Comparing Space Efficiency of Propositional Knowledge Representation Formalisms

Marco Cadoli, Francesco M. Donini, Paolo Liberatore, Marco Schaerf
Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
Via Salaria 113, I-00198, Roma, Italy
{cadoli|donini|liberatore|schaerf}@dis.uniroma1.it

Abstract

We investigate the *space efficiency* of a Propositional Knowledge Representation (PKR) formalism. Informally, the space efficiency of a formalism F in representing a certain piece of knowledge α, is the size of the shortest formula of F that represents α. In this paper we assume that knowledge is either a set of propositional interpretations or a set of formulae (theorems). We provide a formal way of talking about the relative ability of PKR formalisms to compactly represent a set of models or a set of theorems. We introduce two new compactness measures, the corresponding classes, and show that the relative space efficiency of a PKR formalism in representing models/theorems is directly related to such classes. In particular, we consider formalisms for nonmonotonic reasoning, such as circumscription and default logic, as well as belief revision operators.

1 INTRODUCTION

Motivations. During the last years a large number of formalisms for knowledge representation (KR) have been proposed in the literature. Given some (informal) knowledge of a domain, a knowledge engineer has to choose the most appropriate KR formalism to represent it. Formalisms can be chosen for their semantics, the complexity of inference, or other properties. Here we investigate their *space efficiency*. Informally, the space efficiency of a formalism F in representing a certain piece of knowledge α, is the size of the shortest formula of F that represents α. Space efficiency –also called *succinctness* or *compactness*– of a formalism can be measured wrt its ability to represent various forms of knowledge. In this paper we focus on propositional KR (PKR) formalisms, and assume that knowledge is either a set of propositional interpretations or a set of formulae (theorems). Consequently, we consider a PKR reasoning problem to be either model checking, or theorem proving.

Conceptually similar investigations are made in the field of relational database query languages. In that field one of the goals is to know exactly what information it is possible to extract from a database by means of a query. The typical result of such investigations characterizes the *expressiveness* of a query language, often saying that language A captures more/less/same queries than/as language B. Sometimes such results are conditional to non-collapse of some complexity classes.

State of the Art. Most PKR formalisms are "translatable" one into another, although such translation may lead to an exponential increase of the size of the formula. Most of the translations have Propositional Logic (PL) as their target formalism. E.g., [BED91] from default logic to PL, [BED94] from disjunctive logic programs to PL, [Win89] from revised knowledge bases to PL, [GPP89] from circumscription to PL.

Only very recently researchers started analyzing the space efficiency of PKR formalisms; this kind of investigation includes questions such as "is exponential increase of the above mentioned translations intrinsic, or is it possible to design a polynomial-size translation?" In [CDS95] it was shown that many interesting fragments of default logic and circumscription cannot be expressed by polynomial-time fragments of PL without super-polynomially increasing the size of formulae. It was proven that super-polynomial increase of the size is necessary when translating unrestricted propositional circumscription [CDSS95] and most operators for belief revision into PL [CDLS95, Lib95]. In [GKPS95] Gogic, Kautz, Papadimitriou and Selman analyzed the relative succinctness of several PKR for-

malisms in representing sets of models. Among other results, they showed that skeptical default logic represents sets of models more succinctly than circumscription. Unfortunately, all the above results are based on ad-hoc proofs and do not help us to define equivalence classes for the space efficiency of KR formalisms.

In a recent paper [CDLS96], we have introduced a new complexity measure for decision problems, called *compilability*. In the present paper we show how this new measure can be directly used to characterize the space efficiency of PKR formalisms.

Goal. In KR the notion of *polynomial-time solvability* models the concept of *tractable* reasoning problem. Analogously, the notion of *polynomial many-one reducibility* models the relation existing between two reasoning problems whose time complexity is comparable. The latter notion allows one to say, e.g., that inference in PL is one of the hardest problems among those in coNP. Our goal is to provide a formal way of talking about the relative ability of PKR formalisms to compactly represent information, where the information is either a set of models or a set of theorems. In particular, we would like to be able to say that a specific PKR formalism provides "one of the most compact ways to represent models/theorems" among the PKR formalisms of a specific class.

Results. We introduce two new compactness measures (*model* and *theorem compactness*) and the corresponding classes (model-C and thm-C, where C is a complexity class like P, NP, coNP, etc.). Such classes form two hierarchies that are isomorphic to the polynomial-time hierarchy [Sto76]. We show that the relative space efficiency of a PKR formalism is directly related to such classes. In particular, the ability of a PKR formalism to compactly represent sets of models/theorems is directly related to which classes of the model/theorem hierarchy it belongs to. Problems higher up in the model/theorem hierarchy can represent sets of models/theorems more compactly than formalisms that are in lower classes.

