-connections are independent from the Q-connections in terms of any order. So the plan leaves open in which order the actions for the left and right foot are mixed together which may therefore be determined arbitrarily. That is, the resulting plans are partially ordered ones as is desirable for applications.

For the formula $P \,\&\, Q \,\&\, (P \,\&\, Q \Rightarrow P \,\&\, \neg Q) \,\&\, (P \Rightarrow \neg P) \Rightarrow \neg P \,\&\, \neg Q$ its five connections establishing validity determine a linearly ordered plan since the second rule can only be executed after the first one. The formula may be interpreted as the adultery drama by Drew McDermott (personal communication) whereby a husband

(P) shoots his rival ($\neg Q$) and *then* himself ($\neg P$), and not the other way round, which obviously would not yield a proof in our logic.

For decades formalisms for planning were plagued by the notorious *frame problem* [McCarthy and Hayes, 1969]: how to characterize the "frame" of an action, ie. everything not affected by the action. The aspects of the frame problem now called *representational* and *inferential* frame problem [Russell and Norvig, 1994] are no problems any more in our formalism, which is in stark contrast to the most popular competitor, namely the situation calculus (see Section 5). In fact, LCM has been the first method which actually solved these aspects of the frame problem and did so in the optimally possible way.

So our logic and its deductive machinery are fully appropriate for solving planning tasks of the sort considered so far. In certain contexts one might expect an explicit answer, ie. a concrete plan. For that purpose [Bibel, 1986a] introduced state literals, $S(x)$, which keep trace of the states passed through while executing a plan. This also requires that an action rule such as the suicide killing rule above would read $S(x) \,\&\, P \Rightarrow S(suicide(x)) \,\&\, \neg P$. By unification the variable denoting the goal situation will then along with a successful proof always provide a term which expresses the linearized sequence of actions. In this option the planning system might compute all possible proofs and offer all corresponding plans as alternatives. The option also allows for multiple copies of action rules (by way of the quantifier involved) if needed, the way mentioned at the end of the last section.[5]

There are modes other than planning which are of interest in our context. One such mode is called *temporal projection*: given the initial state and the sequence of actions, determine the resulting state. Or, in another mode called *postdiction*, one would like to determine the initial state in order to explain with it the observed outcome of a sequence of actions. Since theorem provers can be used in all these modes as well, namely theorem checking (ie. planning), proof checking (ie. temporal projection), abduction (ie. a form of postdiction), all these and other modes can of course be modelled by proof systems for TL. That is, the deductive approach is as versatile as one would wish.

5 Semantics and alternative formalisms

In principle there are two ways to provide the logic TL introduced in the previous sections with a precise semantics, the direct and the indirect one. In [Bibel *et al.*, 1989] the direct route has been taken. Informally, we may think of a Kripke-style semantics with an actual and further worlds. Whenever a proof activates a rule with a $\Rightarrow$ the transition from the premises to the conclusion amounts to a transition from the present world to some next one which differs from the present one only in the changes specified by the rule. This has already been illustrated at the beginning of Section 3 with a simple example.

The indirect way to specify a semantics consists in embedding TL in an existing formalism for which a semantics is already known. In the next and the third subsection we describe two formalisms which were used for this purpose. Otherwise the section compares TL with several related formalisms.

5.1 TL and LL

A very close relative of TL is *linear logic* [Girard, 1987] which was first published about two years *after* the first publication of LCM [Bibel, 1986b], predecessor and part of TL.[6] We briefly summarize what is known about the relationship of TL with linear logic (LL).

For that purpose we restrict the language of TL to its r-formulas only and refer as rTL (or LCM for that matter) to this restricted part of TL.[7] Further we consider only r-formulas without $|$ and refer to crTL to this part of TL. Theorem 35 in [Fronhöfer, 1996] states (among other things) that crTL and the multiplicative part of LL with the exchange and the weakening rule, also known as (classical) BCK [Restall, 1994] (or affine logic), amount to the same thing (in terms of derivability). A similar result was obtained in [Große *et al.*, 1996]. Its Theorem 4.1 states (among other things) that crTL and conjunctive linear theories as defined in [Masseron *et al.*, 1990] on the basis of LL amount again to the same thing (in terms of derivability). In other words, TL and LL more or less coincide in their multiplicative parts so that crTL may inherit its semantics from LL [Gallier, 1991].[8] Although it has not been shown in detail, it is conjectured that these results may be generalized to rTL. In fact, LL is as expressive as TL since classical logic can be embedded in LL [Girard, 1987]. On the other hand, we have proposed for practical purposes a much more restricted class of formulas both by definition of TL itself and by the restriction of the formulas in its r-part. Finally, LL's proof nets are but another name of LCM's spanning matings satisfying the linearity restriction (with the minor difference that not *all* literals need to be connected in LCM, an advantage of LCM or TL over LL again from a practical point of view). In summary, although TL is a linear-logic-type of logic it offers more attractions than LL for practical purposes.

To [Fronhöfer, 1996] we owe much of the formal background. For instance, its Chapter 3 provides a formal

[5] An alternative for this consists in the introduction of the exponential ! from linear logic which would specify if rules can be used abitrarily many times.

[6] Only later it became clear that [Lambek, 1958] as well as *relevance logic* [Anderson and Belnap, 1975] are important predecessors of LCM as of linear logic.

[7] On the other hand we dispense for the discussion in this section with the restriction on the formula structure introduced at the beginning of Section 3.

[8] Note that we did not follow the unintuitive notation used in LL: TL's $\Rightarrow, \&, |$ correspond to LL's $\multimap, \otimes, \oplus$, respectively (while the two remaining binary junctors of linear logic can be defined in terms of these three ones).

justification of our use of the matrix representation (for crTL at any rate). There is a slight difference between the linearity restriction used in the present paper and the corresponding restriction in Theorem 35 of [Fronhöfer, 1996]. The latter uses the restriction in its original form (mentioned in Section 3).

