Towards a Knowledge Compilation Map for Heterogeneous Representation Languages*

Hélène Fargier
IRIT–CNRS, Univ. Paul Sabatier
Toulouse, France
fargier@irit.fr

Pierre Marquis and **Alexandre Niveau**
CRIL–CNRS, Univ. Artois
Lens, France
{marquis,niveau}@cril.fr

Abstract

The knowledge compilation map introduced by Darwiche and Marquis takes advantage of a number of concepts (mainly queries, transformations, expressiveness, and succinctness) to compare the relative adequacy of representation languages to some AI problems. However, the framework is limited to the comparison of languages that are interpreted in a homogeneous way (formulæ are interpreted as Boolean functions). This prevents one from comparing, on a formal basis, languages that are close in essence, such as OBDD, MDD, and ADD. To fill the gap, we present a generalized framework into which comparing formally heterogeneous representation languages becomes feasible. In particular, we explain how the key notions of queries and transformations, expressiveness, and succinctness can be lifted to the generalized setting.

1 Introduction

There exist myriads of representation languages, with different capabilities; each one fits some applications, but is not suitable for others—there can be no best language. Choosing a good language for a given application is thus a fundamental problem in AI. Levesque and Brachman [1985] have shown that this problem boils down to a compromise between (computational) efficiency and expressiveness; for languages of equal expressiveness, efficiency must be balanced against succinctness [Gogic *et al.*, 1995]. The knowledge compilation map [Darwiche and Marquis, 2002] relies on these aspects to compare languages representing Boolean functions.

However, there are few works about the comparison of "heterogeneous" representation languages, based on different interpretation domains. It is well-known that ordered multi-valued decision diagrams (OMDDs) [Srinivasan *et al.*, 1990] can be transformed in polynomial time into ordered binary decision diagrams (OBDDs) [Bryant, 1986], using for example a log encoding (see Figure 1); these languages are considered somewhat equivalent. Yet, because of the specificity of existing frameworks, this "equivalence" cannot be formally stated, and its use in proofs requires much prudence—we would like to use it as an elementary brick to infer results in an automated fashion. It can be noted that when the notion of a "representation language" is used in a broad sense, e.g., in surveys of the knowledge representation or artificial intelligence domains [Brachman and Levesque, 2004; van Harmelen *et al.*, 2008; Russell and Norvig, 2010], it is only described informally; and whenever there is a need for a formal definition, its scope is restricted to the case in point.

In this paper, we propose a generalized formal framework for representation languages, including a definition of equivalence suitable for heterogeneous languages, with the long-term purpose of casting them in a single knowledge compilation map. We prove that several properties usually expected in specific maps still hold in our general view, modulo certain modifications. We begin with a formal definition of a "representation language" and of important related notions, such as completeness and sublanguage, in Section 2. Then, we show how the usual concepts used for language comparison in the knowledge compilation map can be generalized to heterogeneous languages: Section 3 deals with notions related to representation efficiency (expressiveness, succinctness, polynomial translation), and Section 4 with notions related to computational power (notably queries and transformations). Section 5 concludes the paper, and illustrates a practical use of our framework on a case study (that of MDDs and BDDs).[1]

2 Representation Language

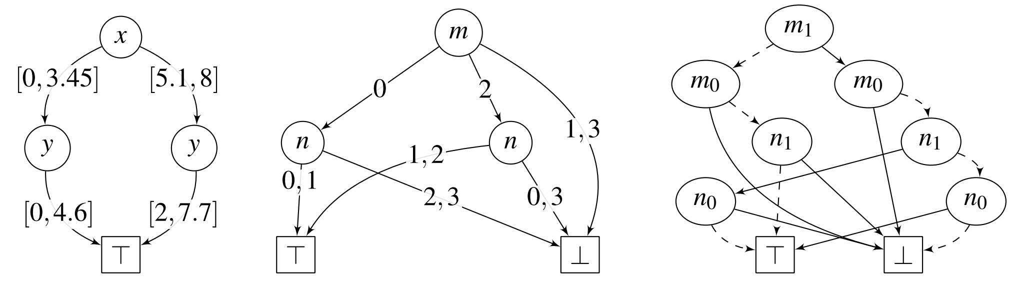
2.1 Definitions

The knowledge compilation (KC) map is built upon the notion of *language*; it was not formally defined in the original map [Darwiche and Marquis, 2002], because it implicitly had the classical meaning of a *formal* language, i.e., a set of words over an alphabet. This "language" terminology only refers to syntax; semantics is given by an implicit *interpretation function* (that of propositional formulæ). There is also a natural hierarchy on languages, namely *inclusion*: a *sublanguage*, or *fragment*, is simply a subset of a language.

In this classical setting, most features of languages are thus *implicit*, including their interpretation function and their

*This work is partially supported by the project BR4CP ANR-11-BS02-008 of the French National Agency for Research.

[1]Proofs are omitted, but an extended version is available at <url: http://www.cril.univ-artois.fr/~niveau/publis/IJCAI13_FMN.pdf>.



Variable domains:

- $x, y \in [0,10]$;
- $m, n \in \{0,1,2,3\}$;
- $m_0, m_1, n_0, n_1 \in \mathbb{B}$.

In the BDD, solid (resp. dashed) edges have the value $\top$ (resp. $\bot$).

Figure 1: An ordered interval automaton (OIA), an ordered multivalued decision diagram (OMDD), and an ordered binary decision diagram (OBDD). The OMDD is a discretization of the OIA, using partitions $\{[0,3.45],(3.45,5.1),[5.1,8],(8,10]\}$ for x and $\{[0,2),[2,4.6],(4.6,7.7],(7.7,10]\}$ for y. The OBDD is obtained from the OMDD by log encoding.

hierarchy, but also the notion of completeness and the domains of variables. This crucial limitation prevents one from directly applying this framework to a more general setting. Consider the bit sequence 10101010: interpreted as a binary-encoded natural integer, it corresponds to 170; as the "sign-and-magnitude" encoding of a relative integer, to -42; as a fixed-point real number, to 10.625; etc. These languages have the same syntax, but different interpretation functions. Our definition of a representation language builds upon that of Fargier and Marquis [2009], in that we use an explicit semantics; but we relax their assumption that it is restricted to the representation of Boolean functions, and consider languages that can represent potentially anything. For that purpose, let $\mathfrak{U}$ be our universe of discourse, containing at least all "objects" that our languages aim at representing, notably real and natural numbers, Boolean functions,[2] etc. We also use a generic alphabet Σ, that we suppose countably infinite; we call *formula* a word over this alphabet, i.e., an element of Σ^*.[3] Now, at the highest level of abstraction, a representation language is a *relation* linking formulæ and objects in the universe.