This classification is obtained through a general framework, and not by making direct comparisons, and ad-hoc proofs, between the various PKR formalisms. Furthermore, our approach also allows for a simple and intuitive notion of completeness for both model and theorem hierarchies. This notion precisely characterizes both the relation between formalisms at different levels, and the relations between problems at the same level. An interesting result is that two PKR formalisms in which model checking or inference belong to the same time complexity may belong to different compactness classes. This may suggest a criterion for choosing between two PKR formalisms in which reasoning has the same time complexity—namely, choose the more compact one. Also, two PKR formalisms may belong to the same theorem compactness class, yet to different model compactness classes. This stresses the importance of first clarifying whether one wants to represent models or theorems when choosing a PKR formalism.

Outline. In the next section we briefly recall some notions on non-uniform computation that are important for what follows. In Section 3 we give basic definitions and assumptions about PKR formalisms, and we propose two general definitions about translations between formalisms. In Section 4 we recall compilability classes from [CDLS96]. In Section 5 we show how compilability classes can be used to compare the space efficiency of PKR formalisms, and in Section 6 we actually compare many known PKR formalisms using our framework. Finally, some conclusions are drawn.

2 NON-UNIFORM COMPUTATION

We assume the reader is familiar with basic complexity classes, such as P, NP and (uniform) classes of the polynomial hierarchy (see, for example, [GJ79]). Here we just briefly introduce non-uniform classes, following Johnson [Joh90].

Definition 1 *An* advice-taking Turing machine *is a Turing machine that has associated with it a special "advice oracle" A, which can be any function (not necessarily a recursive one). On input s, a special "advice tape" is automatically loaded with $A(|s|)$ and from then on the computation proceeds as normal, based on the two inputs, x and $A(|s|)$.*

Note that the advice is only function of the *size* of the input, not of the input itself.

Definition 2 *An advice-taking Turing machine uses* polynomial advice *if its advice oracle A satisfies $|A(n)| \leq p(n)$ for some fixed polynomial p and all non-negative integers n.*

Definition 3 *If* C *is a class of languages defined in terms of resource-bounded Turing machines, then* C/poly *is the class of languages defined by Turing machines with the same resource bounds but augmented by polynomial advice.*

Any class C/poly is also known as *non-uniform* C, where non-uniformity is due to the presence of the advice. Non-uniform and uniform complexity classes are related in [KL80, Yap83]. In particular, Karp and

Lipton proved in [KL80] that if NP $\subseteq$ P/poly then $\Pi_2^p = \Sigma_2^p$ = PH, i.e., the polynomial hierarchy collapses at the second level, while Yap in [Yap83, pg. 292 and Theorem 2] generalized their results showing that if NP $\subseteq$ coNP/poly then $\Pi_3^p = \Sigma_3^p$ = PH, i.e., the polynomial hierarchy collapses at the third level. Such a collapse is considered very unlikely by most researchers in structural complexity.

3 PROPOSITIONAL KR FORMALISMS

We consider a finite alphabet of propositional symbols $L = \{a, b, c, \ldots\}$, possibly with subscripts. We admit connectives of fixed arity for constructing well-formed formulae, as an example, $\wedge$ is the standard binary conjunction, the symbol for defaults $\frac{:}{}$ of default logic [Rei80] is a ternary connective, etc. Since we restrict to propositional formalisms, we admit neither variable symbols, nor quantifiers. We distinguish between two kinds of formulae: *knowledge bases* and *queries*. The languages describing well-formed knowledge bases and well-formed queries may be different within the same formalism, e.g., this is the case for default logic, or logic programming. An *interpretation* for L is a mapping from L in $\{\mathsf{true}, \mathsf{false}\}$. A *model-theoretic semantics* for a formalism is a description of how to extend an interpretation for L to well-formed knowledge bases and queries. Observe that the semantics of a formula needs not to be obtained in an easy way from the semantics of its components and connectives: E.g., for skeptical default logic an interpretation I assigns true to a default theory $\langle D, W\rangle$ iff there is a (propositional) extension of $\langle D, W\rangle$ such that I assigns true to all of its formulae. A *model* of a knowledge base KB in a formalism F is an interpretation M that maps KB to true (written $M \models_F KB$), and similarly for a query Q (written $M \models_F Q$). Sometimes models will be denoted as sets of letters which are mapped into true. A *proof theory* of a formalism F is a definition of which queries are derivable from which knowledge bases in F. When a query Q is derivable from a knowledge base KB in F, we call Q a *theorem* of KB (written $KB \vdash_F Q$). Observe that some formalisms have only a proof theory, with no model-theoretic semantics, e.g., credulous default logic. When a formalism F has both a model-theoretic semantics and a proof theory, we impose the usual relation between them: $KB \vdash_F Q$ iff $\forall M : M \models_F KB$ implies $M \models_F Q$. When F is classical propositional logic PL, we omit the subscript from $\vdash$ and $\models$.

Assumption 1. Throughout the paper, we assume that *a knowledge base in a PKR formalism is used to represent either its set of models, or its set of theorems, or both.*

In this way, all PKR formalisms can be compared on the basis of which sets of models or theorems they can represent, and how succinct is the representation.

Assumption 2. As for queries, *we consider only queries whose size is less than or equal to that of the knowledge base.*

As a consequence, propositional tautologies whose size is larger than that of the knowledge base are not considered among the theorems.