5.2 TL and STRIPS

Many planning systems are based on the STRIPS formalism introduced in [Fikes and Nilsson, 1971]. This uses operators defined by schemas using precondition, add, and delete lists. As an example consider the following move-operator in a blocks world.

MOVE(x, y, z)
  PRE: CLEAR(x), ON(x, y), CLEAR(z)
  ADD: CLEAR(y), ON(x, z)
  DEL: CLEAR(z), ON(x, y)

In TL's notation the same is expressed as

$$\text{CLEAR}(x) \,\&\, \text{ON}(x,y) \,\&\, \text{CLEAR}(z) \Rightarrow \text{CLEAR}(x) \,\&\, \text{ON}(x,z) \,\&\, \text{CLEAR}(y)$$

The preconditions here are the same as in PRE above and the postconditions include all literals from ADD which both is true in general. In TL all preconditions are consumed, ie. "deleted", so that there is nothing like DEL. In compensation non-deleted preconditions have to be stated explicitly again in the postcondition. As mentioned in the previous section the name of the operator could optionally be integrated in the rule within the state literal not shown here for simplicity. In summary, crTL may be regarded as an approximate logical version of the STRIPS formalism.[9] In other words, TL inherits all advantages from STRIPS but as an *additional* feature it has also the expressiveness of classical logic.

5.3 TL and the fluent calculus

In [Hölldobler and Schneeberger, 1990] a classical calculus, in the meantime named *fluent calculus* (FC), has been introduced, which represents the manipulations of actions on the term level of classical logic. This is done in the tradition of representing the object-level of a calculus logically at the meta-level (see eg. [Kowalski, 1979]), illustrated with the previous blocks example.

Predicate symbols such as MOVE, CLEAR or ON become functional symbols, say *mv*, *cl* and *on*, respectively. The logical operation & is represented as a functional as well, say $\circ$. The entire action is then a formula $Action(c, a, e)$ which reads as follows.

$$c = cl(x) \circ on(x,y) \circ cl(z) \wedge a = mv(x,y,z) \wedge \\ e = cl(x) \circ on(x,z) \circ cl(y)$$

As can be seen it specifies the preconditions, the name and the effects of the action in an equational setting. The transition $Result(s, a, s')$ from the state s to the state s' caused by the action a is represented by the following formula.

$$Action(c, a, e) \wedge c \circ z = s \wedge s' = z \circ e$$

Note the variable z summarizing the part of the state not affected by the action. Planning problems are stated in this approach by a goal literal asking for a plan (ie. a sequence of actions) which transforms the initial state (a term like the ones shown) to the goal state on the basis of a theory (ie. a logic program) which describes all possible actions and specifies the properties of $\circ$ appropriately (associative, commutative, non-idempotent, neutral element). More details on FC can be found eg. in [Thielscher, 1997b].

The main advantage of FC is that it has a standard classical semantics. Further, the resulting programs can be run by any equational PROLOG system. It is not clear at this point whether these advantages outweigh the obvious disadvantages of a representational and computational nature. Any unbiased reader will agree that the last two formulas specifying the result of an action is awkward and much harder to read than the corresponding formula in TL shown further above. While appropriate interfaces for casual users may provide a remedy for this, researchers and programmers have still to work on this representational level.

The already mentioned Theorem 4.1 in [Große *et al.*, 1996] also states that crTL and the (conjunctive) fluent calculus just presented amount again to the same thing (in terms of derivability) thus indirectly providing a classical semantics to TL. Somehow one might be able to generalize the fluent calculus to model the full TL. Future practice has to determine which of the two will eventually prevail.

5.4 Situation calculus and others

The most popular formalism for representing actions is clearly the *situation calculus* [McCarthy and Hayes, 1969] described in any standard AI book such as [Russell and Norvig, 1994]. Again this is just a classical predicate calculus (with its standard semantics) which encodes the change from one situation to the next in the form of an *effect axiom* by an extra parameter s in each *fluent* predicate such as ON above. In addition one needs numerous *frame axioms*, or alternatively *successor-state axioms* [Reiter, 1991] which combine effect and frame axioms in an elegant way. One of the main attractions of all formalisms mentioned so far (ie. LCM, TL, LL, STRIPS, FC) is that *no* such additional axioms are needed at all which clearly impede the efficiency of any implementation of the situation calculus. For a more detailed comparative analysis of this drawback see [Fronhöfer, 1996].

Given that planning occurs in time it is not surprising that temporal or dynamic logics offer the potential for formalizing planning. One such approach for reasoning about plans and its properties (ie. not for planning itself) using dynamic logic is reported in [Stephan and Biundo,

[9] Section 8.2 in [Fronhöfer, 1996] gives examples which demonstrate that the correspondance is not an exact one.

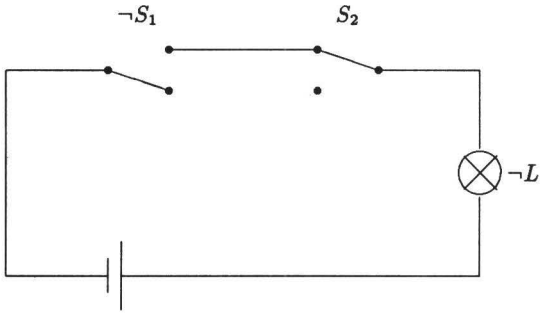


Figure 1: An electric circuit consisting of a battery, two switches, and a light bulb which shines if the two switches are in the upper position.