Definition 2.1 (Representation language). A *representation language* is an ordered pair $\mathtt{L} = \langle \Phi_\mathtt{L}, \mathcal{I}_\mathtt{L} \rangle$, where $\Phi_\mathtt{L} \subseteq \Sigma^*$ is a set of formulæ called the *syntax* of L, and $\mathcal{I}_\mathtt{L}$, called the *semantics* of L, is a many-to-one relation $\mathcal{I}_\mathtt{L} \subseteq \Sigma^* \times \mathfrak{U}$ that is defined at least on all formulæ in $\Phi_\mathtt{L}$.

The semantics of a language is a way of interpreting the symbols in Σ; it can be seen as a function[4] from $\Phi_\mathtt{L}$ to $\mathfrak{U}$, mapping any formula φ in the syntax of L (called an L-*representation*) to its interpretation $[\![\varphi]\!]_\mathtt{L}$. For example, the language of propositional logic can be defined as $\mathtt{PROP} = \langle \Phi_\mathtt{PROP}, \mathcal{I}_\mathtt{PROP} \rangle$, with $\Phi_\mathtt{PROP}$ the set of well-formed formulæ (with connectives restricted to $\neg$, $\vee$, and $\wedge$) and $\mathcal{I}_\mathtt{PROP}$ the usual inductive interpretation function, associating with each formula φ its corresponding Boolean function $[\![\varphi]\!]_\mathtt{PROP} \in \mathfrak{B}$.

[2]We denote as $\mathfrak{B}$ the set of Boolean functions of Boolean variables, and as $\mathfrak{B}_\mathbb{N}$ and $\mathfrak{B}_\mathbb{R}$ the sets of Boolean functions of variables having respectively an integer and a real domain.

[3]We use the Kleene star to denote unbounded Cartesian product: $S^* = \bigcup_{i \in \mathbb{N}} S^i$ (we consider words as tuples of symbols from Σ).

[4]In Def. 2.1, we use a many-to-one relation instead of a partial function to simplify later notation. The many-to-one requirement is not necessary (ambiguous semantics can be useful, a good example being natural languages) but we adopt it for the sake of simplicity.

This semantics is used in lots of languages, such as $\mathtt{CNF} = \langle \Phi_\mathtt{CNF}, \mathcal{I}_\mathtt{PROP} \rangle$ (with $\Phi_\mathtt{CNF}$ the set of formulæ in conjunctive normal form) or $\mathtt{HORN\text{-}C} = \langle \Phi_\mathtt{HORN\text{-}C}, \mathcal{I}_\mathtt{PROP} \rangle$, (with $\Phi_\mathtt{HORN\text{-}C}$ the set of Horn-CNFs). Other examples include BDD, MDD, and IA, the languages of binary and multivalued decision diagrams and of interval automata [Niveau *et al.*, 2010] (see Figure 1), and the language of algebraic decision diagrams, ADD [Bahar *et al.*, 1997]. For space reasons, the paper is focused on these languages (in particular, the relationship between the MDD and BDD families is presented in Section 5), but the framework is much more general.[5]

Remark that there can be elements $\omega \in \mathfrak{U}$ with which $\mathcal{I}_\mathtt{L}$ does not associate *any* formula, neither inside $\Phi_\mathtt{L}$ nor outside. Those elements are thus completely unrelated to the language; for example, $\mathcal{I}_\mathtt{PROP}$ ignores any object that is not a Boolean function. The elements of $\mathfrak{U}$ with which $\mathcal{I}_\mathtt{L}$ associates at least one formula are called L-*interpretations*.

Definition 2.2 (Interpretation space). The *interpretation space* of a representation language L, denoted $\Omega_\mathtt{L}$, is the codomain of $\mathcal{I}_\mathtt{L}$, i.e., $\Omega_\mathtt{L} = \{ \omega \in \mathfrak{U} \mid \exists \varphi \in \Sigma^*, \varphi \, \mathcal{I}_\mathtt{L} \, \omega \}$.

Since CNF and HORN-C have the same semantics, they also have the same interpretation space, which is $\mathfrak{B}$. Note that elements in the interpretation space of a language L are not guaranteed to have a *representation* in $\Phi_\mathtt{L}$; e.g., the syntax of HORN-C is *not expressive enough* to cover the whole interpretive power of its semantics—the language is *incomplete*.

Definition 2.3 (Completeness). A representation language L is *complete* if and only if $\forall \omega \in \Omega_\mathtt{L}, \exists \varphi \in \Phi_\mathtt{L}, [\![\varphi]\!]_\mathtt{L} = \omega$.

2.2 Sublanguages

A representation language is not simply a set of formulæ, therefore the notion of *sublanguage* cannot simply be based on a restriction of syntax, as it is in the classical KC map.

Definition 2.4 (Sublanguage). Let L and L$'$ be two representation languages; L$'$ is a *sublanguage* of L, denoted $\mathtt{L}' \subseteq \mathtt{L}$, if and only if $\Phi_{\mathtt{L}'} \subseteq \Phi_\mathtt{L}$ and $\mathcal{I}_{\mathtt{L}'} \subseteq \mathcal{I}_\mathtt{L}$. Moreover, if $\mathtt{L}' \subseteq \mathtt{L}$ and $\mathcal{I}_{\mathtt{L}'} = \mathcal{I}_\mathtt{L}$, L$'$ is said to be a *fragment* of L.

[5]It notably includes VNNF [Fargier and Marquis, 2007], and languages outside the usual scope of KC, like all flavors of constraint networks (discrete or continuous, weighted, fuzzy, etc.) [see, e.g., Rossi *et al.*, 2006], quadtrees [Finkel and Bentley, 1974], $R^\star$-trees [Beckmann *et al.*, 1990] and, with appropriate interpretation spaces, first-order logic, ontologies, programming languages, etc.

The condition that all formulæ in the sublanguage must respect the syntax of the parent language remains; but they must also respect its semantics (which was implicit in the classical framework). Note that our definition of a representation language distinguishes two kinds of sublanguage hierarchies, that of *fragments*, which orders languages of a fixed semantics, and that of the generalized *sublanguages*, in which even heterogeneous languages can be compared: e.g., it holds that $\mathtt{OBDD} \subseteq \mathtt{OMDD} \subseteq \mathtt{OIA}$, since OBDDs are specific OMDDs (restricted to Boolean variables), and OMDDs are specific OIAs (restricted to discrete variables). This natural property cannot be formally stated within the classical KC map.

