Compilability and Compact Representations of Revision of Horn Knowledge Bases

Paolo Liberatore
Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica

Several methods have been proposed as an attempt to deal with dynamically-changing scenarios. From a computational point of view, different formalisms have different computational properties. In this paper we consider knowledge bases represented as sets of Horn clauses. The importance of this case is twofold: first, inference is polynomial, thus tractable; second, Horn clauses represent causal relations between facts, thus they are of great practical importance, although not all propositional knowledge bases can be represented in Horn form.

The complexity of Horn revision is still high, and in some cases coincides with the complexity of the general (non-Horn) case. We analyze the complexity of belief revision from the point of view of the compilation [Cadoli et al. 1996]: we study the possibility of reducing the complexity by allowing a (possibly expensive) preprocessing of part of the input of the problem.

Extending the work of Cadoli, Donini, Liberatore, and Schaerf [1999], we consider the problem of compact representation of revision in the Horn case, that is, given a knowledge base T and an update P (both represented by Horn clauses) decide whether $T * P$, the result of the revision, can be represented with a propositional formula whose size is polynomial in the size of T and P. We give this representation for all formalisms for which it exists, and we show that the existence of a compact representation is related to the possibility of decreasing the complexity of a formalism via a preprocessing.

Categories and Subject Descriptors: I.2.3 [**Deduction and Theorem Proving**]: Nonmonotonic reasoning and belief revision; I.2 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; F.1.3 [**Complexity Measures and Classes**]: Reducibility and completeness

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Compact representations, compilability

This paper is an extended and revised version of a paper published by the same author in the *Proceedings of the Eighth Australian Joint Artificial Intelligence Conference (AI'95)* [Liberatore 1995].

Address: Via Salaria 113, 00198 Rome, Italy. Email: liberato@dis.uniroma1.it

1. INTRODUCTION

The importance of the dynamic treatment of information in artificial intelligence, databases and philosophy has led many researchers to focus their attention on its computational aspects. While the first theoretical studies in this area [Gärdenfors 1988] are more concerned with the definition of general properties about how changes should be made, more recent work addresses the problem of finding specific methods to revise knowledge.

We are given a set of propositional formulas T representing the old knowledge we have about the world of interest, and an observation P. This new information must be incorporated in the old knowledge base. When T and P are consistent to each other, the result of this step is simply the union of P and T. The interesting case is when T and P, each consistent by itself, are inconsistent to each other. There are many ways to revise T with P. The main assumption of belief revision is that of minimal change: the old description T should be changed as little as possible in order to incorporate P. This is a formalization of the principle of maintaining as much information as possible in this process of merge. The minimal change principle can be realized in several ways, since there are many possible ways to define when a change is minimal. This has led to the definition of different revision operators.

From a computational point of view, most of the semantics proposed for revision are at the second level of the polynomial hierarchy, namely Π_2^p [Eiter and Gottlob 1992]. The knowledge base T is used as a formalization of the "state of affairs" of the world; the most important problem is to decide which facts derive from this piece of information. The problem of deciding whether a formula Q is implied by T is in general a coNP complete problem, thus intractable. For practical use, we have to restrict our attention to special cases in which inference becomes tractable. The most used restriction is to consider only Horn formulas.

Restricting T, P and Q to be Horn formulas, the complexity of deciding whether $T * P \models Q$ (deciding whether Q is implied by the updated knowledge base) often decreases to coNP or P [Eiter and Gottlob 1992]. This is why revision of Horn formulas is of special interest, although not all knowledge bases can be represented in this form.

The problem we address in this paper is the following: given a set T and a formula P, both Horn, whose size are $||T||$ and $||P||$ respectively, can we represent the revised knowledge base (k.b.) with a propositional formula whose size is polynomial w.r.t. $||T|| + ||P||$? If this is not possible, it can be considered impossible, from a practical point of view (limitation of computer memory), to explicitly represent the revised knowledge base.

This question is related to the problem of compilability of belief revision: is it possible to reduce the complexity of deciding whether $T * P \models Q$ if one *precompiles* the formulas T and P? This question makes sense if the revised knowledge base must be queried many times w.r.t. several queries Q. Indeed, in this case, a long preprocessing of T and P may be convenient, if it allows a fast answer to the question $T * P \models Q$ whenever a query Q is posed.

This paper is organized as follows: in the next section we recall the various definitions of revision given in the literature and the formalization of compilability.

In Section 3 we formally define the concept of compact representation. In Section 4 we show the properties of revision from the point of view of compilation. In Section 5 we show when $T * P$ can be represented in polynomial space. The proofs are in the Appendix for ease of reading.

2. PRELIMINARIES

2.1 Notations

Let $X = \{x_1, \ldots, x_n\}$ be a set of propositional variables. Given a variable x_i, a literal is either x_i or $\neg x_i$. We denote by l_i a literal corresponding to the variable x_i (i.e. l_i is either x_i or $\neg x_i$).

A clause is a disjunction of literals: $C = l_{i_1} \vee \cdots \vee l_{i_m}$. A Horn clause is a clause containing at most one positive literal. A Horn formula is a conjunction of Horn clauses. It is well-known [Dowling and Gallier 1984] that problems such as satisfiability, unsatisfiability and logical consequence for Horn formulas can be solved by linear-time algorithms. This means that deciding whether a Horn formula is satisfiable can be checked by an efficient algorithm (the same for unsatisfiability and inference). A 3cnf formula is a conjunction of clauses, each composed of at most three literals. A theory is a set of propositional formulas.

An interpretation is a truth assignment to the variables, i.e. a function from X to $\{\mathsf{true}, \mathsf{false}\}$. We extend this mapping to propositional formulas in the usual way. We denote an interpretation by the set of variables mapped into true. An interpretation M is a model of a propositional formula P (denoted by $M \models P$) if it maps P into true. We use $\mathit{Mod}(P)$ to denote the set of models of P i.e. $\mathit{Mod}(P) = \{M \mid M \models P\}$. An interpretation is a model of a theory T if it is a model of each formula in the theory. We overload the function Mod to denote also the set of models of a theory, i.e., $\mathit{Mod}(T)$ is the set of models of the theory T. Given a set of models A, the function $\mathit{Form}(A)$ gives a formula whose set of models is A. This is a multi-valued function, but this is inessential for the purpose of this paper, i.e., we may assume that $\mathit{Form}(A)$ has some specific form, for example disjunction of maximal terms. We denote by $\mathit{Var}(P)$ the set of variables occurring in P.

Given two interpretations I and J, we denote by $\mathit{Diff}(I, J)$ the symmetric difference between I and J, that is, the set $I \backslash J \cup J \backslash I$. Intuitively, $\mathit{Diff}(I, J)$ is the set of variables that I and J map into different truth values.

A propositional formula can be seen as a way to represent a set of models. A more general representation is that of circuits. We use a simplified definition of circuit, that does not take into account things such as levels, limitations in the number of inputs of gates, etc. A detailed survey by Boppana and Sipser [1990] contains more details. A circuit over a set of variables X is a boolean formula F built over an alphabet composed of X and a new, ordered, set of variables $Y = \{y_1, \ldots, y_m\}$, as follows

$$F = (y_1 \equiv F_1) \wedge (y_2 \equiv F_2) \wedge \cdots \wedge (y_{m-1} \equiv F_{m-1}) \wedge F_m$$

where each F_i is a formula (not a circuit) containing only the variables $X \cup \{y_1, \ldots, y_{i-1}\}$. Let $M \subseteq X$ be an interpretation over X. Note that F can be viewed both as a circuit over X and as a (usual) formula over $X \cup Y$. We define

M to be a model of the *circuit* F (denoted by $M \models F$) if there exists $N \subseteq Y$ such that $M \cup N$ is a model of the *formula* F. We denote the set of models of a circuit F as $Mod(F)$, when there is no possibility of confusion of F being a circuit or a formula. It is easy to see that given an interpretation M and a circuit F, deciding whether $M \models F$ can be done in linear time.

A substitution is a set of pairs, each composed of a variable and a formula. If $S = \{x_i/Q_i\}$ is a substitution and P a formula, then $P[S]$ is the formula obtained by replacing each occurrence of x_i with Q_i in P. A similar definition holds for circuits and theories. Given a set X, we denote by $|X|$ the number of its elements.

Given an alphabet Σ, a string s over Σ is an ordered set of elements of Σ. We denote by $||s||$ the *length* of s, that is, the number of elements in s. Each propositional formula can be represented as a string. We always assume a reasonable encoding for the representation of formulas [Johnson 1990]. Given a propositional formula P we use $||P||$ to denote the size of its representing string. Similarly, $||F||$ and $||T||$, where F and T are a circuit and a theory, respectively, denote the length of the strings representing them.

2.2 Revision

Consider a theory T as a piece of information that we consider true until time t_1. Suppose that we know (we observe) a fact represented by the formula P, at time t_2. What should be our knowledge after t_2? If T and P are mutually consistent (that is, $T \cup \{P\}$ is satisfiable), we can simply take $T \cup \{P\}$ as our new knowledge base, that is, we can put together the old k.b. and the new observation. However, $T \cup \{P\}$ may be inconsistent. In such cases, we have to specify how T and P concur to create the new knowledge base.

Till now, we have regarded the words "revision" and "update" as synonymous. However, they correspond to different hypothesis to explain the inconsistency between T and P.

(1) The old formula T was true until t_1, but between t_1 and t_2 the world of interest has been changed in such a way P is now true.

(2) Nothing has changed, and the world of interest is exactly the same through the time. We assume that P is true, but T is (totally or partially) wrong. This means that the k.b. after t_2 must imply P, but it may not imply T.

Briefly, revision formalizes the treatment of a static world (the second assumption), while update deals with changing scenarios (like the first one). In the sequel we do not pay attention to the difference. Actually, most update and revision operators are very similar: the main assumption made in their definitions is that of minimal change: the formula $T * P$ (the result of a revision or update) should differ from T as little as possible. Definitions given differ on the intended meaning of "less different".

We briefly recall revision and update operators proposed in the last years. A more detailed survey is contained in Katsuno and Mendelzon's paper [Katsuno and Mendelzon 1991].

2.2.1 *Syntax-Based Revision Operators.* Let $W(T, P)$ be the set of the maximal subsets of T consistent with P, that is:

$$W(T,P) = \max_{\subseteq}(\{T' \subseteq T \mid T \cup \{P\} \not\models \bot\})$$

Full meet revision. This is a simple revision satisfying the AGM postulates [Alchourrón et al. 1985]. It is defined as:

$$T *_{FM} P = \begin{cases} \wedge(T \cup \{P\}) & \text{if consistent} \\ P & \text{otherwise} \end{cases}$$

GVUF. GFUV (Ginsberg and Fagin, Ullman, and Vardi) revision [Ginsberg 1986; Fagin et al. 1983] is defined as the skeptical union of the elements of $W(T,P)$:

$$T *_{GFUV} P = \vee W(T,P) \wedge P$$

WIDTIO. (When In Doubt, Throw It Out) is very similar to GFUV:

$$T *_{Wit} P = \wedge(\cap W(T,P) \cup \{P\})$$

Example 1. Let for instance $T = \{a, \neg b, \neg c\}$, and $P = b \vee c$. In this case, T and P contradict each other, so we have to compute the maximal subsets of T that are consistent with P. Each subset of T that does not contain both $\neg b$ and $\neg c$ is consistent with P:

$$W(T,P) = \{\{a, \neg b\}, \{a, \neg c\}\}$$

By definition, GFUV is the disjunction of these subsets, while WIDTIO is obtained by taking their intersection. Finally, FM is obtained by taking P only.

$$\begin{aligned} T *_{GFUV} P &= [(a \wedge \neg b) \vee (a \wedge \neg c)] \wedge (b \vee c) \quad = \quad a \wedge (b \not\equiv c) \\ T *_{FM} P &= b \vee c \\ T *_{Wit} P &= a \wedge (b \vee c) \end{aligned}$$

2.2.2 *Model-Based Revision Operators.* The definition of the revision operators by Dalal [Dalal 1988], Forbus [Forbus 1989], Borgida [Borgida 1985], and Satoh [Satoh 1988] are expressed in terms of the set of models of $T * P$.

Dalal. The models of $T *_D P$ are the models J of P such that there exists a model I of T such that $|Diff(I,J)|$ is minimal. Formally:

$$Mod(T *_D P) = \left\{ J \in Mod(P) \;\middle|\; \begin{array}{l} \exists I \in Mod(T) \text{ such that} \\ \nexists I' \in Mod(T)\ \nexists J' \in Mod(P) \\ \text{with } |Diff(I',J')| < |Diff(I,J)| \end{array} \right\}$$

Forbus. This revision is similar to Dalal's, but T is revised on a model-by-model base.

$$T *_F P = \bigvee_{I \in Mod(T)} Form(I) *_D P$$

Satoh. The models of P that are selected to be models of the revised base are those with a minimal distance to models of T, where the distance is defined as the symmetric difference.

$$Mod(T *_S P) = \left\{ J \in Mod(P) \;\middle|\; \begin{array}{l} \exists I \in Mod(T) \text{ such that} \\ \nexists I' \in Mod(T)\ \nexists J' \in Mod(P) \\ \text{with } Diff(I', J') \subset Diff(I, J) \end{array} \right\}$$

Winslett. This is the model-by-model version of Satoh's revision:

$$T *_W P = \bigvee_{I \in Mod(T)} Form(I) *_S P$$

Borgida. It coincides with Winslett's operator, except in the case when $T \cup \{P\}$ is consistent.

$$T *_B P = \begin{cases} \wedge(T \cup \{P\}) & \text{if consistent} \\ T *_W P & \text{otherwise} \end{cases}$$

Example 2. Let T and P be as follows.

$$\begin{aligned} T &= \{\neg b \wedge \neg c \wedge \neg d\} \\ P &= (a \wedge b \wedge \neg c \wedge \neg d) \vee (c \wedge d \wedge (a \not\equiv e)) \end{aligned}$$

The models of T are $\emptyset$ and $\{a\}$, while P has three models, namely $\{a, b\}$, $\{a, c, d\}$ and $\{c, d, e\}$. The symmetric differences between models of T and models of P are as follows.