1993]. In addition there are numerous variants of the formalisms mentioned so far. The most noteworthy of these in terms of performance is the SATPLAN system which encodes planning within the STRIPS formalism as a satisfiability problem in classical propositional logic [Kautz *et al.*, 1996].

6 Modelling action and causality

Providing a deductive formalism for planning is one thing; a different is to show how it is used to model the various aspects underlying actions and causality. In the present section we will discuss the most important among these aspects and show how TL is able to deal with them. Although we always take the planning for illustration, it should be clear that all the other modes of reasoning discussed in the previous section are handled similarly.

6.1 Ramification

It is said that the wing-stroke of a butterfly might be the cause of a tornado somewhere else on the globe. Human reasoning is clearly unable to consider such remote consequences. Rather we assume the *law of persistence*, which states that nothing is changed by some action except for the changes caused directly by it and by the indirect but overseeable consequences from these. The problem how to compute such indirect consequences is called the *ramification problem*. Consider the electric circuit depicted in Figure 1 for illustration.[10]

We tacitly assume that there is voltage. The depicted state can formally be captured as $\neg S_1 \,\&\, S_2 \,\&\, \neg L$. Now assume we toggle S_1 by means of the action described as $\neg S_1 \Rightarrow S_1$ in TL. The resultant state $S_1 \,\&\, S_2 \,\&\, \neg L$ is inconsistent with physics which teaches that in consequence of this action also light will go on. The problem is how a planning formalism may cope with this ramification. We present here a solution which is similar in spirit to the one in [Thielscher, 1997b] but accommodated to TL's (rather than FC's) formalism.

[10]The discussion in this section closely follows [Thielscher, 1997b] from which the examples are borrowed.

Obviously, we need an additional action of the form $S_1 \,\&\, S_2 \,\&\, \neg L \Rightarrow S_1 \,\&\, S_2 \,\&\, L$ which causes light whenever both switches are on without changing the state of the switches. Further we must teach the reasoning mechanism that (i) this latter action is never activated except in consequence of toggling one of the two switches and (ii) the action is indeed activated whenever its conditions become fulfilled by toggling one or the other switch. There are many ways to implement the control of the proof mechanism specified by these two points. We describe here a rather simple one which does the trick along with a general control specification.

In this simple solution the transition rules are partitioned into primary and secondary ones. The primary rules are our action rules considered so far. The secondary ones are called *causal* rules and are characterized by the occurrence of so-called *causal* literals, denoted as L^c, among the actions preconditions. In our example both switches are causal for the light to go on. Hence the rule along with this additional control information now reads $S_1^c \,\&\, S_2^c \,\&\, \neg L \Rightarrow S_1 \,\&\, S_2 \,\&\, L$. *In the initial situation as well as in any situation resulting from a primary action all causal rules are activated whose causal conditions became true in the situation either directly by the action or indirectly by the activation of causal rules.* That is, any action (including the one "leading" to the initial situation) triggers the activation of causal actions until a stable state is reached. We refer to this as the *causality-has-preference* strategy, or CP for short.

In our example the light action is not applicable in the initial state. But once the S_1-rule is activated the causal rule follows suite according to CP so that the resulting state becomes $S_1 \,\&\, S_2 \,\&\, L$ as expected. The complete formal description of the example would thus altogether have four (primary) rules for opening and closing the switches and three causal rules, namely $\neg S_i^c \,\&\, L \Rightarrow \neg S_i \,\&\, \neg L$, $i = 1, 2$, and the one already presented.

An immediate objection to this solution might be that the transition rules will become cumbersome as the number of conditions increases. Among others it is here where our general approach pays off. Namely, we may of course abbreviate these conditions by a definition in the classical part of the formula, say $C^c \leftrightarrow S_1^c \,\&\, S_2^c$ and thus reduce any such transition rule to something like $C^c \,\&\, \neg L \Rightarrow C \,\&\, L$. Specifying the conditions, under which there will be light, can of course not fully be dispensed with. But there is a full range of varieties how this may be done. For instance, it may be broken down into a set of rules in the present case: current causes light; if (there is voltage and) the circuit is closed then there is current; if the circuit except for S_1 is closed and S_1 is closed then the entire circuit is closed; and so forth.

CP has been tested for a number of examples discussed in the literature, also for those where other methods have failed [Thielscher, 1997b]. For instance, [Ginsberg and Smith, 1988] handles already the given example incorrectly. The remedy suggested in [Lifschitz, 1990] is insufficient either as the extended circuit depicted in Figure 2

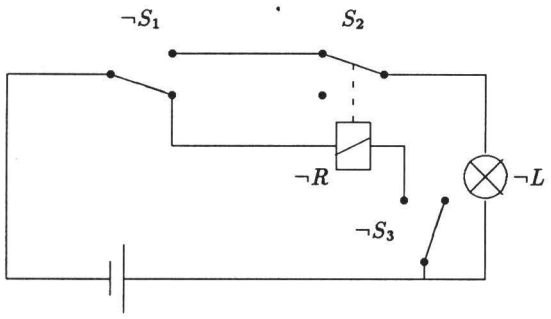


Figure 2: An extended electric circuit, which includes a third switch, S_3, and a relay, R, which attracts switch S_2 if activated.

demonstrates.