3.1 REDUCTIONS AMONG KR FORMALISMS

We now define the forms of reduction between PKR formalisms that we analyze in the following sections. A formula can always be represented as a string over an alphabet Σ, hence from now on we consider translations as functions transforming strings. We assume that Σ contains a special symbol # to denote blanks. The *length* of a string $x \in \Sigma^*$ is denoted by $|x|$.

A function f is called *poly-size* if there exists a polynomial p such that for all x it holds $|f(x)| \leq p(|x|)$. Moreover, a function g is called *poly-time* if there exists a polynomial q such that for all x, $g(x)$ can be computed in time less than or equal to $q(|x|)$. These definitions extend to binary functions considering the sum of the sizes of their arguments.

Let F_1 and F_2 be two PKR formalisms. There exists a *poly-size reduction* from F_1 to F_2, denoted as $f : F_1 \mapsto F_2$, if f is a poly-size function such that for any given knowledge base KB in F_1, $f(KB)$ is a knowledge base in F_2.

Clearly some restrictions must be imposed on such functions. In particular, we define a form of reduction that preserves the models of the original theory:

Definition 4 (Model Preservation) *A poly-size reduction $f : F_1 \mapsto F_2$ satisfies* model-preservation *if for each knowledge base KB in F_1 there exists a poly-time function g_{KB} such that for every interpretation M of the variables of KB it holds that $M \models_{F_1} KB$ iff $g_{KB}(M) \models_{F_2} f(KB)$.*

Example We reduce a fragment of skeptical default logic to circumscription with varying letters, using the transformation shown in [Eth87]. Let $\langle D, W\rangle$ be a *prerequisite-free normal* (PFN) default theory, i.e., all defaults are of the form $\frac{:\gamma}{\gamma}$, where γ is a generic formula. Let Z be the set of letters occurring in $\langle D, W\rangle$. Define P_D as the set of letters $\{a_\gamma | \frac{:\gamma}{\gamma} \in D\}$.

The function f can be defined in the following way: $f(\langle D, W\rangle) = CIRC(T; P_D; Z)$, where $T = W \cup \{a_\gamma \equiv \neg\gamma | a_\gamma \in P_D\}$, P_D are the letters to be minimized, and Z (the set of letters occurring in $\langle D, W\rangle$) are varying letters. We show that f is a model-preserving poly-size reduction. In fact, given a set of PFN defaults D let g_D be a function such that for each interpretation M for Z, $g_D(M) = M \cup \{a_\gamma \in P_D | M \models \neg\gamma\}$. Clearly, f is poly-size, g_D is poly-time, and M is a model of at least one extension of $\langle D, W\rangle$ iff $g_D(M) \models CIRC(T; P_D; Z)$. The subscript D of g_D stresses the fact that the function g_D does not depend in this case on the whole knowledge base $\langle D, W\rangle$, but just on D.

A weaker form of reduction is the following one, where only theorems are preserved:

Definition 5 (Theorem Preservation) *A poly-size reduction $f : F_1 \mapsto F_2$ satisfies* theorem-preservation *if for each knowledge base KB in F_1 there exists a poly-time function g_{KB} such that for every query Q on the variables of KB it holds that $KB \vdash_{F_1} Q$ iff $f(KB) \vdash_{F_2} g_{KB}(Q)$.*

An example of theorem-preserving (and non-model-preserving) poly-size reduction from updated knowledge bases to PL is given in [Win89]. The reduction shown in the previous example is also theorem-preserving, using for g_{KB} the identity function.

We remark that our definitions of reduction are more general than those proposed in [GKPS95]. In particular, in [GKPS95] only a notion analogous to Definition 4 is considered, and only for the case when g_{KB} is the identity – i.e., models in the two formalisms should be identical.

4 COMPILABILITY CLASSES

In this section we summarize some definitions and results proposed in [CDLS96] adapting them to the context and terminology of PKR formalisms. In that paper we introduced a new complexity measure for decision problems, called *compilability*. Following the intuition that a knowledge base is known well before questions are posed to it, we divide a reasoning problem into two parts: one part is *fixed* or *accessible off-line* (the knowledge base), and the second one is *variable*, or *accessible on-line* (the model/query). Compilability aims at capturing the *on-line complexity* of solving a problem composed of such inputs, i.e., complexity wrt the second input when the first one can be preprocessed in an arbitrary way. In the next section we show the close connection between compilability and the space efficiency of PKR formalisms.

Figure 1: The C/poly Machine

We define a *language of pairs* S as a subset of $\Sigma^* \times \Sigma^*$. This is necessary to represent the two inputs to a PKR reasoning problem, i.e., the knowledge base (KB), and the query or interpretation. As an example, the problem CNF CLAUSE INFERENCE (CI) is defined as

$$\text{CI} = \{\langle x, y\rangle \mid x \text{ is a CNF propositional formula, } y \text{ is a clause and } x \vdash y\}$$

It is well known that CI is coNP-complete wrt the sum of the size of both inputs, i.e., it is one of the "hardest" problems among those belonging to coNP. Our goal is to prove that CI is the "hardest" theorem-proving problem among those that can be solved preprocessing in an arbitrary way the first input, i.e., the KB. To this end, we introduce a new hierarchy of classes, the *non-uniform compilability classes*, denoted as nu-comp-C, where C is a generic uniform complexity class, such as NP, coNP, Σ_2^p, ...