	$\{a, b\}$	$\{a, c, d\}$	$\{c, d, e\}$
$\emptyset$	$\{a, b\}$	$\{a, c, d\}$	$\{c, d, e\}$
$\{a\}$	$\{b\}$	$\{c, d\}$	$\{a, c, d, e\}$

The models $\{a\}$ and $\{a, b\}$ differ for one variable only (a), while all other differences are composed of at most two variables. This means that the only model selected by Dalal's revision is the model of P corresponding to this difference, that is, $\{a, b\}$. The result of Satoh's revision is obtained by using the set containment relation as a measure of distance. As a result, $\{a, c, d\}$ is in the result, since the difference $\{b\}$ is not strictly contained in the difference $\{c, d\}$.

In order to compute Forbus' revision, we have to consider the models that are the closest to $\emptyset$ first, and then the closest to $\{a\}$. The model that is closest to $\emptyset$, according to the number of variables on which models differ, is $\{a, b\}$, and this is also the only model that is the closest to $\{a\}$. The result is thus composed of the model $\{a, b\}$ only.

Since $T \wedge P$ is inconsistent, Winslett's and Borgida's revisions coincide. Namely, they are obtained by taking the models of P that are the closest to $\emptyset$ and $\{a\}$, using the set containment relation. It is easy to see that the differences between $\emptyset$ and the three models of P are incomparable w.r.t. this measure. As a result, $T *_W P = P$.

2.2.3 *Revisions with Unreliable Set.* When a new piece of information P leads us to consider T wrong, we may suppose that T is wrong because of an error during the "observation" of a certain set of variables $\Omega \subseteq X$. Such an assumption can be formalized as follows.

$$Mod(T *_\Omega P) = \{J \in Mod(P) \mid \exists I \in Mod(T)\ .\ I \backslash \Omega = J \backslash \Omega\}$$

which means that the value imposed by T to the variables Ω must be ignored.
Standard Semantic Update. Ω is the set of the variables of P. Note that the result depends upon the syntactical form of P: for example $P_1 = x_1$ and $P_2 = x_1 \wedge (x_2 \vee \neg x_2)$ are equivalent, but $\{x_2\} *_{SSU} P_1 = x_1 \wedge x_2$ while $\{x_2\} *_{SSU} P_2 = x_1$.
Weber's update. The set Ω is defined as follows.

$$\Omega = \cup \left\{ Diff(I,J) \;\middle|\; \begin{array}{l} I \in Mod(T),\ J \in Mod(P) \text{ and} \\ \nexists I' \in Mod(T), \quad \nexists J' \in Mod(P) \\ \text{such that } Diff(I',J') \subset Diff(I,J) \end{array} \right\}$$

In this case, the set Ω is obtained as the union of all the minimal differences (w.r.t. set-containment) between models of T and models of P.

Example 3. Let $T = \{a, \neg b \wedge \neg c \wedge \neg d\}$ and $P = b \wedge (c \equiv d)$. The knowledge base has only one model: $\{a\}$. The update has exactly four models: $\{b\}$, $\{a, b\}$, $\{b, c, d\}$ and $\{a, b, c, d\}$. The minimal difference between models of T and models of P is $\{b\}$. As a result, $\Omega = \{b\}$. The result of Weber's revision is in this case $T *_{Web} P = a \wedge b \wedge \neg c \wedge \neg d$.

Let us now consider the standard semantics. The variables of P are $\{b, c, d\}$, thus $T *_{SSU} P = a \wedge b \wedge (c \equiv d)$.

Note that, for most of the revision operators introduced, the formula $T * P$ may be a non-Horn formula even if both T and P are Horn. Namely, if $*$ is any revision presented above but WIDTIO, full meet revision and revisions with a set of unreliable variables, it holds:

$$\{x_1, x_2\} * (\neg x_1 \vee \neg x_2) = (x_1 \not\equiv x_2)$$

and $x_1 \not\equiv x_2$ cannot be expressed as a Horn formula.

2.3 Computational Complexity

We assume that the reader is familiar with the basic concepts of computational complexity. We use the standard notation of complexity classes that can be found in textbooks of computational complexity [Johnson 1990]. Namely, the class P denotes the set of problems whose solution can be found in polynomial time by a *deterministic* Turing machine, while NP denotes the class of problems that can be resolved in polynomial time by a *non-deterministic* Turing machine. The class coNP denotes the set of decision problems whose complement is in NP. A problem G is NP-hard if and only if any other problem in NP can be reduced to G by means of a polynomial-time (many-one) reduction.

Clearly, P $\subseteq$ NP and P $\subseteq$ coNP. We assume, in line with the prevailing assumption of computational complexity, that these containments are strict, that is P $\neq$ NP and P $\neq$ coNP. Therefore, we call a problem that is in P *tractable*, and a problem that is NP-hard or coNP-hard *intractable*.

We also use higher complexity classes defined using oracles. In particular P^A (NP^A) is the class of decision problems that are solved in polynomial time by deterministic (nondeterministic) Turing machines using an oracle for A in polynomial time. The classes Σ_k^p, Π_k^p and Δ_k^p of the polynomial hierarchy are defined as:

$$\Sigma_0^p = \Pi_0^p = \Delta_0^p = \mathrm{P}$$

and for $k \geq 0$,

$$\Sigma_{k+1}^p = \mathrm{NP}^{\Sigma_k^p}, \quad \Pi_{k+1}^p = \mathrm{co}\Sigma_{k+1}^p \ , \quad \Delta_{k+1}^p = \mathrm{P}^{\Sigma_k^p}$$

Notice that $\Delta_1^p = \mathrm{P}$, $\Sigma_1^p = \mathrm{NP}$ and $\Pi_1^p = \mathrm{coNP}$. The polynomial hierarchy PH is defined as $\mathrm{PH} = \bigcup_{i \geq 0} \Sigma_i^p$.

2.4 Compilability Classes

In this section we summarize the definitions and results about compilability that are relevant to our work [Cadoli et al. 1996]. Compilability is a new form of measure of complexity, motivated by the fact that, in many cases, the input of a computational problem can be divided into two parts: one, called *fixed*, is known in advance, while the other one, called *variable*, comes at the same time as the request of the computation. Such a distinction is interesting when many input instances share the same fixed part. In these cases it makes sense to preprocess *off-line* the fixed part, putting it into a form such that solving *on-line* a set of instances becomes easier.

We assume that languages are built over an alphabet Σ. The *length* of a string $x \in \Sigma^*$ is denoted by $||x||$. In this paper we are concerned with problems composed of pairs of inputs, where one part is fixed (accessible off-line) and the second one is variable (only accessible on-line). Therefore, we define a *language of pairs* S as a subset of $\Sigma^* \times \Sigma^*$. A function f is called *poly-size* if there exists a polynomial p such that for all x it holds $||f(x)|| \leq p(||x||)$.

Our intuitive notion of compilability states that a problem S can be reduced into a (hopefully simpler) problem by modifying only its fixed part. Given a complexity class C, we introduce the class of problems compilable to C, that we denote as $\rightsquigarrow$C. The complement of this class is denoted as co-($\rightsquigarrow$C). Throughout this paper we assume that C denotes a complexity class of the polynomial hierarchy.

DEFINITION 1. *A language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ belongs to $\rightsquigarrow$C iff there exists a poly-size function f and a language of pairs S' such that for all $\langle x, y \rangle \in S$ we have that:*

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

Notice that no restriction is imposed on the *time* needed to compute the function f, but only on the *size* of the result, i.e. f is a poly-size function. This definition can be represented in terms of a computing machine as in Figure 1.

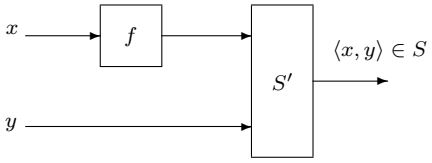

Fig. 1. Machine for the $\rightsquigarrow$C class

This machinery captures our intuitive notion of compilability into C of a problem S with fixed and variable parts. We process off-line the fixed part x, thus obtaining

$f(x)$, and then we decide whether $\langle f(x), y\rangle \in S'$. The whole process is convenient if deciding $\langle f(x), y\rangle \in S'$ is easier than deciding $\langle x, y\rangle \in S$.

In particular, the class $\rightsquigarrow$P contains all the problems that can be solved in polynomial time on-line after an off-line processing of the fixed part.

It is not possible to define completeness for $\rightsquigarrow$C using polynomial-time reductions. In order to define a reduction that preserve the compilability property and are powerful enough to allow the definition of complete problems, we introduce the notion of *comp-reduction*.

DEFINITION 2. *Given two problems A and B, we say that A is* comp-reducible *to B (denoted as $A \rightsquigarrow B$) iff there exist two unary poly-size functions f_1, 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 $\langle f_1(x), g(f_2(x), y)\rangle \in B$.*

Intuitively, $A \rightsquigarrow B$ if 1) the fixed part of B can be obtained from the fixed part of A using a poly-size function (f_1), and 2) the variable part of B can be constructed using both a poly-size function (f_2) applied to the fixed part of A, and a polynomial-time function (g) applied to the variable part of A. The $\rightsquigarrow$ reductions can be represented as in Figure 2.

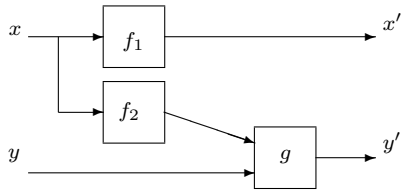

Fig. 2. $\rightsquigarrow$ reductions

These reductions satisfy all basic properties of a reduction. Indeed, we have that:

THEOREM 1. *The comp-reductions $\rightsquigarrow$ are transitive and compatible [Johnson 1990] with the class $\rightsquigarrow$C.*

Therefore, it is possible to define a notion of *completeness* for $\rightsquigarrow$C.

DEFINITION 3. *Let B be a language of pairs. B is* $\rightsquigarrow$C complete *iff B is in $\rightsquigarrow$C and for all problems $A \in \rightsquigarrow$C we have that $A \rightsquigarrow B$.*

Using the above definition we can find complete problems. Given a problem S with one input, we call $*S$ the problem with two inputs defined as follows.

$$*S = \Sigma^* \times S = \{\langle x, y\rangle \mid x \text{ is any string and } y \in S\}$$

THEOREM 2. *For every complexity class* C*, if S is* C *complete (under polynomial many-one reduction) then $*S$ is complete (under $\rightsquigarrow$ reduction) for the corresponding compilability class $\rightsquigarrow$C.*

As an example, starting from the NP-complete problem 3sat, we obtain the $\rightsquigarrow$NP-complete problem $*$3sat. For several important problems we can prove that they probably do not belong to $\rightsquigarrow$P by proving their $\rightsquigarrow$C-completeness for some C above

P. For example, it can be easily shown that the problem FORMULA INFERENCE (fi) defined as

$$\text{FI} = \{ \langle x, y \rangle \mid x \text{ and } y \text{ are propositional formulae and } x \models y \}$$

is $\rightsquigarrow$coNP complete, with a $\rightsquigarrow$ reduction from $*$3unsat. Nevertheless, for many natural problems the non-compilability proofs appeared in the literature [Kautz and Selman 1992; Cadoli et al. 1996; Cadoli et al. 1997; Cadoli et al. 1999; Gogic et al. 1995; Liberatore 1995] cannot be rephrased as proofs of $\rightsquigarrow$C-completeness.

Take, for instance, the problem 3CNF CLAUSE INFERENCE (ci) defined as follows.

$$\text{ci} = \{ \langle x, y \rangle \mid x \text{ is a 3cnf propositional formula, } y \text{ is a clause and } x \models y \}$$

It has been proven [Kautz and Selman 1992; Cadoli et al. 1996] that the problem belongs to $\rightsquigarrow$P iff NP is included in P/poly. However, this problem belongs to $\rightsquigarrow$coNP (just take f as the identity function) but it is not $\rightsquigarrow$coNP-complete.

THEOREM 3. *If* ci *is* $\rightsquigarrow$coNP *complete then* P=coNP.

In order to overcome this shortcoming we introduce the class of problems *non-uniformly compilable* into a class C, denoted as $\|\!\rightsquigarrow$C. This class generalizes $\rightsquigarrow$P and is mainly used to prove non-compilability results.