The depicted state can formally be captured as $\neg S_1 \,\&\, S_2 \,\&\, \neg S_3 \,\&\, \neg R \,\&\, \neg L$. Physics determines the following rules. $\neg S_1^c \,\&\, S_3^c \,\&\, \neg R \Rightarrow \neg S_1 \,\&\, S_3 \,\&\, R$, ie. the relay becomes activated; $R^c \,\&\, S_2 \Rightarrow R \,\&\, \neg S_2$, ie. the relay opens S_2; and $S_1^c \,\&\, S_2^c \,\&\, \neg L \Rightarrow S_1 \,\&\, S_2 \,\&\, L$, ie. light goes on. Closing S_1 in the current state causes light (rather than S_2) to open which suggests that S_2 be a *frame* fluent in the sense of [Lifschitz, 1990]. On the other hand, closing S_3 in the state depicted in the figure results in an activation of the relay and, in consequence, in S_2 opening which suggests that it be a *non-frame* fluent. In other words, Lifschitz' categorization of fluents does *not* work in this example. We need to *categorize the actions* into primary and secondary ones (rather than the fluents) as done in the solution presented in this section.

In [Thielscher, 1997a] an example of two coupled switches (if one is closed or opened the other follows suite) is given which demonstrates that indeed we need distinguish the preconditions of causal rules into causal and conditional ones as we did. The coupling rules are $S_1^c \,\&\, \neg S_2 \Rightarrow S_1 \,\&\, S_2$ and $\neg S_2^c \,\&\, S_1 \Rightarrow \neg S_2 \,\&\, \neg S_1$. Logically the preconditions of the two rules are identical, but only the first is triggered if S_1 is closed and only the second if S_2 is opened as desired.

6.2 Qualification

Although we have now a formalism which is capable of modelling actions and their concequences, there are still a number of additional issues to be considered before we can expect the formalism to play a crucial role in the reasoning of an intelligent agent. In the present section we will show how to deal with situations where certain transition rules normally available should reasonably not be applied.

In all previous examples we tacitly assumed that the actions are executable. In reality this assumption often depends on numerous conditions, too many in fact to be checked in detail. Switches can be brocken, circuits cut, light bulbs be damaged, and so forth. How to deal with this infinity of possibilities is known as the *qualification problem* [McCarthy, 1977]. There is a rich literature on non-monotonic reasoning coping with this problem [Bibel *et al.*, 1993]. TL opens a new way to deal with the problem. Why not using tweety again to illustrate it.

Tweety is a bird (Bt) as well as a penguin (Pt). Birds fly (F) and have wings (W). Penguins are exceptional birds in that they do not fly which we express as a transition rule in the following valid formula.

$$Bt \wedge Pt \wedge (Bx \rightarrow Fx \wedge Wx) \wedge (Px \rightarrow Bx) \wedge \\ (Px^c \,\&\, Fx \Rightarrow Px \,\&\, \neg Fx) \rightarrow \neg Ft \,\&\, Wt$$

The technique introduced in the previous section guarantees that the transition rule is triggered to achieve sort of an update in the form of a pseudo-causal effect which prevents the wrong conclusion of tweety flying.

The technique is applicable even beyond non-monotonic reasoning because we may additionally account for causal relationships. The solution is again adapted to TL from [Thielscher, 1996]'s FC using the example discussed there in great detail.

Assume we want to start the car's engine, (E). This might be prevented by a potato in the tail pipe, (T), which clogs, (C), it. In order to put the potato into the tail pipe, one must lift it which again may be prevented if it is too heavy, (H). The point of the example is twofold. First, we normally expect to be able to start the engine and to put a potato into the tail pipe unless there is reason to believe otherwise. Following McCarthy's idea we therefore additionally introduce an abnormality fluent for the two actions, ie. A_s and A_p. So the start action reads $\neg E \,\&\, \neg A_s \Rightarrow E \,\&\, \neg A_s$ and the put action $\neg T \,\&\, \neg A_p \Rightarrow T \,\&\, \neg A_p$. The clogging is a causal consequence of the potato in the tail pipe, hence we also have the causal rule $T^c \Rightarrow C$.

The second aspect illustrated by the example is that we obtain unintuitive results if we simply minimize the truth of the abnormality predicates as done in non-monotonic reasoning of a non-causal nature [Thielscher, 1996]. This is because the successful execution of an action (like putting a potato into the tail pipe) may *change* the state of our beliefs in the executability of other actions (like starting the engine). In other words, we additionally need a causal rule $C^c \,\&\, \neg A_s \Rightarrow C \,\&\, A_s$ for changing the state of abnormality under such circumstances. Similarly, we have the causal rule $H^c \,\&\, \neg A_p \Rightarrow H \,\&\, A_p$.

Among all occurring fluents $\mathcal{F}$ a subset $\mathcal{F}_a$ is singled out whose members are initially assumed not to hold, by default. In the present example we have $\mathcal{F}_a = \{T, C, H, A_s, A_p\}$ in accordance with the example's intentions. Of these as many as consistent with the available knowledge are "assumed away", ie. their negation is included in the specification of the initial state.[11]

If nothing specifically is known about abnormalities then starting the car will work by way of the start action and the given knowledge. If a potato is known to be (or by the appropriate action is put) in the tail pipe then

[11] For the precise definition of the underlying model preference see [Thielscher, 1996].

both clogging and A_s becomes a causal consequence by the corresponding causal rules. Similarly, in the case of too heavy potatoes this is prevented to happen by the causal rule which makes A_p true. In other words, with the formal model presented here the reasoning leads to the expected consequences under all circumstances.

6.3 Specificity

If you hold (H) a penny (p) and drop it, it will end up lying on the floor (F); ie. the drop action may be described as $Hx \Rightarrow Fx$. If you drop a fragile (or breakable, B) glass (g) rather than a penny, it will be broken (or destroyed, D) afterwards, ie. $Hx \,\&\, Bx \Rightarrow Fx \,\&\, Dx$. How should a system distinguish the two cases and correctly choose the latter action in the case of a glass in hand, ie. an initial state described by $Hg \,\&\, Bg$?