Definition 6 (nu-comp-C **Classes)** *A language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ belongs to* nu-comp-C *iff there exists a binary poly-size function f and a language of pairs S' such that for all $\langle x, y\rangle \in S$ it holds:*

1. $\langle f(x, |y|), y\rangle \in S'$ *iff* $\langle x, y\rangle \in S$;
2. $S' \in$ C.

Notice that the poly-size function f takes as input both x (the KB) and the size of y (the query or interpretation). If we want to rewrite off-line x into a new string (formula), our definition requires that we know in advance the *size* of y. If y is an interpretation of the letters of x, its size is bounded by $|x|$ and therefore it is known in advance. However, if y is a query it is not very reasonable to assume that, at compilation time, we know the size of y. Anyway, the only information necessary at compilation time is an *upper bound* of the size of all y's that can occur. For this reason, we assume that the size of each query is less than or equal to the size of the knowledge base (cf. Assumption 2).

For each C, the class nu-comp-C generalizes the non-uniform class C/poly —i.e., C/poly $\subset$ nu-comp-C — by allowing for a fixed part x. In Figures 1 and 2 we compare the machines corresponding to C/poly and nu-comp-C.

We introduce now a reduction between problems.

Figure 2: The nu-comp-C Machine

Definition 7 (Non-uniform comp-reducibility) *Given two problems A and B, A is* non-uniformly comp-reducible *to B (denoted as $A \leq_{nu-comp} B$) iff there exist two poly-size binary functions f_1 and f_2, and a polynomial-time binary function g such that for every pair $\langle x, y\rangle$ it holds that $\langle x, y\rangle \in A$ if and only if $\langle f_1(x, |y|), g(f_2(x, |y|), y)\rangle \in B$.*

The $\leq_{nu-comp}$ reductions can be pictorially represented as follows:

Such reductions satisfy all important properties of a reduction:

Theorem 1 *The reductions $\leq_{nu-comp}$ satisfy transitivity and are compatible (in the sense of Johnson [Joh90, pg. 79]) with the class* nu-comp-C *for every complexity class* C.

Therefore, it is possible to define the notions of *hardness* and *completeness* for nu-comp-C for every complexity class C.

Definition 8 (nu-comp-C**-completeness)** *Let S be a language of pairs and* C *a complexity class. S is* nu-comp-C-hard *iff for all problems $A \in$* nu-comp-C *we have that $A \leq_{nu-comp} S$. Moreover, S is* nu-comp-C-complete *if S is in* nu-comp-C *and is* nu-comp-C-hard.

We now have the right complexity class to completely characterize the problem CI. In fact CI is nu-comp-coNP-complete. Furthermore, the hierarchy formed by the compilability classes is proper if and only if the polynomial hierarchy is proper [CDLS96, KL80, Yap83] (which is widely conjectured to be true).

We close the section by giving a rationale for the complexity classes we defined. Informally we may say that nu-comp-NP-hard problems are "not compilable to P", as from the above considerations we know that if there exists a preprocessing of their fixed part that makes them on-line solvable in polynomial time, then the polynomial hierarchy collapses. The same holds for nu-comp-coNP-hard problems. In general, a problem which is nu-comp-C-complete for a class C containing P can be regarded as the "toughest" problem in C, even after arbitrary preprocessing of the fixed part. On the other hand, a problem in nu-comp-C is a problem that, after preprocessing of the fixed part, becomes a problem in C (i.e., it is "compilable to C").

5 SUCCINCTNESS OF PKR FORMALISMS

In this section we show how to use our compilability classes to compare the succinctness of PKR formalisms.

Let F_1 and F_2 be two formalisms representing sets of models. We prove in this section that any knowledge base α in F_1 can be reduced, via a poly-size reduction, to a knowledge base β in F_2 satisfying model-preservation if and only if the compilability class of the problem of *model checking* (first input: KB, second input: interpretation) in F_1 is higher than or equal to the compilability class of the problem of model checking in F_2.

Similarly, we prove that theorem-preserving poly-size reductions exist if and only if the compilability class of the problem of *inference* (first input: KB, second input: query, cf. definition of CI) in F_1 is higher than or equal to the compilability class of the problem of inference in F_2.

In order to simplify the presentation and proof of the theorems we introduce some definitions.

Definition 9 (Model/Theorem hardness/completeness) *Let F be a PKR formalism. If the problem of model checking for F belongs to the compilability class* nu-comp-C*, we say that F is in* model-C*. Similarly, if model checking is* nu-comp-C*-complete (hard), we say that F is* model-C*-complete (hard). If the problem of inference for the formalism F belongs to the compilability class* nu-comp-C*, we say that F is in* thm-C*. Similarly, if inference is* nu-comp-C*-complete (hard), we say that F is* thm-C*-complete (hard).*

These definitions implicitly define two hierarchies, the model hierarchy (model-C) and the theorem hierarchy (thm-C). As an example rephrased from [CDS95, Thm. 2], we characterize model and theorem classes of propositional logic (PL).