DEFINITION 4. *A language of pairs* $S \subseteq \Sigma^* \times \Sigma^*$ *belongs to* $\|\!\rightsquigarrow$C *iff there exists a binary poly-size function* f *and a language of pairs* S' *such that for all* $\langle x, y \rangle \in S$ *we have that:*

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

Notice that now the poly-size function f takes as input both x and the size of y. The corresponding computing machine machine can be formulated in terms of a computing machine as follows:

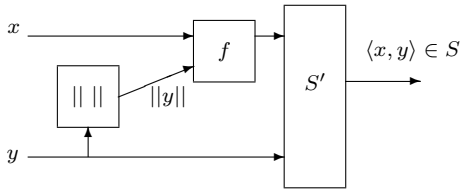

Fig. 3. The $\|\!\rightsquigarrow$C machine

For this class we introduce a distinct reduction that is more general than $\rightsquigarrow$.

DEFINITION 5. *Given two problems* A *and* B*, we say that* A *is* non-uniformly comp-reducible *to* B *(denoted as* $A\|\!\rightsquigarrow B$*) iff there exist two binary poly-size 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* $\langle f_1(x, ||y||), g(f_2(x, ||y||), y) \rangle \in B$.

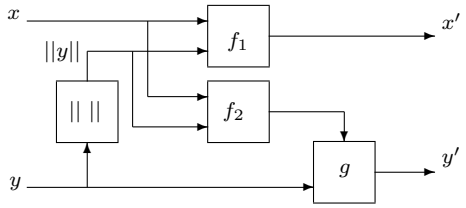

Fig. 4. $\|\rightsquigarrow$ reductions

The $\|\rightsquigarrow$ reductions can be represented as in Figure 4.

These reductions satisfy all important properties of a reduction. Indeed, we have that:

THEOREM 4. *The reductions $\|\rightsquigarrow$ satisfy transitivity and are compatible [Johnson 1990] with the class $\|\rightsquigarrow$C.*

Therefore, it is possible to define a notion of *completeness* for $\|\rightsquigarrow$C.

DEFINITION 6. *Let S be a language of pairs. S is $\|\rightsquigarrow$C complete iff S is in $\|\rightsquigarrow$C and for all problems $A \in \|\rightsquigarrow$C we have that $A\|\rightsquigarrow S$.*

The following theorem shows that $\rightsquigarrow$C and $\|\rightsquigarrow$C share some complete problems.

THEOREM 5. *For every class* C*, if S is* C *complete (under polynomial many-one reduction) then $*S$ is complete (under $\|\rightsquigarrow$ reduction) for the corresponding non-uniform compilability class $\|\rightsquigarrow$C.*

Nevertheless, complete problems for the two classes do not coincide. Indeed, we now have the right complexity class to completely characterize the problem ci:

THEOREM 6. ci *is $\|\rightsquigarrow$*coNP *complete.*

The first result proving unfeasibility of compilation for a problem strictly related to ci has been discovered by Kautz and Selman [1992]. The $\|\rightsquigarrow$ classes are useful to prove incompilability. Indeed, if a $\|\rightsquigarrow\Sigma^p_{i+1}$-hard problem were in $\rightsquigarrow\Sigma^p_i$, then $\Sigma^p_{i+1}=\Sigma^p_{i+2}$. The same for $\rightsquigarrow\Pi^p_i$. For example, if a $\|\rightsquigarrow$NP-hard problem were in P, then $\Sigma^p_2=\Sigma^p_3$.

We close the section by giving a rationale for the classes we defined. $\rightsquigarrow$P captures the idea of "compilable problem". In general, a problem is in $\rightsquigarrow$C if, after adequate preprocessing of its fixed part, solving it on-line is a problem in C. $\|\rightsquigarrow$NP-complete problems (under $\|\rightsquigarrow$ reductions) are what we call "non-compilable", as 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 $\|\rightsquigarrow$coNP-complete problems. In general, a problem which is $\|\rightsquigarrow$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. As for $\rightsquigarrow$NP- and $\rightsquigarrow$coNP-complete problems (under $\rightsquigarrow$ reductions), they are also suggestive of "non-compilability", but we saw in Theorem 3 that the notion of $\rightsquigarrow$C-completeness is not powerful enough to capture formally non-compilability of problems such as ci.

3. COMPACT REPRESENTATIONS

The problem we address in this paper is the following: given a Horn set T and a Horn formula P, is it possible to express $T * P$ with a formula whose size is polynomial in the size of the inputs T and P?

More precisely, we want to determine if there is a formula T_1, equivalent to $T*P$, such that the size of T_1 is polynomial w.r.t. $||T|| + ||P||$. The existence of such T_1 does not follow – in general – from definition. Consider for example T and P defined as follows.

$$\begin{aligned} T &= \{x_1, \ldots, x_n, y_1, \ldots y_n\} \\ P &= (\neg x_1 \vee \neg y_1) \wedge \cdots \wedge (\neg x_n \vee \neg y_n) \end{aligned}$$

The set $W(T, P)$ is exponential, since

$$\begin{aligned} W(T,P) = \{&\{x_1, x_2, \ldots, x_{n-1}, x_n\}, \\ &\{x_1, x_2, \ldots, x_{n-1}, y_n\}, \\ &\{x_1, x_2, \ldots, y_{n-1}, x_n\}, \\ &\{x_1, x_2, \ldots, y_{n-1}, y_n\}, \\ &\vdots \\ &\{y_1, y_2, \ldots, y_{n-1}, y_n\}\} \end{aligned}$$

As a result, $\vee W(T, P)$ is exponential. One may be tempted to conclude that revising T with P (using $*_{GFUV}$) leads to an exponential blow up. However, $\vee W(T, P)$ is equivalent to $\bigwedge\{x_i \not\equiv y_i\}$, which is much smaller. This is our first point in the definition of "compact representation": the revised k.b. can be represented in many ways (i.e. all propositional formulas equivalent to the one given by the definition of revision). Thus, for T and P above, we say that the result of revising T with P can be represented in polynomial space.

For the model based revision operators, the idea is similar: all of them are indeed defined in terms of sets of models, i.e. the definition is $Mod(T*P) = set\_of\_models$. As a result, there is no unique way to define the propositional formula $T * P$. Since we are looking for compact representations, we say that $*$ has a compact representation if and only if there is a formula T_1, such that $Mod(T_1) = Mod(T*P)$, and T_1 has size polynomial in the size of T and P.

Summarizing, we are not looking for the size of $T*P$ as follows from the definition, but for the size of the shortest formula equivalent to $T * P$.

The second point regards the definition of equivalence. In the last paragraphs, we assumed the usual definition of equivalence.

DEFINITION 7. *Two formulas (or circuits, or theories) T_1 and T_2 are logically equivalent*

$$T_1 \equiv T_2 \qquad \textit{iff} \qquad Mod(T_1) = Mod(T_2)$$

This is a strong definition, in the sense that there are cases in which we would like to define equivalent two knowledge bases that do not satisfy this definition.

Consider the following alternative:

$$T_1 \equiv^Q T_2 \text{ iff for any formula } R, \text{ it holds } T_1 \models R \text{ iff } T_2 \models R$$

This second definition highlights the fact that knowledge bases such T_1 and T_2 are used for representing knowledge. In this case, the knowledge represented by T_1 is the set of its implied formulas, thus T_2 can be considered equivalent if it implies the same formulas.

These two forms of equivalence are similar. Indeed, if one allows R to be any formula, they coincide. Assume instead that R is limited to be a formula on the alphabet X, and allow T_1 and T_2 to be built on a bigger alphabet:

DEFINITION 8. *Two formulas (or circuits, or theories) T_1 and T_2 are query equivalent over an alphabet X*

$$T_1 \equiv_X^Q T_2 \textit{ iff for any formula } R \textit{ with } Var(R) \subseteq X, \textit{ it holds } T_1 \models R \textit{ iff } T_2 \models R$$

Where not confusing, we omit the letter X in the symbol $\equiv_X^Q$. Let us analyze this definition in details. First, this definition is different from the previous ones. Assume $X = \{x_1, x_2\}$, and consider $T_1 = x_1 \wedge y_1$, $T_2 = x_1 \wedge \neg y_2$. Clearly, $T_1 \not\equiv T_2$: they are not logically equivalent. However, for any R such that $Var(R) \subseteq X$, we have $T_1 \models R$ if and only if $T_2 \models R$. As a result, $T_1 \equiv_X^Q T_2$: these formulas are query equivalent over X.

The concept of query equivalence is relevant for the study of compact representations. The propositional formulas involved in a revision (T and P) are usually built over a fixed alphabet X. These formulas are used to represent knowledge: we query the k.b. $T * P$ by checking which facts R are true in $T * P$, that is, verifying whether $T * P \models R$. The query R is usually built over the same alphabet X. When we are looking for a formula equivalent to $T * P$, if we are only interested in preserving queries over X, we can consider the query equivalence instead of the logical equivalence.

This can be useful, since formulas that are query equivalent to $T * P$ can be smaller than the logical equivalent ones. For example, let T_1 be the formula as defined as follows.

$$T_1 = [(x_1 \vee x_2 \vee x_3 \vee x_4) \vee (x_5 \wedge \neg(x_1 \vee x_2 \vee x_3 \vee x_4))] \wedge (x_6 \vee \neg(x_1 \vee x_2 \vee x_3 \vee x_4))$$

The size of T_1 can be reduced observing that the subformula $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is repeated three times in the formula. Thus, we can define a *macro* for it:

$$T_2 = [e \equiv (x_1 \vee x_2 \vee x_3 \vee x_4)] \wedge [e \vee (x_5 \wedge \neg e)] \wedge (x_6 \vee \neg e)$$

This new formula is smaller than the previous one. Nevertheless, $T_1 \equiv_X^Q T_2$. Note that these formulas are not logically equivalent, since T_1 has models in which e is false but $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is true, while T_2 does not.

There is a big gain in using query equivalence instead of logical equivalence: sometimes the use of macros can exponentially reduce the size of compact representations.

Let us summarize the above argumentation:

(1) we are trying to determine the size of formulas that are equivalent to $T * P$, rather than the size of $T * P$ itself;

(2) there are two possible ways to define equivalence: the logical (classical) equivalence, and the query equivalence (identity of the sets of implied formulas). The second one allows the use of macros, thus allows for smaller representations.

We define compact representation for a belief revision operator $*$ as a formula which is equivalent to $T * P$ and such that its size is polynomial in $||T|| + ||P||$.

DEFINITION 9. *A compact representation w.r.t. logical equivalence is a formula (circuit, theory) which is logically equivalent to $T * P$ and has size polynomial in $||T|| + ||P||$.*

DEFINITION 10. *A compact representation w.r.t. query equivalence is a formula (circuit, theory) which is query equivalent to $T * P$ and has size polynomial in $||T|| + ||P||$.*

3.1 Model Equivalence

In this section we introduce a third kind of equivalence [Cadoli et al. 1997]. This is motivated by the fact that knowledge is sometimes better represented by sets of models rather that sets of formulas [Halpern and Vardi 1991].

Two formulas are (query) equivalent when they represent the same sets of formulas, that is, when the set of implied formulas are the same. Define two formulas (or circuits, or theories) T_1 and T_2 to be model equivalent if and only if, for each model $I \in Mod(T_1)$ it holds $I \in Mod(T_2)$, and vice versa.

This definition coincides with the logical equivalence one. However, T_1 and T_2 may be built over different sets of variables, but we are only interested in the value of models over a fixed set of variables X.

DEFINITION 11. *We say that T_1 is model equivalent to T_2 over X (written $T_1 \equiv_X^M T_2$) if and only if, for each model $I \in Mod(T_1)$ one can find in polynomial time a model $J \in Mod(T_2)$ such that $I \cap X = J \cap X$, and vice versa.*

Where not confusing, we write simply $T_1 \equiv^M T_2$. With this definition in mind, we define the compact representation of a belief revision operator w.r.t. model equivalence as follows.

DEFINITION 12. *A compact representation w.r.t. model equivalence for a belief revision operator $*$ is a formula (theory, circuit) that is model equivalent to $T * P$ and has size polynomial in the size of T and P.*

3.2 Compilability and Compact Representation

In this section we show the links between compilability of problems in belief revision and the existence of compact representations for them.

The most important reasoning problems in belief revision are:

Query Answering (QA):. Given T, P and Q, decide whether $T * P \models Q$.

Model Checking (MC):. Given T, P and M, decide whether $M \models T * P$.

where T is a set of propositional formulas, P and Q are formulas, and M is a model (all of them are on the alphabet X). Usually, the revised k.b. $T * P$ must be queried many times w.r.t. several queries Q or models M. Thus, it makes sense to compile $T * P$ once (even if this phase is long), if the result of this compilation allows

for a more efficient solving of MC and QA. Thus, it makes sense to determine the compilability classes to which QA and MC belong, for all revision operators introduced. These problem can be formulated as problems over strings as follows.

$$\begin{aligned} \text{QA} &= \{\langle\langle T, P\rangle, Q\rangle \mid T * P \models Q\} \\ \text{MC} &= \{\langle\langle T, P\rangle, M\rangle \mid M \models T * P\} \end{aligned}$$

since T and P are the fixed part of the problem, they are the first element of the pair.

Consider first the logical equivalence. We can prove the following theorem.

THEOREM 7. *If $*$ has a compact representation w.r.t. logical equivalence, then its MC is in $\rightsquigarrow$*P *and its QA is in $\rightsquigarrow$*coNP.