[Hölldobler and Thielscher, 1993; 1995] give an easy solution for this problem. *If in a given state more than one rule applies, the control of the deductive system prefers the most specific one*, referred to as the *more-specific-has-priority*, or SP, strategy. Thereby, r_1 is called *more specific* than r_2, $r_1 > r_2$, if for $r_i = A_i \Rightarrow B_i$, $i = 1, 2$, and some r-formula A_3, $A_1 = A_2 \,\&\, A_3$ (modula associativity and commutativity of &).

Since obviously $Hx \,\&\, Bx > Hx$, the system will indeed choose the right action in our illustrating example if its control is determined by SP.

6.4 Timing of actions

In order to achieve goals in a dynamic world the required actions often need to interact in a timely way. For instance, think of a table with glasses on it which needs to be carried in another corner of the room by two agents. If the persons do not lift the table simultaneously, the glasses will fall down and break. How could an appropriate timing be achieved in TL?

Similarly as in [Bornscheuer and Thielscher, 1997] we introduce a *compound* transition rule $A_1\overline{x} \,\&\, A_2\overline{x} \Rightarrow B_1\overline{x} \,\&\, B_2\overline{x}$ for any two (atomic or compound) transition rules $A_i\overline{x} \Rightarrow B_i\overline{x}$, $i = 1, 2$. A compound rule (as any rule) represents an action whose parts are carried out at the same time. To model our example, the two ends, t_1 and t_2, of the table initially are standing on the floor, $St_1 \,\&\, St_2$. The compound rule $Sx_1 \,\&\, Sx_2 \Rightarrow Hx_1 \,\&\, Hx_2$ is the most specific rule in this scenario which is applicable to the initial state. Hence the SP strategy from the previous subsection will choose it thus leading to the desired state of holding the table at both ends at the same time after lifting it simultaneously. This illustrates one solution to the question raised above.

A different solution may be obtained by introducing time explicitly and thus force actions to occur at a certain time point (eg. at the same time). Since communication of information is an action as well, we may model the negotiation and the agreement over a certain time point t_l for lifting the table ends among the two agents. In this case the lifting rule might look like $Sx \,\&\, Tt_l \Rightarrow Hx \,\&\, T(t_l + 1)$. Each agent would be provided with an initially synchronized and ticking clock, $T(x) \Rightarrow T(x + 1)$, and this way each could make sure that it lifts the table end at the agreed time.

If for other applications we need to fix the time point of states occurring *during* the execution of an action, $A\overline{x} \Rightarrow B\overline{x}$, we may break up and partition the action, for instance, into initial situation, action event, and resultant situation, $A\overline{x} \Rightarrow N\overline{x}$ and $N\overline{x} \Rightarrow B\overline{x}$ [Große, 1996]. Thereby N is a unique n-ary predicate serving as the *name* of the event occurring during the action, and n is the number of variables involved in the rule.

6.5 Indeterminism

Planning supports a more rational behavior. Yet it is a feature of live that much will remain unpredictable. Rational planning must therefore take indeterminism into account explicitly.

There are different kinds of undeterminism. No one, for instance, achieved so far to determine in a reliable way the outcome of throwing a dice in advance. Hence the throwing action has an undetermined outcome, $D \Rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6$. Similarly, we must account for our ignorance of the details of the physics determining the outcome of many actions in reality. We may think of nature as an agent whose intentions we know only up to a certain degree in these cases. As the example demonstrates, TL is able to model such examples to some extent. In a more elaborate approach one would have to integrate probabilities and possibilities into TL the way already achieved for classical logic (see Section 4.4 in [Bibel *et al.*, 1993] for more details and references).

Nature is more predictable than other agents such as human beings where probabilities might not help either. Here we must take into account even contradictory actions. For instance, if you try to open (O) a closed (C) door, $C \Rightarrow \neg C$, which some other agent simultaneously tries to keep closed from the door's other side, $C \Rightarrow C$, the compound action formalizing the two simultaneous actions, $C \,\&\, C \Rightarrow C \,\&\, \neg C$, does lead to nothing explicitly. From a practical point of view we may exclude connections within the same situation such as the one in the conclusion of this rule, and go on with the reasoning in spite of the contradiction. In fact we may even teach the system to consider either outcome of the conclusion in such a case (cf. also [Bornscheuer and Thielscher, 1997]).

6.6 Miscellaneous

It is clear that planning cannot be treated exhaustively in a single paper. In fact we believe that a number of important issues have yet to be settled before an autonomous agent built on the basis of TL (or any other formalism for that matter) will behave intelligently in a dynamic and uncertain environment. We conclude this entire section, therefore, with just mentioning a few further issues which would merit a more extensive treatment were there further space available. The list is certainly not a comprehensive one.

Deductive planning will have to take into account important features from classical planning in some way. An

example for these are *causal* and *protected links* [Russell and Norvig, 1994]. Often such features are already supplied by the deductive engine itself. For instance, a causal link is just a connection and protecting them is subsumed by any sophisticated proof strategy designed to identify a spanning mating as efficiently as possible. In particular if a causal link is the only one to achieve a certain subgoal then we have exactly what in automated deduction is known under the term ISOL reduction [Bibel, 1993]. Since reductions are not reconsidered in any deductive system, link protection is taken care there automatically.

Hierarchical planning is of crucial importance for successful applications. In our view it is a matter of structuring the knowledge base containing action descriptions and of controlling the deduction engine. For instance, at the highest level (0) we would have actions of the sort "*go from here to the train station*", $H \Rightarrow^0 T$. On the next level of detail (1) the system would try to refine the 0-level rule with 1-level rules such that altogether the same global effect is achieved, ie. $H \Rightarrow^1 I_1 \,\&\, I_1' \Rightarrow^1 I_2 \,\&\, \ldots \,\&\, I_n' \Rightarrow^1 T$. A related solution has been worked out for FC in [Eder *et al.*, 1996].