Theorem 2 PL *is in* model-P *and* thm-coNP*-complete.*

We can now formally establish the connection between succinctness of representations and compilability classes. In the following theorems, the complexity classes C, C_1, C_2 belong to the polynomial hierarchy [Sto76]. In Theorems 4 and 6 we assume that the polynomial hierarchy does not collapse.

Theorem 3 *Let F_1 and F_2 be two PKR formalisms. If F_1 is* model-C*-complete and F_2 is* model-C*-hard, then there exists a poly-size reduction $f : F_1 \mapsto F_2$ satisfying model preservation.*

Proof. (sketch) Recall that since F_1 is model-C-complete, model checking in F_1 is in nu-comp-C, and since F_2 is model-C-hard, model checking in F_1 is non-uniformly comp-reducible to model checking in F_2. That is, (adapting Def. 7) there exist two poly-size binary functions f_1 and f_2, and a polynomial-time binary function g such that for every pair $\langle KB, M\rangle$ it holds that $M \models_{F_1} KB$ if and only if $g(f_2(KB, |M|), M) \models_{F_2} f_1(KB, |M|)$. Now observe that $|M|$ can be computed from KB by simply counting the letters appearing in KB; let f_3 be such a counting function, i.e., $|M| = f_3(KB)$. Clearly, f_3 is poly-size. Define the reduction f as $f(KB) = f_1(KB, f_3(KB))$. Since poly-size functions are closed under composition, f is poly-size. We show that f is a model-preserving reduction. In fact, given KB, one can compute $z = f_2(KB, |M|) = f_2(KB, f_3(KB))$. Since f_2 and f_3 are poly-size, z has polynomial size wrt $|KB|$, hence wrt $|M|$. Define $h_{KB}(M) = g(z, M)$. Clearly, h_{KB} is a poly-time function, and from its construction, $M \models_{F_1} KB$ iff $h_{KB}(M) \models_{F_2} f(KB)$. □

Theorem 4 *Let F_1 and F_2 be two PKR formalisms. If the polynomial hierarchy does not collapse, F_1 is* model-C_1*-hard, F_2 is in* model-C_2*, and $C_2 \subset C_1$, then there is no poly-size reduction $f : F_1 \mapsto F_2$ satisfying model preservation.*

Proof. (sketch) We show that if such a reduction exists, then C_1/poly $\subseteq$ C_2/poly and the polynomial hierarchy collapses at some level (see [Yap83]). Let A be a complete problem for class C_1, e.g., if C_1 is Σ_3^p then A may be validity of $\exists\forall\exists$-quantified boolean formulae [Sto76]. Define the problem ϵA as:

$$\epsilon A = \{\langle x, y\rangle \mid x = \epsilon \text{ (the empty string) and } y \in A\}$$

From [CDLS96, Thm. 6], ϵA is nu-comp-C_1-complete. Since model checking in F_1 is model-C_1-hard, ϵA is non-uniformly comp-reducible to model checking in F_1. That is, (adapting Def. 7) there exist two poly-size binary functions f_1 and f_2, and a polynomial-time binary function g such that for every pair $\langle \epsilon, y\rangle$, $\langle \epsilon, y\rangle \in \epsilon A$ if and only if $g(f_2(\epsilon, |y|), y) \models_{F_1} f_1(\epsilon, |y|)$. Let $|y| = n$. Clearly, the knowledge base $f_1(\epsilon, |y|)$ depends just on n, i.e., there is one knowledge base for each integer. Call it KB_n. Moreover, also $f_2(\epsilon, |y|) = f_2(\epsilon, n)$ depends just on n: call it O_n (for Oracle). Observe that O_n has polynomial size wrt n.

If there exists a poly-size reduction $f : F_1 \mapsto F_2$ satisfying model preservation, then given the knowledge base KB_n there exists a poly-time function h_n such that $g(O_n, y) \models_{F_1} KB_n$ if and only if $h_n(g(O_n, y)) \models_{F_2} f(KB_n)$.

Therefore, the nu-comp-C_1-complete problem ϵA can be non-uniformly reduced to a problem in nu-comp-C_2 as follows: Given y, from its size $|y| = n$ one obtains (with an arbitrary preprocessing) $f(KB_n)$ and O_n. Then one checks whether the interpretation $h_n(g(O_n, y))$ (computable in poly-time given y and O_n) is a model in F_2 for $f(KB_n)$. From the fact that model checking in F_2 is in nu-comp-C_2, we have that nu-comp-C_1 $\subseteq$ nu-comp-C_2. From [CDLS96, Thm. 9] it follows that C_1/poly $\subseteq$ C_2/poly, and from [Yap83] the polynomial hierarchy collapses. □

The above theorems show that the hierarchy of classes model-C exactly characterizes the space efficiency of a formalism in representing sets of models. In fact, two formalisms at the same level in the model hierarchy can be reduced one into the other via a poly-size reduction (Theorem 3), while there is no poly-size reduction from a formalism (F_1) higher up in the hierarchy into one (F_2) in a lower class (Theorem 4). In the latter case we say that F_1 is *more space-efficient* than F_2.