This result is useful for proving that an operator does *not* have a compact representation: if the model checking for a revision operator is not in $\rightsquigarrow$P, then there is no compact representation for it w.r.t. logical equivalence. For example, the problem of model checking for Satoh's revision is $\|\!\rightsquigarrow$NP-hard, thus is not in $\rightsquigarrow$P, thus there is no compact representation w.r.t. logical equivalence.

In the case of model and query equivalence we have the following two theorems.

THEOREM 8. *If $*$ has a compact representation w.r.t. model equivalence, then its MC is in $\rightsquigarrow$*P *and its QA is in $\rightsquigarrow$*coNP.

THEOREM 9. *If $*$ has a compact representation w.r.t. query equivalence, then its MC is in $\rightsquigarrow$*NP*, and its QA is in $\rightsquigarrow$*coNP.

As for the first theorem, these ones are useful for proving the non-existence of compact representations.

4. COMPILABILITY OF REVISION

From this section on, we describe the new results of this paper. We analyze the belief revision operators introduced in terms of compilability. As stated in the previous section, the problems considered here are the query answering (QA: given T, P, and Q, decide whether $T * P \models Q$) and the model checking (MC: given T, P, and M, decide whether $M \models T * P$),

The original results of this paper, for what regards compilability, are summarized in Table 1 and Table 2. The proofs of all the new results are reported in Appendix A, except for Dalal's model checking, that are left in the text to show an example of how the results are proved.

Let us give some considerations about these results. Some of them are quite obvious, for instance those about GFUV and WIDTIO. Indeed, as proved by Liberatore and Schaerf [2000], model checking is polynomial for GFUV, thus it is also compilable to P. The result of compilability of WIDTIO operator is also straightforward: we can compute off-line the set $\cap W(T, P) \cup \{P\}$, which has polynomial size being a subset of $T \cup \{P\}$, and we check whether the model M is a model of all formulas in this set. In this case, a compilability result has been inferred from the existence of a compact representation. However, in most cases we proceed in the other way around, that is, we prove the non-compilability, and conclude the non-existence of a compact representation.

	Compilability of Model Checking		
GFUV	in $\rightsquigarrow$P		(Theorem 13)
Dalal	in $\rightsquigarrow$NP	$\not\rightsquigarrow$NP-hard	(Theorem 10)
Forbus	in $\rightsquigarrow\Sigma_2^p$	$\not\rightsquigarrow$coNP-hard and $\not\rightsquigarrow$NP-hard	(Theorem 14)
Satoh	in $\rightsquigarrow$NP	$\not\rightsquigarrow$NP-hard	(Theorem 15)
Borgida/Winslett	in $\rightsquigarrow$NP	$\not\rightsquigarrow$NP-hard	(Theorem 16)
SSU	in $\rightsquigarrow$P		(Theorem 17)
Weber	in $\rightsquigarrow$P		(Theorem 17)
WIDTIO	in $\rightsquigarrow$P		(Theorem 17)

Table 1. The compilability of model checking of revision of Horn clauses

The most evident fact about Table 1 is that different operators have different properties w.r.t. compilability. This is especially surprising since complexity is in many cases very similar. The fact that Forbus' revision is still Σ_2^p-complete, despite the restriction we impose (T and P are Horn) and the fact we allow compilation, seems to prove that this operator is very complex even in very simple cases (the fact that its complexity does not decrease using Horn formulas has already been pointed out by Eiter and Gottlob [1992]).

We give, as an example, the proof of the compilability of Dalal's revision, for the problem of model checking in the Horn case. We need two preliminary lemmas. The first one is implicit in Lemma 7.1 of Eiter and Gottlob's paper [Eiter and Gottlob 1992]. We give a full proof for the sake of completeness.

LEMMA 1. *Let γ be a clause over X, and γ^{neg} be the clause obtained by replacing any positive literal x_i in γ with $\neg y_i$. Then, γ is satisfied by a model $X_1 \subseteq X$ if and only if $X_1 \cup Y_1$ is a model of γ^{neg}, where $Y_1 = \{y_i \mid x_i \notin X_1\}$.*

PROOF. Let γ be the clause

$$\gamma = x_{i_1} \vee \cdots \vee x_{i_k} \vee \neg x_{i_{k+1}} \vee \cdots \vee \neg x_{i_m}$$

By definition, γ^{neg} is

$$\gamma^{neg} = \neg y_{i_1} \vee \cdots \vee \neg y_{i_k} \vee \neg x_{i_{k+1}} \vee \cdots \vee \neg x_{i_m}$$

Let X_1 be a model of γ. We will prove that $X_1 \cup Y_1$ is a model of γ^{neg}, where $Y_1 = \{y_i \mid x_i \notin X_1\}$. Since $X_1 \models \gamma$, there exists an index j such that $1 \leq j \leq k$ and $x_j \in X_1$, or $k+1 \leq j \leq m$ and $x_j \notin X_1$. In the first case, $y_j \notin Y_1$, and since γ^{neg} contains $\neg y_j$, we have $X_1 \cup Y_1 \models \gamma^{neg}$. In the other case, $x_j \notin X_1$ and $\neg x_j \models \gamma^{neg}$, thus $X_1 \cup Y_1 \models \gamma^{neg}$.

Let us assume that $X_1 \cup Y_1 \models \gamma^{neg}$, where $Y_1 = \{y_i \mid x_i \notin X_1\}$. We will prove that X_1 is a model of γ. Since $X_1 \cup Y_1 \models \gamma^{neg}$, there must be an index j such that $1 \leq j \leq k$ and $y_j \notin Y_1$, or $k+1 \leq j \leq m$ and $x_j \notin X_1$. In the first case, $x_j \in X_1$ and $x_j \models \gamma$, thus $X_1 \models \gamma$. In the other case, $x_j \notin X_1$ and $\neg x_j \models \gamma$, thus $X_1 \models \gamma$. □

Notice that γ^{neg} is a Horn clause. This lemma is useful since it relates the models of γ, that is a non-Horn clause, with the models of γ^{neg}, which is Horn. Since we want to prove the NP-completeness of Dalal's revision operator, we need to translate a general (that is, not necessarily Horn) formula into the Horn formulas T and P, plus the model M. The lemma above gives a way for translating a general formula into a Horn formula, and gives a relation about their models.

The next lemma is the core of the proof of non-compilability for model checking. Indeed, it proves that there is a kind of reduction from satisfiability to model checking of a generic revision operator that proves the hardness of model checking of that operator w.r.t. compilability.

LEMMA 2. *Let $\Pi_X = \{\gamma_1, \ldots, \gamma_k\}$ be the set of all the clauses of three literals over the alphabet $X = \{x_1, \ldots, x_n\}$. If there exist three polynomial functions r, s, and t, such that*

$$\Pi \text{ is satisfiable} \quad \text{iff} \quad r(\Pi) \models s(\Pi_X) * t(\Pi_X)$$

for any $\Pi \subseteq \Pi_X$, then MC is $\|\!\rightarrow$NP-hard.

PROOF. This lemma is an easy consequence of Theorem 2 of [Liberatore 1998], showing that the existence of a polynomial monotonic reduction implies the hardness w.r.t. a compilability class. For the case of model checking in belief revision, the definition of monotonic polynomial reduction can be given as follows: a monotonic polynomial reduction is a triple of polynomial functions r, f, v such that for any two sets of clauses Π_1 and Π_2 over the same set of variables X with $\Pi_1 \subseteq \Pi_2$, we have:

$$\Pi_1 \text{ satisfiable} \;\; \text{iff} \;\; r(\Pi_1) \models f(\Pi_2) * v(\Pi_2)$$

Let us assume that Π is satisfiable if and only if $r(\Pi) \models s(\Pi_X) * t(\Pi_X)$. We show a monotonic polynomial reduction: let $f(\Pi) = s(\Pi_X)$ and $v(\Pi) = t(\Pi_X)$. For any Π_1 and Π_2 over the same alphabet X, with $\Pi_1 \subseteq \Pi_2$, it holds:

$$\Pi_1 \text{ is satisfiable} \quad \text{iff} \quad r(\Pi_1) \models s(\Pi_X) * t(\Pi_X) \quad \text{iff} \quad r(\Pi_1) \models f(\Pi_2) * v(\Pi_2)$$

This holds because Π_1 and Π_2 are sets of clauses over the same set of variables X, thus $f(\Pi_1) = s(\Pi_X) = f(\Pi_2)$ and $v(\Pi_1) = t(\Pi_X) = v(\Pi_2)$. As a result, r, f, v is a monotonic polynomial reduction, thus the problem is $\|\!\rightarrow$NP-hard. □

Using the above two lemmas, proving the hardness of Dalal's operator amounts to finding a reduction satisfying the condition of the above lemma, and this is made simpler by the first lemma, relating models of non-Horn formulas with models of Horn formulas. The proof of membership of Dalal's revision in $\rightsquigarrow$NP is much more simpler.

THEOREM 10. *Model Checking for Dalal's revision is $\|\!\rightarrow$NP-hard, in $\rightsquigarrow$NP.*

PROOF. *Membership:* Determine off-line the value of the minimal distance between a model of T and a model of P, that is, the minimal value δ of $Diff(I, J)$ for $I \in Mod(T)$ and $J \in Mod(P)$. By definition, $M \models T *_D P$ if and only if there exists a model M of T such that $Diff(M, I) = \delta$. This subproblem is in NP, as it can be solved by guessing a model M of T, and then check its distance to I.

Hardness: Let $\Pi = \{\gamma_{i_1}, \ldots, \gamma_{i_m}\}$ be a set of propositional clauses over $X = \{x_1, \ldots, x_n\}$, each composed of three literals. Let $\Pi_X = \{\gamma_1, \ldots, \gamma_k\}$ be the set of *all* clauses of three literals built over X. This set has size polynomial w.r.t. X.

We prove that Π is satisfiable if and only if $M \models T *_D P$, where

$$T \;=\; \{c_i \rightarrow \gamma_i^{neg} \mid \gamma_i \in \Pi_X\} \cup \{\neg x_i \vee \neg y_i \mid x_i \in X\}$$

$$P = (\wedge X) \wedge (\wedge Y)$$
$$M = \{c_i \mid \gamma_i \in \Pi\} \cup X \cup Y$$

In these formulas, $C = \{c_1, \ldots, c_k\}$ is a set of new variables, appearing nowhere else, one-to-one with Π_X, and γ_i^{neg} is obtained from γ_i by replacing each positive occurrence of x_i with $\neg y_i$. Note that T and P are a Horn theory and a Horn formula, respectively. Furthermore, γ_i is satisfied by $G \subseteq X$ if and only if $G \cup \{y_i \mid x_i \notin X\}$ is a model of γ_i^{neg}.

Consider the pair of models $I = X$ and $J = X \cup Y$. We have that $I \models T$, $J \models P$, and $|Diff(I, J)| = n$, where n is the number of variables (in X). Notice also that $|Diff(L, O)|$, where L and O are models of T and P respectively, cannot be less than n, since T implies $\neg x_i \vee \neg y_i$, while P has $x_i \wedge y_i$: for each i, either x_i or y_i must be given a different truth value by L and O. As a result, $|Diff(L, O)| \geq n$. This means that the minimal number of elements in $Diff$ is exactly n.

Assume Π satisfiable. Let G be a model of Π, and consider the model

$$N = \{c_i \mid \gamma_i \in \Pi\} \cup G \cup \{y_i \mid x_i \notin G\}$$

Since N is a model of T, and $|Diff(N, M)| = n$, the model M is one of the models of the revised knowledge base $T *_D P$.

On the converse, suppose that $M \models T *_D P$. We prove that there exists a model of Π. First of all, $M \models T *_D P$ implies by definition that there exists a model N of T such that $|Diff(N, M)| = n$. Let $X_1 = N \cap X$ and $Y_1 = N \cap Y$. Since T implies $\neg x_i \vee \neg y_i$ and P implies $x_i \wedge y_i$, either x_i or y_i must be in $Diff(N, M)$, for each i. Since the number of elements of $Diff(N, M)$ must be n, it follows that a. $Diff(N, M)$ is composed only of variables in $X \cup Y$; and b. $Y_1 = \{y_i \mid x_i \notin X_1\}$. As a result, $N \cap C = M \cap C$. Thus, for each $\gamma_i \in \Pi$, it must be $c_i \in N$. Since N is a model of T and T implies $c_i \rightarrow \gamma_i^{neg}$, it holds also that $X_1 \cup Y_1$ is a model of any γ_i^{neg} such that $\gamma_i \in \Pi$, which implies that X_1 is a model of any clause in Π. □

The result of compilability of query answering are reported in Table 2. As in the case of model checking, the revision operators seem to be divided in three classes: the first one is composed of the easier ones (SSU, Weber, WIDTIO), the second one is composed of the other ones except Forbus' revision, while the hardest one is as usual Forbus' revision. The only difference, w.r.t. model checking, is that GFUV is not among the easiest operators. The high complexity of Forbus' revision is at this point not very surprising. Let us try to explain this result. The other model-based operators (Winslett, Borgida, Satoh) are based on a criterion of closeness between models, namely, the set of variables on which two models differ is a measure of their distance. In the case in which T and P are Horn, it is quite easy to express facts about distances with Horn formulas. On the other hand, if the distance between models is the number of literals on which they differ, even very simple facts about distances cannot be expressed as Horn formulas, thus T and P being Horn does not give any advantage. This explanation is leaking in that the properties of Dalal's revision are not explained. For Dalal's revision, indeed, the reason of the decrease of complexity is completely different. Indeed, it is true that complexity does not decrease by restricting T and P to be Horn. On the other hand, we are studying compilability, not complexity. If T and P are given off-line, we can spend lot of

time for computing the minimal distance between models of T and models of P. Using this distance, solving model checking and query answering becomes much simpler.