In dynamic environments any plan is bound to fail to some extent. It is therefore essential to be able to react to such failure with efficient *replanning*. This requires among other things a technique for the manipulation of partial plans [Lindner, forthcoming 1997; Wilkins *et al.*, 1994].

Autonomous agents need to carry out their planning under strict *time constraints*. A logical solution for this particular problem has been proposed in [Nirkhe *et al.*, 1994] which could be adapted to TL.

In reality we have to cope with *continuous* processes while it seems that TL could cope with discrete actions only. Modelling continuous processes within a logic has become an active area of research though. One issue concerns the partitioning of such a process into reasonable discrete parts [Herrmann and Thielscher, 1996]. Another issue concerns the integration of differential equations and their computation within a logic such as TL.

As a final point we mention that TL may also be seen as a logical version of an imperative programming language. For instance, destructive assignment $x := t$ may be modelled as the transition rule $\exists y\, \mathrm{cont}(x, y) \Rightarrow \mathrm{cont}(x, t)$, an attractive potential of TL as a future logic programming language.

7 The case for logic

Intellectics as well as computer science are general disciplines whose generic methods turn out to be applicable in many different areas. For instance, expert system technology has successfully been used in a great variety of applications. The logical approach within intellectics (as within computer science) is even one level more general than the remainder of this field. In theory, generality wins in the long run, in practice specialization always wins in the short run. These facts explain why engineers are extremely skeptical with general approaches such as the logical one. In fact, there is still some resistance even against the less general areas in intellectics or computer science. The facts also explain why the following pattern in the history of intellectics may be observed.

Some subject — eg. knowledge representation in the early seventies — gains in importance for applications. The engineers are the first to notice the need for activity and start developing specialized solutions. As soon as preliminary solutions are emerging the logicists get notice and claim their competence in the subject due to the logical generality — cf. the KR debate in the mid-seventies. Overwhelming theoretical evidence supporting the claim is provided by logicists over the subsequent years which finally leads to its acceptance (as it happened for KR in the eighties and is currently happening in the case of planning). Yet the logical approach is still not entering the engineering practice because it is believed to lack efficiency in comparison with specialized systems. Eventually, someone in automated deduction disproves this belief experimentally (see eg. [Paramasivam and Plaisted, to appear 1997] in the case of KR). Nevertheless the well established tools persist in applications for obvious reasons.

Logicists deplore the time "lost" by the multiple development of the same techniques in many disguises and dream of a state of the art were all these myriads of man-years invested into the advancement of logical tools. For each application they are bound always to come too late in competition with the specialists. This is not only because there are always more specialists than logicists, but also because the logicists face far more complex problems to be solved than the specialists due to the generality of their approach.

The pattern just described will not be changed by the complaints of the logicists. But change it will for the following reasons. The applications of AI techniques are all rather limited in scope so far. There is still no technological push towards more generally intelligent systems. Only when such a push sets in will generality count in a crucial way. Intelligence has so many aspects that no individual (group of) specialist(s) will ever be able to build systems featuring more than one or two of these aspects. Rather a truly intelligent system will require the incorporation of knowledge available only through a great many specialists which cannot be realized by the present technology of systems building.

This current technology is intrinsically functional. Any complex system consists of numerous modules which functionally depend on each other. That means that there must be at least one single person who oversees the whole, at least at some level of abstraction. This in turn means that the present technology is limited to the extent that single persons can cope with the complexity of systems. A breakthrough is needed before this barrier is overcome. It will happen by way of the logical approach since only logic features the property of *conjunctivity* (vs. functionality): A logical specification remains a specification if additional knowledge is added

conjunctively to it. This way many people may contribute their knowledge in the most natural form into a common system which would then be synthesized from the joint specification in a rather formal (ie. again logical) manner [Bibel, 1992b]. Section 6 demonstrates a bit of this nature of logic since there we have drawn from knowledge in a number of different areas of intellectics. Much progress will be needed before this may happen on a grander scale. Yet the progress made so far in the field of deduction is promising indeed, as can be seen from numerous results achieved lately.

Among these the success of the system EQP/Otter stands out which automatically found a proof for the conjecture that any Robbins algebra is Boolean, a mathematical problem which was open for more than sixty years [McCune, 1996]. From a technological point of view systems such as SETHEO [Moser *et al.*, 1997] or KoMeT [Bibel *et al.*, 1994] have even an advantage against Otter (in the case of SETHEO demonstrated by its first place in the ATP competition during CADE-96) and have thus the potential for more such striking results. Before concluding this paper we give pointers to impressive results of a different sort from the area of planning.

Section 8.3 in [Fronhöfer, 1996] describes a simple and straightforward implementation of crTL on top of SETHEO, called *linear backward chainer* (LBC). The performance of LBC/SETHEO is then compared there with a widely used specialized planning system, UCPOP [Penberthy and Weld, 1992]. In these experiments with randomly generated blocks-world problems with five to seven blocks LBC/SETHEO outperforms UCPOP by several orders of magnitude. Because TL is so close to classical logic it can take advantage from the cumulative investment in advanced deductive systems for classical logic such as SETHEO, which can serve as the logic engine in an expert system or as a theorem prover or as a planner or what have you. Note that no specialization to planning whatsoever is encoded in this approach.

Similarly, but even more impressively, [Kautz and Selman, 1996] coded one of the best specialized planning systems, Graphplan [Blum and Furst, 1995], as a propositional satisfiable problem which is then solved by one of the best complete satisfiability algorithms, TABLEAU [Crawford and Auton, 1993], or, alternatively, by the authors' Walksat, a stochastic local search algorithm for solving SAT problems. The experiments outperform any known planning system by several orders of magnitude. Note that this performance again is achieved by general logical systems which were used for a variety of other purposes. It would be interesting to see how a first-order prover such as SETHEO would fare for the specialized coding used in these experiments.