Analogous results (with similar proofs) hold for poly-size reductions preserving theorems.

Theorem 5 *Let F_1 and F_2 be two PKR formalisms. If F_1 is* thm-C*-complete and F_2 is* thm-C*-hard, then there exists a poly-size reduction $f : F_1 \mapsto F_2$ satisfying theorem preservation.*

Theorem 6 *Let F_1 and F_2 be two PKR formalisms. If the polynomial hierarchy does not collapse, F_1 is* thm-C_1*-hard, F_2 is in* thm-C_2*, and $C_2 \subset C_1$, then there is no poly-size reduction $f : F_1 \mapsto F_2$ satisfying theorem preservation.*

Theorems 3-6 show that compilability classes characterize very precisely the relative capability of PKR formalisms to represent sets of models or sets of theorems. For example, as a consequence of Theorems 2 and 6 there is no poly-size reduction from PL to Horn clauses that preserves the theorems unless the polynomial hierarchy collapses. Kautz and Selman proved non-existence of such a reduction for a problem strictly related to CI in [KS92] using an ad-hoc proof.

6 APPLICATIONS

We now apply the theorems presented in the previous section to several PKR formalisms. We assume that

definitions of propositional logic, default logic [Rei80], circumscription [McC80], and stable model semantics for logic programs [GL88] are known. Definitions of other less known PKR formalisms follow.

WIDTIO and SBR are two belief revision formalisms. Let K be a set of propositional formulae, representing an agent's knowledge about the world. When a new formula A has to be added to K, the problem of the possible inconsistency between K and A arises. $K * A$ denotes the result of such incorporation. We define:

$$W(K, A) = \{K' \mid K' \text{ is a maximal consistent subset of } K \cup \{A\} \text{ containing } A\}$$

Any $K' \in W(K, A)$ is a maximal choice of formulas in K that are consistent with A and, therefore, we may retain when incorporating A.

SBR (Skeptical Belief Revision [FUV83, Gin86]) The revised theory is defined as a set of theories: $K * A \doteq \{K' \mid K' \in W(K, A)\}$. Logical consequence in the revised theory is defined as logical consequence in each of the theories:

$$K * A \models_{SBR} Q \quad \text{iff} \quad \begin{array}{l}\text{for all } K' \in W(K, A) \\ \text{we have that } K' \models Q\end{array}$$

The model semantics is defined as:

$$M \models_{SBR} K * A \quad \text{iff} \quad \begin{array}{l}\text{there exists a } K' \in W(K, A) \\ \text{such that } M \models K'\end{array}$$

WIDTIO (When In Doubt Throw It Out [Win90]) A simpler (but somewhat drastical) approach is the so-called WIDTIO, where we retain only the formulae of K that belong to all (maximally consistent) sets of $W(K, A)$. Thus, consequence is defined as:

$$K * A \models_W Q \quad \text{iff} \quad \bigcap W(K, A) \models Q$$

The model semantics of this formalism is defined as:

$$M \models_W K * A \quad \text{iff} \quad M \models \bigcap W(K, A)$$

We also consider the Generalized Closed World Assumption (GCWA) that is a formalism to represent knowledge in a closed world.

GCWA
(Generalized Closed World Assumption [Min82]) The model semantics is defined as (a is a letter):

$$M \models_G KB \quad \text{iff } M \models KB \cup \{\neg a \mid \text{for any positive clause } \gamma, \text{ if } KB \not\models \gamma \text{ then } KB \not\models \gamma \vee a\}$$

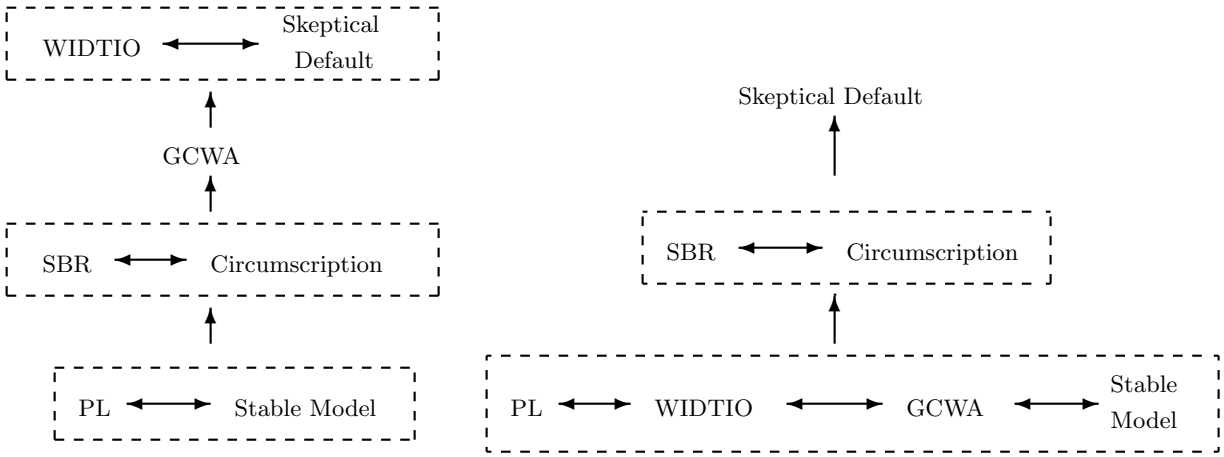
Using Theorems 3-6 and results previously known from the literature, we can obtain some new results on model- and theorem-compactness of PKR formalisms. Old and new results on space efficiency of PKR formalisms are presented in Table 1. Results with no reference are new (a dash "–" denotes a folklore result).