	Query Answering		
GFUV	in $\rightsquigarrow$coNP	$\|\!\rightsquigarrow$coNP-hard	(Theorem 18)
Dalal	in $\rightsquigarrow$coNP	$\|\!\rightsquigarrow$coNP-hard	(Theorem 19)
Forbus	in $\rightsquigarrow\Pi_2^p$	$\|\!\rightsquigarrow$NP-hard, and $\|\!\rightsquigarrow$coNP-hard	(Theorem 20)
Satoh	in $\rightsquigarrow$coNP	$\|\!\rightsquigarrow$coNP-hard	(Theorem 19)
Borgida/Winslett	in $\rightsquigarrow$coNP	$\|\!\rightsquigarrow$coNP-hard	(Theorem 19)
SSU	in $\rightsquigarrow$P		(Theorem 21)
Weber	in $\rightsquigarrow$P		(Theorem 21)
WIDTIO	in $\rightsquigarrow$P		(Theorem 22)

Table 2. The compilability of query answering of revision of Horn clauses

5. COMPACT REPRESENTATIONS OF REVISION

In Table 3 we summarize the results of existence of compact representations for the belief revision operators introduced. The negative results are obtained from the results of compilability. We give the explicit representation of $T * P$ in the cases in which a compact one exists.

Notice that there are some results missing in the table, and namely in the first column. This happens when we did not succeed in finding a compact representation w.r.t. logical equivalence, for a revision whose model checking is in $\rightsquigarrow$P. Indeed, in such cases, we cannot prove, using the compilability classes, that a compact representation of this kind does not exist. Since we do not have found a compact representation, we cannot prove that it exists, neither that it does not. We conjecture, however, that in all the unknown cases, a compact representation does not exist.

	form of equivalence		
	logical	model	query
GFUV	?	YES (Theorem. 26)	YES (Theorem 26)
Dalal	NO	NO	YES [Cadoli et al. 1999]
Forbus	NO	NO	NO
Satoh	NO	NO	YES (Theorem 28)
Borgida/Winslett	NO	NO	YES (Theorem 27)
SSU	?	YES (Theorem 12)	YES (Theorem 11)
Weber	?	YES (Theorem 12)	YES [Cadoli et al. 1999]
WIDTIO	YES	YES	YES

Table 3. Does $*$ have a compact representation?

The negative results in the table follow from the results of Table 1 and Theorems 7,8,9. For example, since model checking of Dalal's revision is $\|\!\rightsquigarrow$NP-hard, it follows from Theorem 8 that it does not have a compact representation w.r.t. logical and model equivalence. Some results of Table 3 are also easy consequences of the results by Cadoli, Donini, Liberatore, and Schaerf [1999]: since Dalal's and

Weber's revisions have a compact representation w.r.t. query equivalence in the general case, this representation also holds for the Horn case. The existence of a compact representation w.r.t. query equivalence for GFUV follows from the fact that model equivalence implies query equivalence.

Let us now consider the difference between the general (non-Horn) case and the Horn case. The result of WIDTIO can be trivially expressed as a formula which is at most as large as $T \wedge P$. The results proved by Cadoli, Donini, Liberatore, and Schaerf [1999] shows that, except for Dalal's and Weber's revision for the case of query equivalence, the result of revision can be superpolinomially larger than the knowledge base and the revising formula. In Table 4 we emphasize the difference between the general case and the Horn case. Namely, an entry "NO/YES" means that a compact representation exists for the Horn case, but not for the general case. Entries "YES" and "NO" means that the properties of the revision operator does not change. Note that the results by Cadoli, Donini, Liberatore, and Schaerf [1999] do not include the standard semantics. Moreover, all negative results for logical equivalence are indeed obtained for model equivalence (although this form of equivalence is not mentioned in the paper).

	form of equivalence		
	logical	model	query
GFUV	NO/?	NO/YES	NO/YES
Dalal	NO	NO	YES
Forbus	NO	NO	NO
Satoh	NO	NO	NO/YES
Borgida/Winslett	NO	NO	NO/YES
Weber	NO/?	NO/YES	YES
WIDTIO	YES	YES	YES

Table 4. Compact representation in the general/Horn cases.

Clearly, the restriction to Horn formulae does not give an advantage in all cases. The only cases in which compact representations exists only in the Horn case are: GFUV (both model and query equivalence); Weber (model equivalence only); Satoh, Borgida, and Winslett (query equivalence only).

Finally, we remark that, while there is a general technique for proving that a revision operator does not have a compact representation, showing that it does have a compact representation must be done using a different proof for each operator. Namely, given T and P, we have to exhibit a formula, equivalent to $T * P$, whose size is polynomial in the size of T and P.

5.1 Compact Representations

For some revision operators, we can prove that there exist formulas equivalent to $T*P$. Some results are easy consequences of definition: since $T *_{FM} P$ and $T *_{Wit} P$ are always equivalent to subsets of $T \cup \{P\}$, the definition of these two revisions gives a compact representation of them.

In order to show how the proofs of existence of compact representations work for the non-trivial cases, we give the full proof for Weber's revision operator, while the full proof for the other operators are in the Appendix.

In the general case (i.e. when T and P are allowed to be arbitrary formulas) Dalal's and Weber's operators have a compact representation w.r.t. query equivalence [Cadoli et al. 1999]. In the Horn case, we prove that this holds also considering equivalence w.r.t. models. We show, as an example, the compact representation for revision operators based on the "unreliable set" assumption. The proofs for all the other operators are in Appendix D.

We need first a result about circuits. Consider two sets of variables

$$B = \{b_j^i \mid 1 \leq i \leq n,\ 1 \leq j \leq m\}$$
$$H = \{h_j^i \mid 1 \leq i \leq n,\ 1 \leq j \leq m\}$$

We give a one-to-one correspondence between the interpretations over $B \cup H$ and the sets of m clauses over a set of n variables. In other words, we associate a truth evaluation over $B \cup H$ to any given set of clauses.

Let $\Pi = \{\gamma_1, \ldots, \gamma_m\}$ be a set of clauses. The interpretation associated with Π is

$$m(\Pi) = \{b_j^i \mid \neg x_j \in \gamma_i\} \cup \{h_j^i \mid x_j \in \gamma_i\}$$

Roughly speaking, we use the value associated to the b_j^i's to represent the negative literals in the clauses, while the h_j^i represents the positive ones. For example, let $X = \{x_1, x_2, x_3\}$ be the set of variables, and $\Pi = \{x_1, \neg x_2 \vee x_3\}$ the set of clauses. The interpretation $m(\Pi)$ associated to Π is $\{h_1^1, b_2^2, h_3^2\}$. Note that $m(\Pi)$ is an interpretation over $B \cup H$ and *not an interpretation over* X. In other words, $m(\Pi)$ is an interpretation over a different alphabet. We remark that a set of clauses Π is associated to a single interpretation $m(\Pi)$ over a different alphabet, since this may be confusing in the sequel. Notice that the function m is one-to-one, thus given an interpretation $R \subseteq B \cup H$, there exists exactly one set of clauses Π such that $m(\Pi) = R$. We denote this set by $\Pi = m^{-1}(R)$ as usual.

We define the function s, from interpretations over $B \cup H$ to truth values, as follows.

$$s(R) = \begin{cases} \mathsf{true} & \text{if } m^{-1}(R) \text{ is satisfiable and Horn} \\ \mathsf{false} & \text{otherwise} \end{cases}$$

In words, the function s tells whether the set of clauses associated to an interpretation is satisfiable or not.

For example, the set of clauses corresponding to $R = \{b_1^1, h_1^2\}$ is $\{\neg x_1, x_1\}$, which is unsatisfiable. As a result, $s(R) = \mathsf{false}$. Another example is $S = \{b_1^1, h_2^1, h_1^2\}$: the set of clauses associated with S is $\{\neg x_1 \vee x_2, x_1\}$ which is satisfiable. Thus $s(S) = \mathsf{true}$.

Now, s is a function from truth assignments on $B \cup H$ to $\{\mathsf{true}, \mathsf{false}\}$, thus it can be represented with a propositional formula over the alphabet $B \cup H$. This formula can be represented also as a circuit of polynomial size.

LEMMA 3. *There exists a circuit* sat *(over the variables $B \cup H$) such that:*

(1) Its size is polynomial in $|B| + |H|$;

(2) $R \subseteq B \cup H$ is a model of sat *if and only if $m^{-1}(R)$ is satisfiable and is a Horn set.*

PROOF. Although it is possible to give a constructive proof (an explicit representation of sat), for the sake of simplicity we prove it indirectly.

It is well known that deciding the satisfiability of a set of Horn clauses is a polynomial problem [Dowling and Gallier 1984]. Checking whether a set of clauses is composed of Horn clauses only is also polynomial. Thus, given an interpretation $R \subseteq B \cup H$, it is possible to decide in polynomial time whether its associated set of clauses is Horn and satisfiable. Thus, deciding whether R is a model of the circuit sat is polynomial. Now, every polynomial-time algorithm can be encoded in a polynomial-size circuit [Boppana and Sipser 1990, Theorem 2.2]. Thus, it is possible to represent sat with a circuit whose size is polynomial in $|B| + |H|$. □

This circuit sat has a model $R \subseteq B \cup H$ if and only if R is associated to a satisfiable set of Horn clauses.

In this section we consider only the revision operators defined in terms of a set of unreliable variables. The results about the other ones are given in Appendix D. The general definition of these revision operators is the following one.

$$Mod(T *_\Omega P) = \{J \in Mod(P) \mid \exists I \in Mod(T) \ . \ I \backslash \Omega = J \backslash \Omega\}$$

Given the set Ω, one can find a compact representation w.r.t. query equivalence for $*_\Omega$. The following theorem has already been published [Cadoli et al. 1999], and is reported here for the sake of completeness.

THEOREM 11. *There exists a formula T_1, which is a. polynomial in the size of T and P; and b. query equivalent to $T *_\Omega P$.*

PROOF. Consider $T_1 = T[\{x_i/y_i \mid x_i \in \Omega\}] \wedge P$, where $\{y_i\}$ is a set of new variables not in X. We prove that $T_1 \equiv^Q_X T *_\Omega P$.

Let R be a formula such that $Var(R) \subseteq X$. We prove that $T *_\Omega P \models R$ if and only if $T_1 \models R$.

Suppose that $T *_\Omega P \models R$. Consider $K \in Mod(T_1)$: we will prove that $K \in Mod(R)$. By definition of T_1, $K \backslash X$ must be a model of P. Furthermore, K is a model of the theory obtained by replacing each occurrence of each $x_i \in \Omega$ with y_i in T. As a result, $I = K \backslash \Omega \cup \{x_i \mid x_i \in \Omega \ \ y_i \in K\}$ is a model of T. This proves that there exists a model I of T such that $I \backslash \Omega = (K \backslash X) \backslash \Omega$, thus $K \backslash X$ is a model of $T *_\Omega P$, which implies that it is also a model of R, since $T *_\Omega P \models R$. Since R does not contain any variables not in X, it holds also that K is a model of R.

Suppose that $T_1 \models R$, and consider $J \in Mod(T *_\Omega P)$. We will prove that $J \models R$. By definition of $*_\Omega$, $J \models P$, and there exists a model $I \in Mod(T)$ such that $I \backslash \Omega = J \backslash \Omega$. Now, consider that

$$I \backslash \Omega \cup \{y_i \mid x_i \in \Omega \ \ x_i \in I\}$$

is a model of $T[\{x_i/y_i \mid x_i \in \Omega\}]$, while J is a model of P. Since $I \backslash \Omega = J \backslash \Omega$, we have that

$$J \cup \{y_i \mid x_i \in \Omega \ \ x_i \in I\}$$

is a model of T_1. Thus

$$J \cup \{y_i \mid x_i \in \Omega \ \ x_i \in I\} \models R$$

Since R does not contain any variable not in X, this proves that $J \models R$. $\square$

The last theorem shows that $*_\Omega$ has a compact representation w.r.t. query equivalence. The next theorem regards the existence of a compact representation w.r.t. model equivalence.

THEOREM 12. *There exists a compact representation w.r.t. model equivalence for* $*_\Omega$.