It is interesting to notice that contrary to what is usually expected, incompleteness is not inherited by sublanguages. Thus, $\mathtt{OIA}$ is not complete (edge labels are restricted to closed intervals), but $\mathtt{OBDD}$ is, although $\mathtt{OBDD} \subseteq \mathtt{OIA}$. This comes from the fact that completeness is relative to the interpretation space of the language, which is not necessarily the same in a sublanguage. The expected property actually holds on fragments, which have stronger requirements.

Proposition 2.5. *Let* $\mathtt{L}$ *be a representation language, and* $\mathtt{L}'$ *a fragment of* $\mathtt{L}$*. If* $\mathtt{L}'$ *is complete, then* $\mathtt{L}$ *also is.*

2.3 Operations on Languages

In the classical KC map, the hierarchy of fragments is induced by a set of syntactic properties verified by languages: considering only read-once BDDs yields $\mathtt{FBDD}$, adding ordering yields $\mathtt{OBDD}$, etc. These can be seen as *syntactic restrictions*; they can reduce expressiveness, since the interpretation space does not change. On the other hand, the hierarchy of decision diagrams in the broad sense ($\mathtt{OBDD} \subseteq \mathtt{OMDD}$) cannot be generated by only restricting syntax, since some interpretations must be removed: it can be built using *semantic restrictions*.

Definition 2.6 (Language restrictions). The *(syntactic) restriction* of a language $\mathtt{L}$ to a set of formulæ $\Phi \subseteq \Sigma^*$ is the fragment $\mathtt{L}|_{\Phi} = \langle \Phi_{\mathtt{L}} \cap \Phi, \mathcal{I}_{\mathtt{L}} \rangle$.

The *(semantic) restriction* of a language $\mathtt{L}$ to an interpretation space $\Omega \subseteq \mathfrak{U}$ is the sublanguage $\mathtt{L}|^{\Omega} = \langle \{ \varphi \in \Phi_{\mathtt{L}} \mid [\![\varphi]\!]_{\mathtt{L}} \in \Omega \}, \mathcal{I}_{\mathtt{L}} \cap (\Sigma^* \times \Omega) \rangle$.

Using this definition, $\mathtt{HORN\text{-}C}$ is the syntactic restriction of $\mathtt{CNF}$ to Horn-CNFs, for example, while $\mathtt{BDD}$ is the semantic restriction of $\mathtt{MDD}$ to functions of Boolean variables. Another way of building a language is to *combine* existing languages.

Definition 2.7 (Union and intersection). The *union* of two languages $\mathtt{L}$ and $\mathtt{L}'$ is $\mathtt{L} \cup \mathtt{L}' = \langle \Phi_{\mathtt{L}} \cup \Phi_{\mathtt{L}'}, \mathcal{I} \cup \mathcal{I}' \rangle$, and their *intersection* is $\mathtt{L} \cap \mathtt{L}' = \langle \Phi_{\mathtt{L}} \cap \Phi_{\mathtt{L}'}, \mathcal{I} \cap \mathcal{I}' \rangle$.

This allows for example to formally state that $\mathtt{BDD} = \mathtt{MDD} \cap \mathtt{ADD}$, which expresses the fact that BDDs are both exactly the MDDs on Boolean variables, and the ADDs with two leaves.

We are now ready to define the usual concepts of the KC map: expressiveness, succinctness, and polynomial translatability, and support of queries and transformations.

3 Representation Efficiency

In the KC map, expressiveness has a fundamental role: a difference in succinctness between two languages is only significant if they have the same expressiveness. Languages with different interpretation spaces are *a fortiori* not compared in the map. Yet, there are works about translations between heterogeneous languages [Walsh, 2000; Gottlob, 1995], and it would be interesting to cast known results in the map. This calls for a more general definition of expressiveness, succinctness, and polynomial translatability, that would not require a strict equality of interpretations, but only their equivalence modulo some *semantic correspondence*, i.e., some relation linking the two heterogeneous interpretation spaces—which induces a translation between formulæ.

Definition 3.1 (Semantic correspondence, translation). A *semantic correspondence* between two interpretation spaces Ω_1 and Ω_2 is a subset of $\Omega_1 \times \Omega_2$. We will call *translation from* $\mathtt{L}_1$ *to* $\mathtt{L}_2$ a semantic correspondence $\mathcal{T} \subseteq \Omega_{\mathtt{L}_1} \times \Omega_{\mathtt{L}_2}$.

Examples of well-known translations include the various encodings of discrete constraint networks into propositional formulæ [Walsh, 2000], such as the direct encoding $\mathcal{T}_{\text{dir}}$ and the log encoding $\mathcal{T}_{\text{log}}$, which associate functions in $\mathfrak{B}_{\mathbb{N}}$ with functions in $\mathfrak{B}$; or the discretization translation $\mathcal{T}_{\text{discr}}$, which relates a continuous constraint network with the discrete constraint networks it can be discretized into. We also use the generic identity translation Id, which denotes any relation $\{\langle \omega, \omega \rangle \colon \omega \in \Omega\}$ with $\Omega \subseteq \mathfrak{U}$, in a slight abuse of notation.

3.1 Expressiveness and Succinctness

A first criterion for the comparison of representation languages is their relative expressiveness [Gogic *et al.*, 1995; Fargier and Marquis, 2008]. When they have the same semantics, expressiveness compares their ability to represent objects. However, we want to compare languages of different semantics, therefore we will define expressiveness as depending on a semantic correspondence, and similarly extend relative succinctness [Gogic *et al.*, 1995; Darwiche and Marquis, 2002], which compares the ability of languages to represent objects *compactly*.

Definition 3.2 (Expressiveness, succinctness). Let $\mathtt{L}_1$ and $\mathtt{L}_2$ be two languages, and $\mathcal{T}$ a translation from $\mathtt{L}_1$ to $\mathtt{L}_2$.

$\mathtt{L}_2$ is *at least as expressive* as $\mathtt{L}_1$ modulo $\mathcal{T}$, denoted[6] $\mathtt{L}_1 \geq_e^{\mathcal{T}} \mathtt{L}_2$, if and only if for each $\mathtt{L}_1$-representation φ_1, there exists an $\mathtt{L}_2$-representation φ_2 such that $[\![\varphi_1]\!]_{\mathtt{L}_1} \mathcal{T} [\![\varphi_2]\!]_{\mathtt{L}_2}$.