These successes show the way to future progress in AI: Invest as much as could reasonably be afforded into the advancement of deductive techniques and systems like TABLEAU, SETHEO, or Walksat; code your special problem as efficiently as possible like in Graphplan; run the deductive system of your choice. In particular let's forget about specialized programming! Rather let us program deductively; let us plan deductively; above all, let's attack the challenge of creating artificial intelligence in a logical and deductive way!

Acknowledgements. The author is indebted to his students and collaborators for their continuing support. In particular, I want to thank Bertram Fronhöfer, Steffen Hölldobler, Enno Sandner and Michael Thielscher, but also Vladimir Lifschitz for numerous discussions and suggestions concerning contents and text of this paper.

References

[Agre, 1995] P. Agre. Computational research on interaction and agency. *Artificial Intelligence Journal*, 72(1–2):1–52, 1995.

[Anderson and Belnap, 1975] A.R. Anderson and N.D. Belnap, Jr. *Entailment: The Logic of Relevance and Necessity*, volume 1. Princeton University Press, Princeton NJ, 1975.

[Bibel *et al.*, 1989] W. Bibel, L. Farinas del Cerro, B. Fronhöfer, and A. Herzig. Plan generation by linear proofs: On semantics. In D. Metzing, editor, *Proceedings GWAI'89*, pages 49–62, Berlin, 1989. Springer.

[Bibel *et al.*, 1993] Wolfgang Bibel, Steffen Hölldobler, and Torsten Schaub. *Wissensrepräsentation und Inferenz.* Vieweg Verlag, Brauschweig, 1993.

[Bibel *et al.*, 1994] W. Bibel, S. Brüning, U. Egly, and T. Rath. Towards an Adequate Theorem Prover Based on the Connection Method. In I. Plander, editor, *Proceedings of the Sixth International Conference on Artificial Intelligence and Information-Control of Robots*, pages 137–148. World Scientific Publishing Company, 1994. 1994.

[Bibel, 1986a] W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[Bibel, 1986b] W. Bibel. A deductive solution for plan generation. In J. W. Schmidt and C. Thanos, editors, *Foundations of Knowledge Base Management*, pages 413–436, Crete, 1986.

[Bibel, 1987] W. Bibel. *Automated Theorem Proving.* Vieweg Verlag, Braunschweig, 2. edition, 1987.

[Bibel, 1992a] W. Bibel. Intellectics. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence.* John Wiley, New York, 1992.

[Bibel, 1992b] W. Bibel. Towards predicative proramming. In Michael R. Lowry and Robert McCartney, editors, *Automating Software Design*, pages 405–423. AAAI Press, Menlo Park CA, 1992.

[Bibel, 1993] W. Bibel. *Deduction: Automated Logic.* Academic Press, London, 1993.

[Blum and Furst, 1995] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1636–1642, San Mateo, CA, 1995. Morgan Kaufmann.

[Bornscheuer and Thielscher, 1997] Sven-Erik Bornscheuer and Michael Thielscher. Explicit and implicit indeterminism: Reasoning about uncertain and contradictory specifications of dynamic systems. *J. of Logic Programming*, 1997.

[Brüning *et al.*, 1992] Stefan Brüning, Gerd Große, Steffen Hölldobler, Josef Schneeberger, Ute Sigmund, and Michael Thielscher. Disjunction in plan generation by equational logic programming. In A. Horz, editor, *Beiträge zum 7. Workshop Planen und Konfigurieren*, pages 18–26, St. Augustin, Germany, 1992. GI, Arbeitspapiere der GMD *723*.

[Brüning *et al.*, 1993] Stefan Brüning, Steffen Hölldobler, Josef Schneeberger, Ute Sigmund, and Michael Thielscher. Disjunction in resource-oriented deductive planning. In D. Miller, editor, *Proceedings of the International Logic Programming Symposium (ILPS)*, page 670, Cambridge MA, 1993. MIT Press. Postersession.

[Crawford and Auton, 1993] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 21–27, Menlo Park, CA, 1993. AAAI Press.

[Eder *et al.*, 1996] Kerstin Eder, Steffen Hölldobler, and Michael Thielscher. An abstract machine for reasoning about situations, actions, and causality. In R. Dyckhoff, H. Herre, and P. Schroeder-Heister, editors, *Proceedings of the International Workshop on Extensions of Logic Programming (ELP)*, pages 137–151, Berlin, Germany, 1996. Springer, LNAI *1050*.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence Journal*, 2:189–208, 1971.

[Frege, 1879] Gottlob Frege. *Begriffsschrift.* Louis Nebert, Halle, 1879.

[Fronhöfer, 1987] Bertram Fronhöfer. Linearity and plan generation. *New Generation Computing*, 5:213–225, 1987.

[Fronhöfer, 1996] Bertram Fronhöfer. *The Action-as-Implication Paradigm.* CS Press, München, 1996.

[Gallier, 1991] J. Gallier. Constructive logics. Part II: Linear logic and proof nets. Research Report PR2-RR-9, Digital Equipment Corporation, Paris, May 1991.

[Ginsberg and Smith, 1988] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence Journal*, 35(2):165–195, 1988.

[Girard, 1987] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[Große *et al.*, 1992] Gerd Große, Steffen Hölldobler, Josef Schneeberger, Ute Sigmund, and Michael Thielscher. Equational logic programming, actions and change. In K. Apt, editor, *Proceedings of the International Joint Conference and Symposium on Logic Programming*, pages 177–191, Cambridge MA, 1992. MIT Press.