First of all, notice that space efficiency is not always related to time complexity. As an example, we compare in detail WIDTIO and circumscription. From the table it follows that both model checking and theorem proving are harder for WIDTIO than for circumscription. Nevertheless, since circumscription is thm-Σ_2^p-complete and WIDTIO is thm-coNP-complete (and thus in thm-Σ_2^p), there exists a poly-size reduction from WIDTIO to circumscription satisfying theorem preservation. The converse does not hold: since circumscription is thm-Σ_2^p-complete and WIDTIO is thm-coNP, there is no theorem-preserving poly-size reduction from the former formalism to the latter. Hence, circumscription is a more compact formalism than WIDTIO to represent theorems. Analogous considerations can be done for models. Intuitively, this is due to the fact that for WIDTIO both model checking and inference require a lot of work on the revised knowledge base alone—computing the intersection of all elements of $W(K, A)$. Once this is done, one is left with model checking and inference in PL. Hence, WIDTIO has the same space efficiency as PL, which is below circumscription.

Figures 3 and 4 contain the same information of Table 1, but highlight existing reductions. Each figure contains two diagrams, the left one showing the existence of polynomial-time reductions among formalisms, the right one showing the existence of poly-size reductions. An arrow from a formalism to another denotes that the former can be reduced to the latter one. We use a bidirectional arrow to denote arrows in both directions and a dashed box to enclose formalisms that can be reduced one into another. Note that some formalisms are more appropriate in representing sets of models, while others perform better on sets of formulae. An interesting relation exists between Skeptical Default Reasoning and circumscription. While there is no model-preserving poly-size reduction from circumscription to Skeptical Default Reasoning, as already shown in [GKPS95], a theorem-preserving poly-size reduction exists.

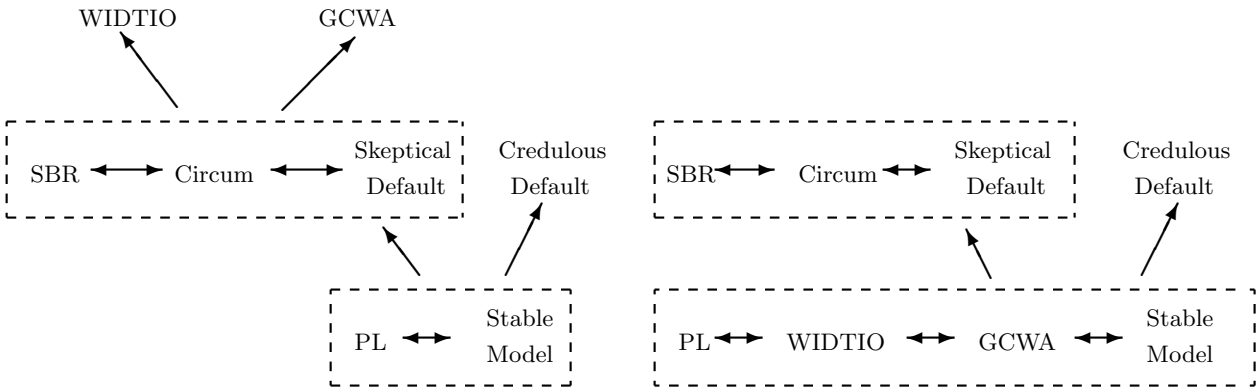
7 CONCLUSIONS

In a recent paper [CDLS96] we introduced a new complexity measure, i.e., *compilability*. In this paper we have shown how this measure is inherently related to the succinctness of PKR formalisms. We analyzed



a. Time Complexity b. Space Efficency

Figure 3: Complexity of Model Checking vs. Space Efficiency of Model Representation