$\mathtt{L}_2$ is *at least as succinct* as $\mathtt{L}_1$ modulo $\mathcal{T}$, denoted $\mathtt{L}_1 \geq_s^{\mathcal{T}} \mathtt{L}_2$, if and only if there exists a polynomial $P(\cdot)$ such that for each $\mathtt{L}_1$-representation φ_1, there exists an $\mathtt{L}_2$-representation φ_2 such that $|\varphi_2| \leq P(|\varphi_1|)$ and $[\![\varphi_1]\!]_{\mathtt{L}_1} \mathcal{T} [\![\varphi_2]\!]_{\mathtt{L}_2}$.

Comparable expressiveness is necessary for a translation to be well-adapted to the comparison of two languages. It is clearly not a sufficient condition: for any two languages $\mathtt{L}_1$ and $\mathtt{L}_2$, the trivial translation $\mathcal{T} = \Omega_{\mathtt{L}_1} \times \Omega_{\mathtt{L}_2}$ verifies $\mathtt{L}_1 \geq_e^{\mathcal{T}} \mathtt{L}_2$, yet it does not allow one to infer any result about $\mathtt{L}_1$ and $\mathtt{L}_2$. Sufficient conditions will be studied in Section 4.

As in the classical KC framework, succinctness is a refinement of expressiveness: $\mathtt{L} \geq_s^{\mathcal{T}} \mathtt{L}' \implies \mathtt{L} \geq_e^{\mathcal{T}} \mathtt{L}'$. How-

[6]We also use (i) $\mathtt{L}_1 \leq_e^{\mathcal{T}} \mathtt{L}_2$ to mean $\mathtt{L}_2 \geq_e^{\mathcal{T}^{-1}} \mathtt{L}_1$, (ii) $\mathtt{L}_1 \sim_e^{\mathcal{T}} \mathtt{L}_2$ to mean that both $\mathtt{L}_1 \geq_e^{\mathcal{T}} \mathtt{L}_2$ and $\mathtt{L}_1 \leq_e^{\mathcal{T}} \mathtt{L}_2$, and (iii) $\mathtt{L}_1 >_e^{\mathcal{T}} \mathtt{L}_2$ to mean that $\mathtt{L}_1 \geq_e^{\mathcal{T}} \mathtt{L}_2$ but $\mathtt{L}_1 \not\leq_e^{\mathcal{T}} \mathtt{L}_2$. This notation applies, *mutatis mutandis*, to succinctness and polynomial translatability.

ever, while this implies that the classical succinctness relation, which corresponds to $\geq_s^{\mathrm{Id}}$, is not very informative when applied to heterogeneous languages, our generalization does not suffer from this drawback: it is possible to formally state succinctness results like $\texttt{MDD} \not\geq_s^{\mathcal{T}_{\mathrm{dir}}} \texttt{CNF}$. Note that we can apply the KC notation to any representation language: for example, denoting $\texttt{RADIX}_n$ the language of natural integers represented in radix n, it is well-known that $\texttt{RADIX}_1 >_s^{\mathrm{Id}} \texttt{RADIX}_2$, and that for any $n > 1$, $\texttt{RADIX}_n \sim_s^{\mathrm{Id}} \texttt{RADIX}_2$.

Note also that specific translations can be used to compare the succinctness of languages having the same interpretation space but different expressiveness. For example, languages HORN-C and $\texttt{KROM-C} = \langle \Phi_{\texttt{KROM-C}}, \mathcal{I}_{\texttt{PROP}} \rangle$, where $\Phi_{\texttt{KROM-C}}$ is the set of 2-CNF formulæ, are incomparable with respect to $\geq_e^{\mathrm{Id}}$, therefore they are also incomparable with respect to classical succinctness $\geq_s^{\mathrm{Id}}$; however, it can be interesting to know which one is the most succinct when restricted to those Boolean functions that both can represent. It is possible to express this within the generalized framework, defining an ad-hoc translation $\mathcal{T}$, for which $\texttt{HORN-C} \geq_s^{\mathcal{T}} \texttt{KROM-C}$ holds if and only if every HORN-C-representation is either not representable in KROM-C, or representable in polynomial size.[7]

3.2 Polynomial Translatability

Succinctness requires the existence of a polynomial-size translation, but not the existence of an algorithm building it, let alone its tractability. The last refinement of expressiveness is polynomial translatability [Fargier and Marquis, 2009].

Definition 3.3 (Polynomial translatability). Let $\texttt{L}_1$ and $\texttt{L}_2$ be two representation languages, and $\mathcal{T}$ a translation from $\texttt{L}_1$ to $\texttt{L}_2$; $\texttt{L}_1$ is *polynomially translatable* into $\texttt{L}_2$ modulo $\mathcal{T}$, denoted $\texttt{L}_1 \geq_p^{\mathcal{T}} \texttt{L}_2$, if and only if there exists a polynomial-time algorithm mapping any $\texttt{L}_1$-representation φ_1 to an $\texttt{L}_2$-representation φ_2 such that $[\![\varphi_1]\!]_{\texttt{L}_1} \, \mathcal{T} \, [\![\varphi_2]\!]_{\texttt{L}_2}$. Moreover, if the output is guaranteed to be *at most* polynomially *smaller* than the input, that is, if there is a polynomial $P(\cdot)$ such that for any φ_1, the output φ_2 verifies $|\varphi_1| \leq P(|\varphi_2|)$, then the translation is said to be *stable*, and we denote $\texttt{L}_1 \geq_{p\sim}^{\mathcal{T}} \texttt{L}_2$.

It holds, for example, that $\texttt{OMDD} \geq_{p\sim}^{\mathcal{T}_{\mathrm{dir}}} \texttt{OBDD}$ and $\texttt{OMDD} \geq_{p\sim}^{\mathcal{T}_{\log}} \texttt{OBDD}$ [Srinivasan *et al.*, 1990] (e.g., in Figure 1, the OBDD results from the log encoding of the OMDD). The stability condition is generally not hard to achieve; in particular, it discards special cases in which the translated formula is exponentially smaller than the original one.

In the classical KC framework, polynomial translation corresponds to $\geq_p^{\mathrm{Id}}$, and has important consequences on the satisfaction of queries and transformations. For example, denoting MODS the restriction of PROP to smooth and deterministic DNFs [Darwiche and Marquis, 2002], the fact that $\texttt{MODS} \geq_p^{\mathrm{Id}} \texttt{OBDD}$ (i.e., MODS-representations can be transformed into equivalent OBDDs in polynomial time) implies that MODS supports all queries supported by OBDD; and since $\texttt{NNF} \sim_p^{\mathrm{Id}} \texttt{PROP}$ (propositional formulæ with connectives limited to $\neg$, $\vee$, and $\wedge$ can be put in negation normal form in polynomial time), NNF and PROP support the exact same set of queries and transformations. In the latter case, the two languages are hence considered as strictly equivalent. However, these properties do not hold when a translation is used, because queries and transformations applying to one language do not necessarily apply to the other. For example, OIAs can be discretized into OMDDs (e.g., in Figure 1, the OMDD is a discretization of the OIA), i.e., $\texttt{OIA} \geq_p^{\mathcal{T}_{\mathrm{discr}}} \texttt{OMDD}$, but while OMDD supports model enumeration, OIAs generally have an infinite number of models. Conditions under which this kind of inference is possible are studied in Section 4.