[Große *et al.*, 1996] Gerd Große, Steffen Hölldobler, and Josef Schneeberger. Linear deductive planning. *Journal of Logic and Computation*, 6(2):233–262, 1996.

[Große, 1996] Gerd Große. *State Event Logic.* PhD thesis, Technical University Darmstadt, Darmstadt, Germany, 1996.

[Herrmann and Thielscher, 1996] Christoph Herrmann and Michael Thielscher. Reasoning about continuous processes. In B. Clancey and D. Weld, editors, *AAAI-96 — Proceedings of the National Conference on Artificial Intelligence*, pages 639–644, Palo Alto CA, 1996. AAAI Press.

[Hölldobler and Schneeberger, 1990] S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.

[Hölldobler and Thielscher, 1993] Steffen Hölldobler and Michael Thielscher. Actions and specificity. In D. Miller, editor, *Proceedings of the International Logic Programming Symposium*, pages 164–180, 1993.

[Hölldobler and Thielscher, 1995] Steffen Hölldobler and Michael Thielscher. Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence*, 14:99–133, 1995.

[Kautz and Selman, 1996] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-96)*, pages 1194–1201, Portland, OR, 1996. Morgan Kaufmann, San Mateo, CA.

[Kautz *et al.*, 1996] Henry Kautz, David McAllester, and Bart Selman. Encoding plans in propositional logic. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, pages 374–384, Cambridge, MA, 1996. Morgan Kaufmann, San Mateo, CA.

[Kowalski, 1979] Robert Kowalski. *Logic for Problem Solving.* North–Holland, New York, 1979.

[Kushmerick, 1996] Nicholas Kushmerick. Cognitivism and situated action: Two views on intelligent agency. *Computers and Artificial Intelligence*, 15(5):393–417, 1996.

[Laird *et al.*, 1987] J. E. Laird, A. Newell, and P. S. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence Journal*, 33(1):1–64, 1987.

[Lambek, 1958] J. Lambek. The mathematics of sentence structure. *Am. Math. Monthly*, 65, 1958.

[Letz *et al.*, 1992] Reinhold Letz, Johannes Schumann, Stephan Bayerl, and Wolfgang Bibel. SETHEO — A high-performance theorem prover for first-order logic. *Journal of Automated Reasoning*, 8(2):183–212, 1992.

[Lifschitz, 1990] V. Lifschitz. Frames in the space of situations. *Artificial Intelligence Journal*, 46:365–376, 1990.

[Lindner, forthcoming 1997] Matthias Lindner. *Interactive Planning for the Assistance of Human Agents.* PhD thesis, Techn. University Darmstadt, Germany, forthcoming 1997.

[Masseron *et al.*, 1990] M. Masseron, C. Tollu, and J. Vauzielles. Generating plans in linear logic. In *Foundations of Software Technology and Theoretical Computer Science*, pages 63–75. Springer, LNCS *472*, 1990.

[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, vol. 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1038–1044, Cambridge, CA, 1977. Morgan Kaufmann, San Mateo, CA.

[McCune, 1996] William McCune. Robbins algebras are boolean. *Association for Automated Reasoning Newsletter*, 35:1–3, 1996.

[Moser *et al.*, 1997] M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schumann, and K. Mayr. SETHEO and E-SETHEO. *Journal of Automated Reasoning*, 1997.

[Nilsson, 1991] Nils J. Nilsson. Logic and artificial intelligence. *Artificial Intelligence Journal*, 47(1–3):31–56, 1991.

[Nirkhe *et al.*, 1994] M. Nirkhe, S. Kraus, and D. Perlis. How to plan to meet a deadline between now and then. *J. of Logic and Computation*, 1994.

[Paramasivam and Plaisted, to appear 1997] M. Paramasivam and David A. Plaisted. Automated deduction techniques for classification in description logic systems. *Journal of Automated Reasoning*, to appear, 1997.

[Penberthy and Weld, 1992] J. S. Penberthy and D. Weld. Ucpop: A sound, complete, partial order planner for ADL. In *Proceedings of the International Conference of Knowledge Representation and Reasoning (KR-92)*, pages 103–114, San Mateo, CA, 1992. Morgan Kaufmann.

[Reiter, 1991] Raymond Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, New York, 1991.

[Restall, 1994] G. Restall. *On Logics without Contraction.* Phd thesis, University of Queensland, Australia, 1994.

[Russell and Norvig, 1994] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cliffs, N.J., 1994.

[Stephan and Biundo, 1993] Werner Stephan and Susanne Biundo. A new logical framework for deductive planning. In R. Bajcsy, editor, *13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 32–38, San Mateo, CA, 1993. Morgan Kaufmann.

[Thielscher, 1996] Michael Thielscher. Causality and the qualification problem. In L.C. Aiello, J. Doyle, and S. Shapiro, editors, *KR-96 — Proceedings of the International Conference of Knowledge Representation and Reasoning*, pages 51–62, San Mateo CA, 1996. KR Inc., Morgan Kaufmann.

[Thielscher, 1997a] Michael Thielscher. *Challenges for Action Theories: Solving the Ramification and Qualification Problem.* Habilitation thesis, Technical Universität Darmstadt, Darmstadt, Germany, 1997.

[Thielscher, 1997b] Michael Thielscher. Ramification and causality. *Artificial Intelligence Journal*, 89(1–2):317–364, 1997.

[Wilkins *et al.*, 1994] David E. Wilkins, Karen L. Myers, John D. Lowrance, and Leonard P. Wesley. Planning and reacting in uncertain and dynamic environments. *J. of Experimental and Theoretical Artificial Intelligence Journal*, 6:197–227, 1994.