a. Time complexity b. Space efficency

Figure 4: Complexity of Inference vs. Space Efficiency of Theorem Representation

	Time Complexity		Space Efficiency	
	Model	Theorem	Model	Theorem
Propositional Logic	P –	coNP-complete [Coo71]	model-P –	thm-coNP-complete [CDS95]
WIDTIO	Σ_2^p-complete [LS96]	Π_2^p-hard, in $\Delta_3^p[\log n]$ [EG92]	model-P [CDLS95]	thm-coNP-complete [CDLS95]
Skeptical Belief Revision	coNP-complete [LS96]	Π_2^p-complete [EG92]	model-coNP-complete [LS96]	thm-Π_2^p-complete [CDLS95]
circumscription	coNP-complete [Cad92]	Π_2^p-complete [EG93]	model-coNP-complete [CDSS95]	thm-Π_2^p-complete [CDSS95]
GCWA	coNP-hard, in $\Delta_2^p[\log n]$ [EG93]	Π_2^p-hard, in $\Delta_3^p[\log n]$ [EG93]	model-P [CDSS95]	thm-coNP-complete [CDSS95]
Skeptical Default Reasoning	Σ_2^p-complete [Got92]	Π_2^p-complete [Got92]	model-Σ_2^p-complete	thm-Π_2^p-complete
Credulous Default Reasoning	N/A	Σ_2^p-complete [Got92]	N/A	thm-Σ_2^p-complete
Stable Model Semantics	P –	coNP-complete [MT91]	model-P –	thm-coNP-complete

Table 1: Complexity and Space Efficiency of Formalisms

PKR formalisms wrt two succinctness measures: succinctness in representing sets of models and succinctness in representing sets of theorems.

We provided a formal way of talking about the relative ability of PKR formalisms to compactly represent information. In particular, we were able to formalize the intuition that a specific PKR formalism provides "one of the most compact ways to represent models/theorems" among the PKR formalisms of a specific class.

Using this tool we have been able to improve on our previous work on compact representations [CDS95, CDSS95, CDLS95] as well as the work by Gogic, Kautz, Papadimitriou and Selman [GKPS95].

Acknowledgments

This work was partially supported by ASI (Italian Space Agency) and CNR (National Research Council of Italy).

References

[BED91] R. Ben-Eliyahu and R. Dechter. Default logic, propositional logic and constraints. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 379–385, 1991.

[BED94] R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12:53–87, 1994.

[Cad92] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Information Processing Letters*, 44:113–118, 1992.

[CDLS95] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. The size of a revised knowledge base. In *Proceedings of the Fourteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS-95)*, pages 151–162, 1995.

[CDLS96] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Feasibility and unfeasibility of off-line processing. In *Proceedings of the Fourth Israeli Symposium on Theory of Computing and Systems (ISTCS-96)*, pages 100–109, 1996.

[CDS95] M. Cadoli, F. M. Donini, and M. Schaerf. Is intractability of non-monotonic reasoning a real drawback? Technical Report RAP.09.95, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", July 1995. To appear in *Artificial Intelligence Journal*. Short version appeared in *Proc. of AAAI-94*, pages 946–951.

[CDSS95] M. Cadoli, F. M. Donini, M. Schaerf, and R. Silvestri. On compact representations of propositional circumscription. Technical Report RAP.14.95, Dipartimento di Informatica e Sistemistica, Università di Roma "La

Sapienza", July 1995. To appear in *Theoretical Computer Science*. Short version appeared in *Proc. of STACS-95*, pages 205–216.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third ACM Symposium on Theory of Computing (STOC-71)*, pages 151–158, 1971.

[EG92] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence Journal*, 57:227–270, 1992.

[EG93] T. Eiter and G. Gottlob. Propositional circumscription and extended closed world reasoning are Π_2^p-complete. *Theoretical Computer Science*, pages 231–245, 1993.

[Eth87] D. V. Etherington. *Reasoning with incomplete information*. Morgan Kaufmann, Los Altos, Los Altos, CA, 1987.

[FUV83] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *Proceedings of the Second ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS-83)*, pages 352–365, 1983.

[Gin86] M. L. Ginsberg. Conterfactuals. *Artificial Intelligence Journal*, 30:35–79, 1986.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, Ca, 1979.

[GKPS95] G. Gogic, H. Kautz, C. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 862–869, 1995.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth Logic Programming Symposium*, pages 1070–1080. The MIT Press, 1988.

[Got92] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2:397–425, 1992.

[GPP89] M. Gelfond, H. Przymusinska, and T. Przymusinsky. On the relationship between circumscription and negation as failure. *Artificial Intelligence Journal*, 38:49–73, 1989.

[Joh90] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[KL80] R. M. Karp and R. J. Lipton. Some connections between non-uniform and uniform complexity classes. In *Proceedings of the Twelfth ACM Symposium on Theory of Computing (STOC-80)*, pages 302–309, 1980.

[KS92] H. A. Kautz and B. Selman. Forming concepts for fast inference. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 786–793, 1992.

[Lib95] P. Liberatore. Compact representation of revision of Horn clauses. In *Proceedings of the Eighth Australian Joint Artificial Intelligence Conference*, 1995.

[LS96] P. Liberatore and M. Schaerf. The complexity of model checking for belief revision and update. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996. To appear.

[McC80] J. McCarthy. Circumscription - A form of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.

[Min82] J. Minker. On indefinite databases and the closed world assumption. In *Proceedings of the Sixth International Conference on Automated Deduction (CADE-82)*, pages 292–308, 1982.

[MT91] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence Journal*, 13:81–132, 1980.

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[Win89] M. Winslett. Sometimes updates are circumscription. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 859–863, 1989.

[Win90] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.

[Yap83] C. K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.