PROOF. A model J of P is also a model of $T *_\Omega P$ if and only if there exists a model of T that agrees with it at least on the variables not in Ω. Given a model $J \in Mod(P)$, this is equivalent to saying:

$$J \models T *_\Omega P \quad \Leftrightarrow \quad \text{the Horn set } \left[T \cup \bigcup_{x_i \notin \Omega \ x_i \in J} x_i \cup \bigcup_{x_i \notin \Omega \ x_i \notin J} \neg x_i \right] \text{ is satisfiable}$$

Let $T = \{\gamma_1, \ldots, \gamma_m\}$, $|X| = n$, and $|\Omega| = k$. We give a circuit that represents $T *_\Omega P$. Let B and H be two sets of $n(m+n)$ variables each, defined as follows.

$$\begin{aligned} B &= \{b_j^i \mid 1 \leq i \leq n \ \ 1 \leq j \leq m+n\} \\ H &= \{h_j^i \mid 1 \leq i \leq n \ \ 1 \leq j \leq m+n\} \end{aligned}$$

Consider the circuit T_1 built as follows.

$$T_1 = P \wedge \mathsf{sat} \wedge \text{REPR}_T \wedge \text{REPR}_X$$

where sat is the circuit of Lemma 3 over the variables $B \cup H$, REPR_X and REPR_T are the formulas defined as follows.

$$\begin{aligned} \text{REPR}_T &= \bigwedge\{b_j^i \mid \neg x_j \in \gamma_i\} \wedge \bigwedge\{\neg b_j^i \mid \neg x_j \notin \gamma_i\} \wedge \\ &\quad \bigwedge\{h_j^i \mid x_j \in \gamma_i\} \wedge \bigwedge\{\neg h_j^i \mid \neg x_j \notin \gamma_i\} \\ \text{REPR}_X &= \bigwedge_{x_j \notin \Omega} \{(x_j = \neg b_j^{m+j}) \wedge (x_j = h_j^{m+j})\} \wedge \bigwedge_{x_j \notin \Omega} \{\neg b_j^{m+i} \wedge \neg h_j^{m+i} | i \neq j\} \wedge \\ &\quad \bigwedge_{\substack{x_j \in \Omega \\ m+1 \leq i \leq m+n}} (b_j^i \wedge h_j^i) \end{aligned}$$

In words: $B \cup H$ is used to represents a set of $m+n$ clauses over a set of n variables; the formula REPR_T says that the first m clauses represented by $B \cup H$ are the clauses in T, and the formula REPR_X says that the last n clauses represented by $B \cup H$ must represent the value of the variables x_i's, i.e. the $n+i$'th clause is x_i if x_i is true, $\neg x_i$ otherwise. In other words, a model of this circuit must have a value of the first $n \times m$ elements of B and H that represents T, while the other $n \times n$ must represents the value of x_j if $x_j \notin \Omega$, otherwise they must represent tautological clauses (a formula $b_j^i \wedge h_j^i$ represents a clause in which both x_j and $\neg x_j$ are present, and this is a tautology).

Let for example $T = \{x_1, \neg x_1 \vee x_2\}$ and $P = \neg x_2$. Formula REPR_T is used to encode the clauses x_1 and $\neg x_1 \vee x_2$ using the variables $b_1^1, \ldots, b_2^2$ and $h_1^1, \ldots, h_2^2$. Namely, the four variables $b_1^1, h_2^1, b_2^1, h_2^1$ represent the first clause.

$$\begin{array}{rcll} \text{REPR}_T & = & \neg b_1^1 \wedge h_1^1 \wedge \neg b_2^1 \wedge \neg h_2^1 \wedge & \text{(the first clause is } x_1\text{)} \\ & & b_1^2 \wedge \neg h_1^2 \wedge \neg b_2^2 \wedge h_2^2 & \text{(the second clause is } \neg x_1 \vee x_2\text{)} \end{array}$$

Moreover, $\Omega = \{x_2\}$, which allows for finding REPR_X.

$$\text{REPR}_X = (x_1 = \neg b_1^3) \wedge (x_1 = h_1^3) \wedge (\neg b_2^3 \wedge \neg h_2^3) \wedge (b_1^4 \wedge h_1^4 \wedge b_2^4 \wedge h_2^4)$$

Indeed, since $x_1 \notin \Omega$, we have to impose that b_i^3 and h_i^3 represent the clause composed of x_1 only. Since $x_2 \in \Omega$, the clause represented by b_i^4 and h_i^4 must be tautological. This is obtained by setting to true the value of all variables: this way, the clause represented by these variables is $x_1 \vee \neg x_1 \vee x_2 \vee \neg x_2$.

Finally, T_1 is:

$$\begin{array}{rcl} T_1 & = & \neg x_2 \wedge \mathsf{sat} \wedge \Big(\neg b_1^1 \wedge h_1^1 \wedge \neg b_2^1 \wedge \neg h_2^1 \wedge b_1^2 \wedge \neg h_1^2 \wedge \neg b_2^2 \wedge h_2^2\Big) \wedge \\ & & \Big((x_1 = \neg b_1^3) \wedge (x_1 = h_1^3) \wedge (\neg b_2^3 \wedge \neg h_2^3) \wedge (b_1^4 \wedge h_1^4 \wedge b_2^4 \wedge h_2^4)\Big) \end{array}$$

The subformula sat has not been explicitely expanded: this is the formula that is satisfied by a truth assignment over $B \cup H$ if and only if the set of clauses it represents is both Horn and satisfiable. By Lemma 3 such formula exists and has polynomial size. Before formally prove that such reduction works, we note that the result of revising T with P can be represented by the much shorter formula $x_1 \wedge \neg x_2$. What we prove is not that formula T_1 is the shortest possible representation of $T *_\Omega P$, but only that it is equivalent to it, and that it has polynomial size.

The sat circuit is used to test whether the set of clauses represented by $B \cup H$ is satisfiable. Now, $B \cup H$ is T together with a set of clauses representing the value in a model of P of the variables not in Ω. As a result, the whole circuit has a model K if and only if

(1) $K \cap X \in \mathit{Mod}(P)$

(2) the set of clauses $T \cup \bigcup_{x_i \notin \Omega \ x_i \in J} x_i \cup \bigcup_{x_i \notin \Omega \ x_i \notin J} \neg x_i$ is satisfiable.

which is equivalent to say that $K \cap X \in \mathit{Mod}(T *_\Omega P)$. This proves that given a model K of T_1, we can determine in polynomial time a model of $T *_\Omega P$, since $K \cap X$ is a model of $T *_\Omega P$. The converse also holds: given a model J of $T *_\Omega P$, we can complete it to obtain a model of T_1 in the following manner:

$$\begin{array}{rcl} K & = & J \cup \{b_j^i \mid \gamma_i \models \neg x_j\} \cup \{h_j^i \mid \gamma_i \models x_j\} \cup \\ & & \{b_j^{m+j} \mid x_j \notin J,\ x_i \notin \Omega\} \cup \{h_j^{m+j} \mid x_j \in J,\ x_i \notin \Omega\} \end{array}$$

Since this model can be determined in polynomial time, the claim is proved. □

6. OPEN PROBLEMS

An open problem of our analysis is the compact representation w.r.t. logical equivalence of some belief revision operators, for example GFUV and the standard semantics update. We noted that we have no actual procedure to prove that such a polynomial-size representation does not exist. This problem is relevant for operators for which we can find a compact representation w.r.t. model equivalence, but we are not able to find an explicit representation w.r.t. logical equivalence.

We give some considerations. It is known that

$$\text{MIY} = Form(\{G \subseteq X \mid |G| < |X|/2\})$$

cannot be represented by an AC^0 circuit. However, it is query equivalent over X to the following Horn theory:

$$\begin{aligned} T \;=\; & \{r^1_{n/2}\} \cup \{r^i_j \rightarrow r^i_{j-1} \mid 1 \leq i \leq n+1,\ 1 \leq j \leq n/2\} \cup \\ & \{r^i_j \rightarrow r^{i+1}_{j-1} \mid 1 \leq i \leq n,\ 1 \leq j \leq n/2\} \cup \\ & \{r^i_j \wedge x_i \rightarrow r^{i+1}_j \mid 1 \leq i \leq n,\ 1 \leq j \leq n/2\} \cup \{\neg r^{n+1}_1\} \end{aligned}$$

that is, $\text{MIY} \equiv^M_X T$. Now, consider the revision $T *_{SSU} P$, where

$$P = \wedge\{r^i_j \mid 1 \leq i \leq n+1,\ 1 \leq j \leq n/2\}$$

By definition, $T *_{SSU} P$ is *logically* equivalent to $\text{MIY} \wedge P$, and MIY and P do not share variables. As a result, $T *_{SSU} P$ cannot be represented by an AC^0 circuit. The fact that $\text{MIY} \notin \text{AC}^0$ implies that no circuit with a constant number of levels is equivalent to MIY, which implies that such circuits cannot represent $T *_{SSU} P$ either. Note that this result is not conditioned to the non-collapse of the polynomial hierarchy, that is, it holds even if the polynomial hierarchy collapses (e.g. P=NP). However, this proof is ad hoc for the standard semantics: other revision operators may require different proof tecniques.

Finally, the compilability/compactability in the iterated case is still an open problem for many revision operators. Some of the results of the present paper extend to the iterated case, but not all. Namely, all hardness results clearly hold in the iterated case, as well as all negative results about compact representations. The same do not hold for the positive results: suppose for instance that a revision operator $*$ "doubles" the size of a knowledge base at each revision step, that is, $T * P$ can be represented as a formula which is two times the size of T. After n revision steps, this representation is $2^{|n|}$ larger than the original knowledge base, which means that a revision operator which admit a compact representation for a single revision does not have compact representations in the iterated case.

7. CONCLUSIONS

In this paper we have investigated two problems about belief revision: their compilability and the existence of compact representations. These two problems are closely related, since negative results about compilability (decide if the complexity decreases when a preprocessing is allowed), can be used to prove that there is no

compact representation. For some belief revision operators we found a compact representation for the revised knowledge base.

In order to prove the non-compilability of operators, we used a particular kind of reductions, different from the usual polynomial-time reductions: in some cases, two operators have the same computational complexity but different compilability properties: the complexity of an operator is not enough to characterize its properties from the point of view of the compilability.

Let us now discuss the significance of the results presented in this paper. We consider two separate cases:

(1) revisions are not very frequent;
(2) we need to revise our knowledge base often.

We can see that the results presented here are significant in both cases. Consider the case in which revisions are sparse. This means that between one revision and the next one there is a long interval. As a result, we can use this spare time to compile the knowledge base and the revising formula in order to obtain a fast answer to queries. Since the revised knowledge base will not probably be revised again shortly afterwards, this compiled structure may be useful for answering to many queries, thus making the querying problem simpler.

On the other hand, if there are a lot of revisions, the existence of compact representations allows for simplifying the problem from another point of view. Indeed, let us consider the case in which we have a knowledge base T and a sequence of revising formulas $P_1, \ldots, P_m$. If m is very large, the total size of the input is dominated by the number of revising formulas. On the other hand, if the result of each revision can be represented in compact form, this means that we can forget the history of the past revisions and store just the current one. This is clearly impossible if such compact representation does not exist.

ACKNOWLEDGMENTS

The author wishes to thank Marco Schaerf for his comments on an earlier version of this paper.

APPENDIX

A. COMPILABILITY OF MODEL CHECKING

In this section we give the proofs of the compilability results about belief revision in the Horn case. In order to prove hardness, we use the results of Lemma 1 and Lemma 2.

THEOREM 13. *Model Checking for GFUV is in* $\rightsquigarrow$P.

PROOF. Since model checking is in P, as proved by Liberatore and Schaerf [2000], and P $\subset \rightsquigarrow$P, the claim follows. □

THEOREM 14. *Model Checking for Forbus' revision is* $\|\!\rightsquigarrow$coNP*-hard and* $\|\!\rightsquigarrow$*NP-hard, in* $\rightsquigarrow\Sigma_2^p$.

PROOF. *Membership:* The $\rightsquigarrow\Sigma_2^p$ membership follows from the complexity (which is Σ_2^p) of the model checking, as proved by Liberatore and Schaerf [2000].

Hardness: We prove the $\|\!\rightsquigarrow$coNP-hardness using a statement similar to that of Lemma 2. Let $\Pi = \{\gamma_{i_1}, \ldots, \gamma_{i_m}\}$ be a set of clauses over X, each composed of three literals. Let $\Pi_X = \{\gamma_1, \ldots, \gamma_k\}$ be the set of all the three-literal clauses over X. We prove that Π is unsatisfiable if and only if $M \models T *_F P$, where

$$\begin{aligned} T &= X \cup Y \cup Z \\ P &= (\wedge\{\neg z_i \mid z_i \in Z\}) \vee [\bigwedge_{1 \leq i \leq n} (\neg x_i \vee \neg y_i) \wedge \bigwedge_{\gamma_i \in \Pi_X} (c_i \rightarrow \gamma_i^{neg})] \\ M &= X \cup Y \cup \{c_i \mid \gamma_i \in \Pi\} \end{aligned}$$

where $Z = \{z_1, \ldots, z_{n+1}\}$, $Y = \{y_1, \ldots, y_n\}$ and $C = \{c_1, \ldots, c_k\}$ are sets of new variables, where $n = |X|$ and $k = |\Pi_X|$. The clause γ_i is satisfied by $G \subseteq X$ if and only if $G \cup \{y_i \mid x_i \notin G\}$ satisfies γ_i^{neg}.