3.3 Properties of Comparison Relations

Let $\geq$ denote any of the three relations we have defined, $\geq_e$, $\geq_s$, or $\geq_p$. Using the Id translation, we recover the original definition of each relation; $\geq^{\mathrm{Id}}$ is suitable only to homogeneous languages, but has the advantage of being a preorder. It is not the case for any $\mathcal{T}$, simply because interpretation spaces can be different (in which case $\geq^{\mathcal{T}}$ notably cannot be reflexive). However, when $\mathcal{T}$ is an endorelation, $\geq^{\mathcal{T}}$ inherits some of its properties.

Proposition 3.4. *Let $\Omega \subseteq \mathfrak{U}$, and $\mathcal{T} \subseteq \Omega^2$. If $\mathcal{T}$ is reflexive (resp. transitive), then $\geq^{\mathcal{T}}$ is also reflexive (resp. transitive).*

Such translations can be used to compare languages over the same interpretation space, as shown with HORN-C and KROM-C. When $\mathcal{T}$ has no remarkable property, the next proposition nevertheless expresses a kind of "pseudo-transitivity".

Proposition 3.5. *If, for some representation languages $\texttt{L}_1$, $\texttt{L}_2$, and $\texttt{L}_3$, and some semantic correspondences $\mathcal{T}$ and $\mathcal{T}'$, it holds that $\texttt{L}_1 \geq^{\mathcal{T}} \texttt{L}_2$ and $\texttt{L}_2 \geq^{\mathcal{T}'} \texttt{L}_3$, then $\texttt{L}_1 \geq^{\mathcal{T}' \circ \mathcal{T}} \texttt{L}_3$ (where $\circ$ denotes the composition of relations).*

This proposition has a useful corollary, allowing succinctness results in a given fragment hierarchy to be extended to another, granted that a "reversible" polynomial translation exists between them (see Section 5 for an example).

Corollary 3.6. *Let $\texttt{L}_1$ and $\texttt{L}_2$ (resp. $\texttt{L}_1'$ and $\texttt{L}_2'$) be two languages of interpretation space Ω (resp. Ω'). If there exists a bijective correspondence $\mathcal{T}$ between Ω and Ω' such that $\texttt{L}_1 \leq^{\mathcal{T}} \texttt{L}_1'$ and $\texttt{L}_2 \geq^{\mathcal{T}} \texttt{L}_2'$, then $\texttt{L}_1 \geq^{\mathrm{Id}} \texttt{L}_2 \implies \texttt{L}_1' \geq^{\mathrm{Id}} \texttt{L}_2'$.*

4 Computational Power

Representation languages are used to solve problems about the objects they represent. A solution to a problem is generally found thanks to an algorithm, that is, a sequence of operations on the objects; the practical implementation of the algorithm depends on the chosen representation languages.

4.1 Operations

Definition 4.1 (Semantic operation). A *semantic operation* on a universe $\Omega \subseteq \mathfrak{U}$ is a piecewise total[8] relation $\rho \subseteq \Omega^\kappa \times \mathscr{P} \times \mathscr{A}$, where $\mathscr{P} \subseteq \mathfrak{U}$ and $\mathscr{A} \subseteq \mathfrak{U}$ are sets containing *parameters* and *answers* respectively, and either $\kappa \in \mathbb{N}$ (the semantic operation is then *bounded*), or $\kappa = *$ (the semantic operation is then *unbounded*).

[7] One can define $\mathcal{T}$ as the set of pairs of Boolean functions $\langle f, g \rangle \in \mathfrak{B}^2$ for which if g has a KROM-C-representation, then $f = g$.

[8] A relation $\mathcal{R} \subseteq D_1 \times \cdots \times D_n$ is piecewise total if and only if $\forall i \in \{1, \ldots, n\}, \forall d_i \in D_i, \exists \langle r_1, \ldots, r_n \rangle \in \mathcal{R}, r_i = d_i$. Less formally, this means that its projection on each of its dimensions is total.

Let us give examples based on the KC map, with Ω the set of Boolean functions of real variables: $\Omega = \mathfrak{B}_{\mathbb{R}}$. The operation associated with the "conditioning" transformation is the partial function $\rho_{\mathbf{CD}}$ from $\Omega \times (\mathbb{R})^*$ to Ω, defined as $\rho_{\mathbf{CD}} \colon \langle f, \vec{x} \rangle \mapsto f|_{\vec{x}}$ (with $\rho_{\mathbf{CD}}$ only defined on pairs $\langle f, \vec{x} \rangle$ such that f is defined on $\vec{x}$). Here $\kappa = 1$, $\mathscr{P} = (\mathbb{R})^*$ is the set of all assignments, and $\mathscr{A} = \Omega$ is the set of Boolean functions. The operation associated with the "model extraction" query is not a function, but a relation $\rho_{\mathbf{MX}}$ associating any Boolean function f with *all* of its models; here $\kappa = 1$, $\mathscr{A} = (\mathbb{R})^*$, and $\mathscr{P}$ is ignored.[9] The operation associated with the "conjunction" transformation is the mapping $\rho_{\wedge\mathbf{C}} \colon \langle f_1, \dots, f_{n_\kappa} \rangle \mapsto \bigwedge_{i=1}^{n_\kappa} f_i$ for any $n_\kappa \in \mathbb{N}$. Here, the operation is unbounded: $\kappa = *$.

The complexity of a semantic operation $\rho \subseteq \Omega^\kappa \times \mathscr{P} \times \mathscr{A}$ depends on the representation languages considered. The most important one is the "input" language, that is, the language representing elements of Ω. Semantic operations can be much more generic than needed: the operations presented above apply to all Boolean functions of real variables, but they can also be used, in a more restricted fashion, to compare Boolean functions of Boolean variables. A semantic operation can indeed be *applied* to a subset Ω' of its universe: we denote $\rho|^{\Omega'}$ the largest (with respect to inclusion) semantic operation verifying $\rho|^{\Omega'} \subseteq \rho \cap (\Omega'^\kappa \times \mathscr{P} \times \mathscr{A})$, and $\mathscr{P}_{\Omega'}$ and $\mathscr{A}_{\Omega'}$ the sets of parameters and answers of $\rho|^{\Omega'}$, from which all "superfluous" elements have been removed. We will say that a semantic operation on $\Omega \subseteq \mathfrak{U}$ is *applicable* to a language $\mathtt{L}$ when $\Omega_{\mathtt{L}} \subseteq \Omega$ holds: for example, $\rho_{\mathbf{CD}}$ is applicable to $\mathtt{CNF}$, and then $\mathscr{P}_{\Omega_{\mathtt{CNF}}}$ is the set of Boolean assignments and $\mathscr{A}_{\Omega_{\mathtt{CNF}}}$ the set of Boolean functions of Boolean variables. We use this notion of application to define the syntactic operations that are associated with semantic operations.

Definition 4.2 (Operation). A *(syntactic) operation* is a tuple $\mathscr{O} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$ such that (i) ρ is a semantic operation: $\rho \subseteq \Omega^\kappa \times \mathscr{P} \times \mathscr{A}$; (ii) $\mathtt{L}$ is a representation language to which ρ is applicable: $\Omega_{\mathtt{L}} \subseteq \Omega$; (iii) $\mathtt{PRM}$ is a representation language covering all parameters compatible with $\mathtt{L}$: $\mathscr{P}_{\Omega_{\mathtt{L}}} \subseteq \Omega_{\mathtt{PRM}}$; (iv) $\mathtt{ANS}$ is a representation language covering all answers compatible with $\mathtt{L}$: $\mathscr{A}_{\Omega_{\mathtt{L}}} \subseteq \Omega_{\mathtt{ANS}}$.

Note how the choice of the main representation language in the syntactic operation restricts the semantic operation. When choosing $\mathtt{L}$, we discard all elements in the domain of ρ that are not $\mathtt{L}$-interpretations. Languages in an operation induce a syntactic version of its underlying semantic operation.

Definition 4.3 (Complexity of an operation). Let $\mathscr{O} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$ be a syntactic operation, with $\rho \subseteq \Omega^\kappa \times \mathscr{P} \times \mathscr{A}$. The problem associated with $\mathscr{O}$ is the following: for any tuple $\langle \varphi_1, \dots, \varphi_{n_\kappa}, \pi \rangle$ of formulæ from $\Phi_{\mathtt{L}}^\kappa \times \Phi_{\mathtt{PRM}}$, compute a formula $\alpha \in \Phi_{\mathtt{ANS}}$ such that $\langle [\![\varphi_1]\!]_{\mathtt{L}}, \dots, [\![\varphi_{n_\kappa}]\!]_{\mathtt{L}}, [\![\pi]\!]_{\mathtt{PRM}}, [\![\alpha]\!]_{\mathtt{ANS}} \rangle \in \rho$, if there exists one. The complexity of $\mathscr{O}$ is that of its associated problem.

It is the complexity of this syntactic problem that defines the complexity of the corresponding operation. Thus, "operation $\mathscr{O}$ is in polynomial time" means that there exists a polynomial-time algorithm that solves the problem associated with $\mathscr{O}$. An operation is hence a way of specifying a problem; classical *decision problems* in computability theory constitute a special case of operations. A decision problem can be modeled as $\mathscr{O} = \langle \rho_\Phi, \Sigma^*, \mathtt{bool} \rangle$, where ρ_Φ is the indicative function of a formal language Φ, and $\mathtt{bool}$ the representation language associating the symbol "0" with $\bot$ and the symbol "1" with $\top$. More generally, an operation allows one to express a *function problem* without losing the semantics.

4.2 Queries and Transformations

We recover queries and transformations as special operations by fixing some elements, namely, languages and complexity.

Definition 4.4 (Query, transformation). A *query* (resp. a *transformation*) is a tuple $\mathbf{Q} = \langle \rho, \mathtt{PRM}, \mathtt{ANS} \rangle$ (resp. $\mathbf{T} = \langle \rho, \mathtt{PRM} \rangle$) such that for any representation language $\mathtt{L}$ to which ρ is applicable, $\mathscr{O}_{\mathbf{Q},\mathtt{L}} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$ (resp. $\mathscr{O}_{\mathbf{T},\mathtt{L}} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{L} \rangle$) is an operation. Language $\mathtt{L}$ *supports* $\mathbf{Q}$ (resp. $\mathbf{T}$) if and only if $\mathscr{O}_{\mathbf{Q},\mathtt{L}}$ is in input-output polynomial time (resp. $\mathscr{O}_{\mathbf{T},\mathtt{L}}$ is in polynomial time).

For languages representing Boolean functions, denoting $\mathtt{term}$ the language of conjunctions of atoms of the form $[x = n]$ (where $x \in \Sigma$ is a variable and $n \in \mathbb{R}$), the model extraction query can be defined as $\mathbf{MX} = \langle \rho_{\mathbf{MX}}, \mathtt{term} \rangle$, the conditioning transformation can be defined as $\mathbf{CD} = \langle \rho_{\mathbf{CD}}, \mathtt{term} \rangle$, and the conjunction transformation as $\wedge\mathbf{C} = \langle \rho_{\wedge\mathbf{C}} \rangle$. These operations apply to any language representing Boolean functions, be it restricted or not to a subset of this space, e.g., to functions of Boolean or integer variables.

Support of given queries and transformations is an indicator of the absolute and relative computational power of representation languages; with our definition, the concept applies universally to any representation language. However, it should be clear that the actual list of queries and transformations useful to compare a given hierarchy of languages generally *depends* on their interpretation space. Nevertheless, it is sometimes possible to infer results about queries and transformations on a hierarchy from results on another hierarchy; e.g., support of most queries and transformations for languages in the $\mathtt{MDD}$ family can be inferred from results on the $\mathtt{BDD}$ family, because these families are polynomially equivalent modulo a "suitable" translation. We will now present sufficient conditions for this kind of inference to be possible.

4.3 Operations and Polynomial Translation

In the general case, we are interested in deducing the tractability of an operation $\mathscr{O}$ from the tractability of another, into which $\mathscr{O}$ can be translated. We start by defining the notion of translation between two semantic operations.