Formula P, while it is not in Horn form, can be shown to be equivalent to an Horn formula of polynomial size. Indeed, by applying distributivity between $\vee$ and $\wedge$, it can be rewritten as the conjunction of $\wedge\{\neg z_i \mid z_i \in Z\}) \vee \bigwedge_{1 \leq i \leq n}(\neg x_i \vee \neg y_i)$ and $\wedge\{\neg z_i \mid z_i \in Z\}) \vee \bigwedge_{\gamma_i \in \Pi_X}(c_i \rightarrow \gamma_i^{neg})$. Each of these two subformulae can in turns be rewritten as a set of Horn clauses. The first one is indeed equivalent to $\bigwedge_{z_j \in Z,\ 1 \leq i \leq n} \neg z_j \vee \neg x_i \vee \neg y_i$, while the second one is equivalent to $\bigwedge_{z_j \in Z,\ \gamma_i \in \Pi_X} \neg z_j \vee \neg c_i \vee \gamma_i^{neg}$.

First, T has exactly one model for each subset of C. As for Dalal's proof, if $I \models T$ and $J \models P$, then $Diff(I, J) \geq n$, since T implies $x_i \wedge y_i \wedge z_i$, while P implies $\neg x_i \vee \neg y_i \vee \neg z_i$.

Suppose Π satisfiable. Let $G \subseteq X$ be a model of Π. Consider the models

$$\begin{aligned} I &= X \cup Y \cup Z \cup \{c_i \mid \gamma_i \in \Pi\} \\ J &= G \cup \{y_i \mid x_i \notin G\} \cup Z \cup \{c_i \mid \gamma_i \in \Pi\} \end{aligned}$$

We have $|Diff(I, J)| = n$, while $|Diff(I, M)| = n+1$. Furthermore, given any other model N of T, it holds $|Diff(N, M)| = |Diff(N, J)| + 1$, since N can differ from I only for the value of the c_i's, for which J and M agree. As a result, $M \not\models T *_F P$.

Suppose Π unsatisfiable. Let I be the model above. Let $C_1 = \{c_i \mid \gamma_i \in \Pi\}$. The models of P closest to I are of two kinds:

$$\begin{aligned} L &= X \cup Y \cup C_1 \\ N &= X_1 \cup Y_1 \cup Z \cup C_2 \end{aligned}$$

where $X_1 \subseteq X$, $Y_1 \subseteq Y$, and $C_2 \subseteq C_1$. We have $|Diff(I, L)| = n + 1$. Furthermore, since Π is unsatisfiable, from Lemma 1 follows that Y_1 cannot be the "opposite" of X_1. Since $\bigwedge(\neg x_i \vee \neg y_i)$ is true, there must be at least an index i such that $x_i \notin X_1$ and $y_i \notin Y_1$. As a result, $|X_1| + |Y_1| < n$, thus $Diff(I, N) > n$. As a result, M is a model of $T *_F P$.

The proof of $\|\!\rightsquigarrow$NP-hardness is identical to that of Winslett's operator. □

THEOREM 15. *Model Checking for Satoh's revision is $\|\!\rightsquigarrow$NP-hard, in $\rightsquigarrow$*NP.

PROOF. *Membership:* follows from the complexity result. Indeed, model checking is NP-hard when T and P are Horn formulae [Liberatore and Schaerf 2000], and NP $\subset \rightsquigarrow$NP.

Hardness: We prove hardness using Lemma 2. Let $\Pi = \{\gamma_{i_1}, \ldots, \gamma_{i_m}\}$ be a set of clauses over X, each composed of three literals. Let $\Pi_X = \{\gamma_1, \ldots, \gamma_k\}$ be the set of all the clauses of three literals over X. We prove that Π is satisfiable if and only if $M \models T *_S P$, where

$$\begin{aligned} T &= \{c_i \rightarrow \gamma_i^{neg} \mid 1 \leq i \leq k\} \cup \{\neg x_i \vee \neg y_i \mid 1 \leq i \leq n\} \cup \\ &\quad \{x_i \vee y_i \rightarrow z_i \mid 1 \leq i \leq n\} \cup \{(\wedge\{z_i \mid 1 \leq i \leq n\}) \rightarrow r,\ r \equiv u\} \\ P &= \wedge(X \cup Y \cup \{\neg z_i \mid z_i \in Z\}) \wedge \neg r \\ M &= X \cup Y \cup \{u\} \cup \{c_i \mid \gamma_i \in \Pi\} \end{aligned}$$

where $C = \{c_1, \ldots, c_k\}$, $Y = \{y_1, \ldots, y_n\}$, $Z = \{z_1, \ldots, z_n\}$, u, and r are new variables, and γ_i^{neg} is γ_i where each positive literal x_i is replaced by $\neg y_i$. Note that $G \subseteq X$ is a model of a clause γ_i if and only if $G \cup \{y_i \mid x_i \notin G\}$ is a model of γ_i^{neg}.

Assume that Π is satisfiable. Let $G \subseteq X$ be a model of Π. This model can be extended to form a model of T:

$$I = G \cup \{y_i \mid x_i \notin G\} \cup Z \cup \{r, u\} \cup \{c_i \mid \gamma_i \in \Pi\}$$

The difference between I and M is

$$\mathit{Diff}(I, M) = X \backslash G \cup \{y_i \mid x_i \in G\} \cup Z \cup \{r\}$$

We claim that this difference is minimal, i.e. there is no other pair of interpretations J and N such that $J \in \mathit{Mod}(T)$, $N \in \mathit{Mod}(P)$, and $\mathit{Diff}(J, N) \subset \mathit{Diff}(I, M)$.

Let us assume that $\mathit{Diff}(J, N) \subseteq \mathit{Diff}(I, M)$. We will prove that $\mathit{Diff}(I, M) \subseteq \mathit{Diff}(J, N)$.

Let $X_1 = J \cap X$ and $Y_1 = J \cap Y$. Since $X \cup Y \subseteq N$, it follows that

$$\mathit{Diff}(J, N) = X \backslash X_1 \cup Y \backslash Y_1 \cup \ldots$$

Since $\mathit{Diff}(J, N) \cap X \subseteq \mathit{Diff}(I, M) \cap X$, it holds $X \backslash X_1 \subseteq X \backslash G$, which implies $G \subseteq X_1$. Since $X_1 \cup Y_1$ is a model of $\neg x_i \vee \neg y_i$, we have $Y_1 \subseteq \{y_i \mid x_i \notin G\}$, which in turn implies $Y \backslash \{y_i \mid x_i \notin G\} \subseteq Y \backslash Y_1$. Since $Y \backslash \{y_i \mid x_i \notin G\} = \mathit{Diff}(J, N) \cap Y$ and $Y \backslash Y_1 = \mathit{Diff}(I, M) \cap Y$, we have $Y_1 = \{y_i \mid x_i \notin G\}$, and thus $X_1 = G$. This implies $J \cap Z = I \cap Z = Z$, which implies $\{r, u\} \subseteq J$. Thus, $J = I$. Now, the model of P which is closer to I is M, thus $\mathit{Diff}(I, M) \subseteq \mathit{Diff}(J, N)$.

Assume Π unsatisfiable. We consider the models I of T that are closest to M, and then prove that $\mathit{Diff}(I, M)$ is never a minimal difference (i.e. there is always a pair of models J and N that are closer).

Let I be a model of T. Assume that $I \cap C = C_2 \neq M \cap C = C_1$. We will prove that there is a model N of P such that $\mathit{Diff}(I, N) \subset \mathit{Diff}(I, M)$. This is simple to prove:

$$\mathit{Diff}(I, M) = \mathit{Diff}(C_1, C_2) \cup \mathit{Diff}(I \cap (X \cup Y \cup Z \cup \{r, u\}), M \cap (X \cup Y \cup Z \cup \{r, u\}))$$

Now, let

$$N = C_2 \cup (M \cap (X \cup Y \cup Z \cup \{r, u\}))$$

which is model of P: this model is obtained from M by replacing C_1 with C_2. The difference between I and N is given by

$$\mathit{Diff}(I, N) = \mathit{Diff}(I \cap (X \cup Y \cup Z \cup \{r, u\}), M \cap (X \cup Y \cup Z \cup \{r, u\})) \subset \mathit{Diff}(I, M)$$

As a result, the difference between I and M is not minimal.

Now consider the other case, i.e. $I \cap C = M \cap C = C_1$. Let $X_1 = I \cap X$ and $Y_1 = I \cap Y$. Since Π is unsatisfiable, $\{\gamma_i^{neg} \mid \gamma_i \in \Pi\}$ cannot be satisfied by a model $X_1 \cup Y_1$ if $Y_1 = \{y_i \mid x_i \notin X_1\}$. As a result, since $T \cup C_1 \models \{\gamma_i^{neg} \mid \gamma_i \in \Pi\}$, the set Y_1 cannot be the "opposite" of X_1, i.e. there must be an i such that $x_i \notin X_1$ and $y_i \notin Y_1$.

When either x_i or y_i are true in a model of T, z_i is forced to be true in that model. In the model I, however, neither x_i nor y_i is true. Thus, z_i can be true or false. Suppose, without loss of generality, that the condition $x_i \notin X_1$ and $y_i \notin Y_1$ holds only for one value of i. If z_i is true, r also must be true (and thus u also must be true). This is the first possible model of T:

$$I_1 = C_1 \cup X_1 \cup Y_1 \cup Z \cup \{r, u\}$$

However, z_i can also be false. In this case r can be true or false, leading to the possible models I_2 and I_3.

$$\begin{aligned} I_2 &= C_1 \cup X_1 \cup Y_1 \cup Z \backslash \{z_i\} \cup \{r, u\} \\ I_3 &= C_1 \cup X_1 \cup Y_1 \cup Z \backslash \{z_i\} \end{aligned}$$

Since $Diff(I_2, M) \subset Diff(I_1, M)$, we do not need to consider I_1 any more (for sure, $Diff(I_1, M)$ is not a minimal difference). The difference between I_2, I_3 and M are

$$\begin{aligned} Diff(I_2, M) &= X \backslash X_1 \cup Y \backslash Y_1 \cup Z \backslash \{z_i\} \cup \{r\} \\ Diff(I_3, M) &= X \backslash X_1 \cup Y \backslash Y_1 \cup Z \backslash \{z_i\} \cup \{u\} \end{aligned}$$

Now, consider

$$N = C_1 \cup X \cup Y$$

N is a model of P, and

$$Diff(I_3, N) = X \backslash X_1 \cup Y \backslash Y_1 \cup Z \backslash \{z_i\}$$

which is strictly contained both in $Diff(I_2, M)$ and $Diff(I_3, M)$. As a result, there is no model I of T such that $Diff(I, M)$ is minimal. Thus, M is not a model of $T *_S P$. □

THEOREM 16. *Winslett's and Borgida's revisions are $\|\!\rightsquigarrow$NP-hard, in $\rightsquigarrow$*NP.

PROOF. These two operators are in the same compilability class, since the only difference is the check of consistency of $T \wedge \{P\}$, which can be done off-line.

Membership: Both these revisions have an NP model checking, thus $\rightsquigarrow$NP.

Hardness: We prove hardness using Lemma 2. Let $\Pi = \{\gamma_{i_1}, \ldots, \gamma_{i_m}\}$ be a set of clauses, each composed of three literals, and $\Pi_X = \{\gamma_1, \ldots, \gamma_m\}$ be the set of all the clauses of three literals over X. We prove that Π is satisfiable if and only if $M \models T *_W P$, where

$$\begin{aligned} T &= \{c_i \rightarrow \gamma_i^{neg} \mid 1 \leq i \leq m\} \cup \{\neg x_i \vee \neg y_i \mid x_i \in X\} \\ P &= \wedge\{x_i \equiv y_i \mid 1 \leq i \leq n\} \\ M &= \{c_i \mid \gamma_i \in \Pi\} \cup X \cup Y \end{aligned}$$

By Lemma 1, Π is satisfied by a model G if and only if T is satisfied by the model $G \cup \{y_i \mid x_i \notin G\}$.

Let I be a model of T. We prove that the models of P that are closest to I have the same value over the c_i's. Let J be a model of P such that $I \cap C = C_1$ and $J \cap C = C_2 \neq C_1$. We have

$$\mathit{Diff}(I, J) = \mathit{Diff}(C_1, C_2) \cup \mathit{Diff}(I \cap (X \cup Y), J \cap (X \cup Y))$$

Let $K = C_1 \cup (J \cap (X \cup Y))$. Clearly, K is a model P. Furthermore,

$$\mathit{Diff}(I, K) = \mathit{Diff}(I \cap (X \cup Y), J \cap (X \cup Y)) \subset \mathit{Diff}(I, J)$$

This proves that the models of T that are closest to I give the same truth value to the c_i's. As a result, M can be a model of $T *_W P$ if and only if there is a model $I \in \mathit{Mod}(P)$ such that $I \cap C = M \cap C$ and M is one of the model of P that are closest to I.

Let us suppose Π satisfiable. Let G be the model of Π, and consider the following model of T.

$$I = \{c_i \mid \gamma_i \in \Pi\} \cup G \cup \{y_i \mid x_i \notin X_1\}$$

We claim that $\mathit{Diff}(I, M)$ is minimal.

$$\mathit{Diff}(I, M) = X \backslash G \cup Y \backslash \{y_i \mid x_i \notin G\}$$

Consider any other model $N \in \mathit{Mod}(P)$, and assume that $\mathit{Diff}(I, N) \subset \mathit{Diff}(I, M)$.