Definition 4.5 (Semantic operation translation). Let ρ_1 and ρ_2 be two semantic operations: $\rho_1 \subseteq \Omega_1^\kappa \times \mathscr{P}_1 \times \mathscr{A}_1$ and $\rho_2 \subseteq \Omega_2^\kappa \times \mathscr{P}_2 \times \mathscr{A}_2$. A *semantic operation translation* from ρ_1 to ρ_2 is a triple $\langle \mathcal{T}_\Omega, \mathcal{T}_\mathscr{P}, \mathcal{T}_\mathscr{A} \rangle$, where (i) $\mathcal{T}_\Omega \subseteq \Omega_1 \times \Omega_2$, (ii) $\mathcal{T}_\mathscr{P} \subseteq \mathscr{P}_1 \times \mathscr{P}_2$, and (iii) $\mathcal{T}_\mathscr{A} \subseteq \mathscr{A}_1 \times \mathscr{A}_2$, that verifies $\rho_1 = \mathcal{T}_\mathscr{A}^{-1} \circ \rho_2 \circ (\mathcal{T}_\Omega^\kappa \cdot \mathcal{T}_\mathscr{P})$, where $\circ$ (resp. $\cdot$) denotes the composition (resp. product) of relations.

We begin to see why some translations are more useful than others: for example, the trivial semantic correspondence

[9] When a semantic operation needs no parameter, we omit $\mathscr{P}$ for simplicity (we implicitly consider that it is a singleton).

$\mathcal{T} = \Omega_1 \times \Omega_2$ is not likely to be used in a semantic operation translation. The interest of having $\mathtt{L}_1 \geq_p^{\mathcal{T}} \mathtt{L}_2$ actually depends on how this specific $\mathcal{T}$ relates to the operations considered. Moreover, nothing can be deduced if the translation of parameters and answers is not tractable: we need the notion of polynomial translation between syntactic operations.

Definition 4.6 (Polynomial translation between operations). Let $\mathscr{O}_1 = \langle \rho_1, \mathtt{L}_1, \mathtt{PRM}_1, \mathtt{ANS}_1 \rangle$ and $\mathscr{O}_2 = \langle \rho_2, \mathtt{L}_2, \mathtt{PRM}_2, \mathtt{ANS}_2 \rangle$ be two operations. We say that $\mathscr{O}_1$ is *polynomially translatable* into $\mathscr{O}_2$ if and only if there exists a semantic operation translation $\langle \mathcal{T}_\Omega, \mathcal{T}_\mathscr{P}, \mathcal{T}_\mathscr{A} \rangle$ from ρ_1 to ρ_2 verifying (i) $\mathtt{L}_1 \geq_p^{\mathcal{T}_\Omega} \mathtt{L}_2$; (ii) $\mathtt{PRM}_1 \geq_p^{\mathcal{T}_\mathscr{P}} \mathtt{PRM}_2$; and (iii) $\mathtt{ANS}_1 \leq_p^{\mathcal{T}_\mathscr{A}} \mathtt{ANS}_2$. If the polynomial translation between answer languages is stable, that is, if $\mathtt{ANS}_1 \leq_{p\sim}^{\mathcal{T}_\mathscr{A}} \mathtt{ANS}_2$, then the translation between operations is said to be *answer-stable*.

This notion encompasses the familiar polynomial many-one reduction of computational complexity theory: a problem, i.e., an operation $\mathscr{O}_1 = \langle \rho_{\Phi_1}, \Sigma_1^*, \mathtt{bool} \rangle$, is polynomial-time many-one reducible to another, i.e., to an operation $\mathscr{O}_2 = \langle \rho_{\Phi_2}, \Sigma_2^*, \mathtt{bool} \rangle$, if and only if $\mathscr{O}_1$ is polynomially translatable into $\mathscr{O}_2$. In a fashion similar to reductions between problems, the tractability of an operation depends on that of operations into which it can be polynomially translated.

Theorem 4.7. *Let $\mathscr{O}_1$ and $\mathscr{O}_2$ be two operations such that $\mathscr{O}_1$ is polynomially translatable into $\mathscr{O}_2$. If $\mathscr{O}_2$ is in polynomial time, then $\mathscr{O}_1$ is in polynomial time. When the translation is answer-stable, it holds that if $\mathscr{O}_2$ is in input-output polynomial time, then $\mathscr{O}_1$ is in input-output polynomial time.*

By this theorem, $\langle \rho_{\mathbf{MX}}, \mathtt{MDD}, \mathtt{term} \rangle$, $\langle \rho_{\mathbf{CD}}, \mathtt{MDD}, \mathtt{term}, \mathtt{MDD} \rangle$, and $\langle \rho_{\wedge\mathbf{C}}, \mathtt{MDD}, \mathtt{MDD} \rangle$ are tractable, relying on the tractability of the corresponding operations on languages of the BDD family. However, the fact that OBDD supports **SFO** (the *forgetting* of a single variable) does not imply that OMDD supports **SFO**, because forgetting a multivalued variable boils down to forgetting an unbounded number of Boolean variables, which is NP-hard on OBDD. Similarly, translations that do not maintain the number of models cannot be used to infer that OMDD supports **CT** from the fact that OBDD supports **CT**. Note that the condition of answer-stability is necessary for the second statement of the theorem to hold, because the time complexity of the overall procedure depends on the size of $\mathscr{O}_2$'s answer, which can be exponential in the input.

The deductions allowed by this theorem can pertain to very disparate operations. However, in the context of a KC map, the setting is generally more restricted: typically, one has found a semantic correspondence between two interpretation spaces, inducing a polynomial translation between languages in the two hierarchies, and would like to make deductions such as "if L supports this query, then L′ also supports it"; i.e., one is interested in a *single* semantic operation applicable to both languages. This can actually be cast as a corollary of Theorem 4.7, as long as one considers only queries and transformations that are *suitable* to the given translation.

Definition 4.8 ($\mathcal{T}$-suitability). Let $\mathcal{T}$ be a semantic correspondence between some $\Omega_1 \subseteq \mathfrak{U}$ and some $\Omega_2 \subseteq \mathfrak{U}$. A query $\mathbf{Q} = \langle \rho, \mathtt{PRM}, \mathtt{ANS} \rangle$ is $\mathcal{T}$-*suitable* if and only if: (i) ρ is applicable to Ω_1 and Ω_2; (ii) there exist two semantic correspondences $\mathcal{T}_\mathscr{P}$ and $\mathcal{T}_\mathscr{A}$ such that $\langle \mathcal{T}, \mathcal{T}_\mathscr{P}, \mathcal{T}_\mathscr{A} \rangle$ is a translation from $\rho|^{\Omega_1}$ to $\rho|^{\Omega_2}$; (iii) $\mathtt{PRM}|^{\mathscr{P}\Omega_1} \geq_p^{\mathcal{T}_\mathscr{P}} \mathtt{PRM}|^{\mathscr{P}\Omega_2}$; (iv) $\mathtt{ANS}|^{\mathscr{A}\Omega_1} \leq_{p\sim}^{\mathcal{T}_\mathscr{A}} \mathtt{ANS}|^{\mathscr{A}\Omega_2}$. A transformation $\mathbf{T} = \langle \rho, \mathtt{PRM} \rangle$ is $\mathcal{T}$-*suitable* if and only if it verifies conditions i–iii, with $\mathcal{T}_\mathscr{A} = \mathcal{T}$ (the last condition does not apply, since there is no ANS language in a transformation).

Most queries and transformations of the classical KC map are suitable to direct and log encoding, including **MX**, **CD**, $\wedge$**C**, and those considered by Darwiche and Marquis [2002]—with the exception of **SFO**, for reasons stated earlier. This is particularly interesting with regard to the following result.

Theorem 4.9. *Let $\mathtt{L}_1$ and $\mathtt{L}_2$ be two representation languages, and $\mathcal{T}$ a translation from $\mathtt{L}_1$ to $\mathtt{L}_2$. If $\mathtt{L}_1 \geq_p^{\mathcal{T}} \mathtt{L}_2$, then every $\mathcal{T}$-suitable query supported by $\mathtt{L}_2$ is also supported by $\mathtt{L}_1$. If $\mathtt{L}_1 \sim_p^{\mathcal{T}} \mathtt{L}_2$, then every $\mathcal{T}$-suitable query or transformation supported by $\mathtt{L}_2$ is also supported by $\mathtt{L}_1$.*

This allows to automatically extend most known results about BDDs to MDDs, and more generally, most results from the Boolean KC map to languages over non-Boolean variables. Note however that results are not exactly the same, as they depend on the suitability of each query and transformation to the translation considered; thus, OMDD does not support **SFO** [Amilhastre *et al.*, 2012], even though OBDD does.

5 Conclusion

In this paper, we have presented a framework for comparing representation languages. While taking as few hypotheses as possible about what constitutes an admissible representation language, we showed how the usual concepts of the knowledge compilation map could be adapted to this broader setting, allowing the comparison of heterogeneous languages with respect to their representation efficiency and their computational capabilities, and the exploitation of known results in the Boolean case to numerous languages, over discrete or continuous variables and non-Boolean valuation.

As an illustration of the latter point, let us take the simple example of the "bounded MDD" family: we define the k-MDD language as the restriction of MDD to variables with domains of cardinality k, and its fragments k-FMDD, k-OMDD, and k-$\mathtt{OMDD}_<$ as its restriction to read-once, ordered, and $<$-ordered diagrams [Darwiche and Marquis, 2002], respectively.

Proposition 5.1. *It holds that k-MDD $<_s$ k-FMDD $<_s$ k-OMDD $<_s$ k-$\mathtt{OMDD}_<$, and each of the queries and transformations considered by Darwiche and Marquis [2002] is supported by k-MDD (resp. k-FMDD, k-OMDD, k-$\mathtt{OMDD}_<$) if and only if it is supported by* BDD *(resp.* FBDD, OBDD, $\mathtt{OBDD}_<$*).*

This proposition follows directly from the fact that k-MDD $\sim_p^{\mathcal{T}_k}$ BDD, k-FMDD $\sim_p^{\mathcal{T}_k}$ FBDD, k-OMDD $\sim_p^{\mathcal{T}_k}$ OBDD, and k-$\mathtt{OMDD}_<$ $\sim_p^{\mathcal{T}_k}$ $\mathtt{OBDD}_<$, denoting $\mathcal{T}_k$ the restriction of $\mathcal{T}_{\text{dir}}$ to variables with domains of cardinality k, by applying Corollary 3.6 ($\mathcal{T}_k$ is bijective) and Theorem 4.9 (all queries and transformations defined by Darwiche and Marquis [2002] are $\mathcal{T}_k$-suitable), respectively.

This is a first step towards a generalized knowledge compilation map, in which heterogeneous language hierarchies could be presented in a unified way.

References

[Amilhastre *et al.*, 2012] Jérôme Amilhastre, Hélène Fargier, Alexandre Niveau, and Cédric Pralet. Compiling CSPs: A complexity map of (non-deterministic) multivalued decision diagrams. In *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1–8, 2012.

[Bahar *et al.*, 1997] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2/3):171–206, 1997.

[Beckmann *et al.*, 1990] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD Conference*, pages 322–331, 1990.

[Brachman and Levesque, 2004] Ronald J. Brachman and Hector J. Levesque. *Knowledge Representation and Reasoning*. Elsevier, 2004.

[Bryant, 1986] Randall E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)*, 17:229–264, 2002.

[Fargier and Marquis, 2007] Hélène Fargier and Pierre Marquis. On valued negation normal form formulas. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 360–365, 2007.

[Fargier and Marquis, 2008] Hélène Fargier and Pierre Marquis. Extending the knowledge compilation map: Krom, Horn, affine and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 442–447, 2008.

[Fargier and Marquis, 2009] Hélène Fargier and Pierre Marquis. Knowledge compilation properties of trees-of-BDDs, revisited. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 772–777, 2009.

[Finkel and Bentley, 1974] Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[Gogic *et al.*, 1995] Goran Gogic, Henry A. Kautz, Christos H. Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 862–869, 1995.

[Gottlob, 1995] Georg Gottlob. Translating default logic into standard autoepistemic logic. *Journal of the ACM*, 42(4):711–740, 1995.

[Levesque and Brachman, 1985] Hector J. Levesque and Ronald J. Brachman. A fundamental tradeoff in knowledge representation and reasoning (revised version). In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 41–70. Kaufmann, Los Altos, CA, 1985.

[Niveau *et al.*, 2010] Alexandre Niveau, Hélène Fargier, Cédric Pralet, and Gérard Verfaillie. Knowledge compilation using interval automata and applications to planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 459–464, 2010.

[Rossi *et al.*, 2006] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[Russell and Norvig, 2010] Stuart J. Russell and Peter Norvig. *Artificial Intelligence — A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

[Srinivasan *et al.*, 1990] Arvind Srinivasan, Timothy Kam, Sharad Malik, and Robert K. Brayton. Algorithms for discrete function manipulation. In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, pages 92–95, November 1990.

[van Harmelen *et al.*, 2008] Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors. *Handbook of Knowledge Representation*, volume 1. Elsevier Science, 2008.

[Walsh, 2000] Toby Walsh. SAT v CSP. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, pages 441–456, 2000.