Suppose for example that $x_i \notin \mathit{Diff}(I, N)$ and $x_i \in \mathit{Diff}(I, M)$. Consider the case $x_i \in I$. We have

a. $x_i \in N$
b. $x_i \notin M$
c. $y_i \notin I$

which imply

a. $y_i \in N$
b. $x_i \notin M$

$\Rightarrow$

a. $y_i \in \mathit{Diff}(I, N)$
b. $y_i \notin \mathit{Diff}(I, M)$

which is in contradiction with the hypothesis $\mathit{Diff}(I, N) \subset \mathit{Diff}(I, M)$.

Consider the case $x_i \notin I$. We have

a. $x_i \notin N$
b. $x_i \in M$
c. $y_i \in I$

which imply

a. $y_i \notin N$
b. $y_i \in M$

$\Rightarrow$

a. $y_i \in \mathit{Diff}(I, N)$
b. $y_i \notin \mathit{Diff}(I, M)$

which is in contradiction with the hypothesis $\mathit{Diff}(I, N) \subset \mathit{Diff}(I, M)$.

This part of the proof can be used to prove also that $y_i \notin \mathit{Diff}(I, N)$ together with $y_i \in \mathit{Diff}(I, M)$ is impossible.

As a result, there is no model N of P such that $\mathit{Diff}(I, N) \subset \mathit{Diff}(I, M)$, thus $M \models T *_W P$.

Suppose Π unsatisfiable. As said above, M can be a model of $T *_W P$ only if there is a model I of T such that $I \cap C = \{c_i \mid \gamma_i \in \Pi\} = C_1$, and M is one of its closest models.

Consider a generic model $I \in \mathit{Mod}(P)$.

$$I = C_1 \cup X_1 \cup Y_1$$

Since Π is unsatisfiable, it cannot be $Y_1 = \{y_i \mid x_i \notin X_1\}$. The interpretation I is a model of $\neg x_i \vee \neg y_i$, thus there exists an index i such that $y_i \notin X_1$ and $y_i \notin Y_1$. Now consider the model

$$N = M \backslash \{x_i, y_i\}$$

we have

$$\begin{aligned} \mathit{Diff}(I, M) &= X \backslash X_1 \cup Y \backslash Y_1 \\ \mathit{Diff}(I, N) &= X \backslash (X_1 \cup \{x_i\}) \cup Y \backslash (Y_1 \cup \{y_i\}) \end{aligned}$$

thus $\mathit{Diff}(I, N) \subset \mathit{Diff}(I, M)$: the interpretation M is not a model of $T *_W P$. The hardness of Borgida's revision is proved in a similar manner: setting $T' = T \cup \{w\}$ and $P' = P \wedge \neg w$, we have that $T' *_B P' = (T *_W P) \wedge \neg w$. As a result, Π is satisfiable if and only if $M \models T' *_B P'$. □

THEOREM 17. *Model checking of $*_{SSU}$, $*_{Web}$, and $*_{Wit}$ is in $\rightsquigarrow$P.*

PROOF. As proved by Liberatore and Schaerf [2000], all these revision operators have a polynomial model checking. Since P $\subset \rightsquigarrow$P, the result follows. □

B. COMPILABILITY OF QUERY ANSWERING

In this section we prove the compilability of the problem of query answering for the revisions defined. Many results of compilability of query answering follows from two simple lemmas.

LEMMA 4. *If the problem of model checking for a revision is in $\rightsquigarrow$NP, then the corresponding query answering is in $\rightsquigarrow$coNP.*

PROOF. Since MC is in $\rightsquigarrow$NP, we have

$$M \models T * P \quad \text{iff} \quad \langle f_1(\langle T, P\rangle), g(f_2(\langle T, P\rangle), M)\rangle \in S$$

where f_1 and f_2 are poly-size functions, g is polynomial and S is an NP language. We can prove that

$$T * P \models Q \quad \text{iff} \quad \langle f_1(\langle T, P\rangle), g'(f_2(\langle T, P\rangle), Q)\rangle \in R \tag{1}$$

where g' is a polynomial function and R a coNP language. This would prove that QA is in $\rightsquigarrow$coNP. Define

$$\begin{aligned} g'(a, Q) &= \langle a, Q\rangle \\ R &= \{\langle a, \langle b, Q\rangle\rangle \mid \forall M \,.\, M \models Q \text{ or } \langle a, g(b, M)\rangle \notin S\} \end{aligned}$$

The function g' is clearly polynomial. The language R is in coNP, since it involves a check of non-membership to the NP language S. We now prove the statement (1).

$$T * P \models Q \quad \text{iff} \quad \forall M \,.\, M \models Q \text{ or } M \not\models T * P$$

$$\begin{array}{ll} \text{iff} & \forall M \ . \ M \models Q \text{ or } \langle f_1(\langle T,P\rangle), g(f_2(\langle T,P\rangle),M)\rangle \notin S \\ \text{iff} & \langle f_1(\langle T,P\rangle), \langle f_2(\langle T,P\rangle),Q\rangle\rangle \in R \\ \text{iff} & \langle f_1(\langle T,P\rangle), g'(f_2(\langle T,P\rangle),Q\rangle) \in R \end{array}$$

This proves the claim. □

The second lemma allows the determination of hardness of QA, if the hardness of MC is known.

LEMMA 5. *If MC is* $\|\!\rightarrow$*NP-hard then QA is* $\|\!\rightarrow$coNP*-hard.*

PROOF. Since MC is $\|\!\rightarrow$NP-hard, one can reduce $*\mathsf{sat}$ to it with a $\|\!\rightarrow$ reduction, that is,

$$\Pi \text{ satisfiable} \quad \text{iff} \quad \langle f_1(||\Pi||), g(f_2(||\Pi||),\Pi)\rangle \in MC$$

where f_1, f_2 are poly-size functions, while g is polynomial. Now MC is actually

$$\langle\langle T,P\rangle,M\rangle \in \text{MC} \quad \text{iff} \quad M \models T * P$$

It can be reformulated as QA

$$\begin{array}{lll} M \models T * P & \text{iff} & T * P \not\models \neg \mathit{Form}(M) \\ & \text{iff} & \langle\langle T,P\rangle, \neg \mathit{Form}(M)\rangle \notin \text{QA} \end{array}$$

As a result,

$$\Pi \text{ is satisfiable} \quad \text{iff} \quad \langle f_1(||\Pi||), \neg \mathit{Form}(g(f_2(||\Pi||),\Pi))\rangle \notin \text{QA}$$

thus

$$\Pi \text{ is unsatisfiable} \quad \text{iff} \quad \langle f_1(||\Pi||), g'(f_2(||\Pi||),\Pi)\rangle \in \text{QA}$$

where $g'(a,b) = \neg \mathit{Form}(g(a,b))$ is clearly polynomial. As a result, the query answering problem for $*$ is $\|\!\rightarrow$coNP-hard. □

With a similar proof, one can conclude that if MC is $\|\!\rightarrow$coNP-hard, then QA is $\|\!\rightarrow$NP-hard.

THEOREM 18. *Query answering for GFUV is* $\|\!\rightarrow$coNP*-hard, in* $\rightsquigarrow$coNP.

PROOF. Membership follows from the fact that query answering is in $\rightsquigarrow$coNP, as proved by Eiter and Gottlob [1992].

We prove hardness by reduction from the problem of query answering in Satoh's revision. Namely, we will prove that, if $\mathit{Var}(Q) \subseteq X$, then

$$T *_S P \models Q \quad \text{iff} \quad T' *_{GFUV} P' \models Q$$

where $T' = \{w_1, \ldots, w_n\}$, and

$$P' = T[X/Y] \wedge P \wedge \bigwedge_{1 \leq i \leq n} w_i \rightarrow (x_i \equiv y_i)$$

where $W = \{w_i\}$ and $Y = \{y_i\}$ are sets of new variables, one-to-one with X.

We prove first that if $T *_S P \models Q$ then $T' *_{GFUV} P' \models Q$. Let $K \in T' *_{GFUV} P'$. We will prove that $K \models Q$.

By definition, there must be a $T'' \subseteq T'$ that is maximally consistent with P', and such that $K \in \mathit{Mod}(T'' \cup \{P'\})$. Let

$$I = \{x_i \mid y_i \in K\}$$
$$J = K \cap X$$

Clearly, I is a model of T and J is a model of P. Moreover, $x_i \in \mathit{Diff}(I, J)$ implies that y_i and x_i have different truth value in K, thus w_i is inconsistent with T''. As a result, $T'' \subseteq \{w_i \mid x_i \notin \mathit{Diff}(I, J)\}$. We prove that $\mathit{Diff}(I, J)$ is minimal. Suppose that it is not, and let I' and J' be two models of T and P such that $\mathit{Diff}(I', J') \subset \mathit{Diff}(I, J)$. Now, let

$$T''' = \{w_i \mid x_i \notin \mathit{Diff}(I', J')\}$$

We have that $T'' \subset T'''$. Furthermore, $\{y_i \mid x_i \in I'\} \cup J' \cup T'''$ is a model of $T''' \cup \{P\}$, thus contradicting the hypothesis of T'' being maximally consistent with P.

Now, we have proved that $\mathit{Diff}(I, J)$ is minimal. As a result, J has the property that there exists a $I \in \mathit{Mod}(T)$ such that $\mathit{Diff}(I, J)$ is minimal. Thus $J \in T *_S P$, and since $T *_S P \models Q$, we have $J \models Q$. Since Q contains only variables in X, and $J = K \cap X$, we have $K \models Q$. This proves that each model K of $T' *_{GFUV} P'$ is also a model of Q, thus proving that $T' *_{GFUV} P' \models Q$.

We prove now the converse, that is, if $T' *_{GFUV} P' \models Q$ then $T *_S P \models Q$. Let J be a model of $T *_S P$. We prove that J is also a model of Q.

By definition, there exists a model I of T such that $\mathit{Diff}(I, J)$ is minimal. Let

$$T'' = \{w_i \mid x_i \notin \mathit{Diff}(I, J)\}$$

This set is consistent with P', and is also maximal (this is a consequence of the fact that $\mathit{Diff}(I, J)$ is minimal). Moreover, $K = \{y_i \mid x_i \in I\} \cup J \cup T''$ is a model of $T'' \cup P$, thus is also a model of $T' *_{GFUV} P'$, and thus $K \models Q$. Since Q contains only variables in X, and $J = K \cap X$, we have $J \models Q$. □

THEOREM 19. *Query answering for $*_D$, $*_S$, $*_W$, $*_B$ is $\|\not\rightsquigarrow$coNP-hard, and is in $\rightsquigarrow$coNP.*

PROOF. The compilability of QA of all these revisions follows from the compilability of MC and the two previous lemmas. □

THEOREM 20. *Query answering for $*_F$ is $\|\not\rightsquigarrow$NP-hard and $\|\not\rightsquigarrow$coNP-hard, in $\rightsquigarrow\Pi_2^p$.*

PROOF. The claim can be proved from the compilability of MC, and a proof similar to the two lemmas at the beginning of this paragraph. □

THEOREM 21. *Query answering for $*_{SSU}$ and $*_{Web}$ is in $\rightsquigarrow$P.*

PROOF. First, compute Ω. Now, as proved by Cadoli, Donini, Liberatore, and Schaerf [1999], $T *_\Omega P$ is query equivalent to $\wedge T[S] \wedge P$, where $S = \{x_i/y_i \mid x_i \in \Omega\}$, and the y_i's are new variables appearing nowhere else. By definition

$$T *_\Omega P \models Q \quad \text{iff} \quad \wedge T[S] \wedge P \models Q$$

Now, $\wedge T[S] \wedge P$ and Q are Horn formulas. Thus, deciding if the former implies the latter is a polynomial task. □

THEOREM 22. *Query answering for $*_{Wit}$ is in $\rightsquigarrow$P.*

PROOF. By definition, there exists $T' \subseteq T$ such that $T *_{Wit} P = \wedge T' \wedge P$. As a result, $T *_{Wit} P \models Q$ if and only if $\wedge T' \wedge P \models Q$ and the latter is polynomial since the formulas involved are Horn. □

C. COMPILABILITY AND COMPACT REPRESENTATIONS

THEOREM 23. *(Theorem 7) If $*$ has a compact representation w.r.t. logical equivalence, then its MC is in $\rightsquigarrow$P and its QA is in $\rightsquigarrow$coNP.*

PROOF. By hypothesis, there is a formula T_1 which is logically equivalent to $T*P$, and has size polynomial w.r.t. $||T|| + ||P||$. The problem of model checking for $*$ is in $\rightsquigarrow$P, since $M \models T * P$ can be checked by verifying whether $\langle f(T,P), M\rangle \in S$, where

$$\begin{aligned} f(T,P) &= T_1 \\ S &= \{\langle T', M'\rangle \mid M' \text{ is a model of } T'\} \end{aligned}$$

By hypothesis T_1 has size polynomial, thus f is poly-size. Furthermore, S is a polynomial language. As a result, the problem of model checking is in $\rightsquigarrow$P.

The problem of query answering is in $\rightsquigarrow$coNP, since $T * P \models M$ is equivalent to $\langle f(T,P), Q\rangle \in R$, where f is as above, and R is

$$R = \{\langle T', Q'\rangle \mid T' \text{ implies } Q'\}$$

R is a coNP language, thus the query answering is in $\rightsquigarrow$coNP. □

THEOREM 24. *(Theorem 8) If $*$ has a compact representation w.r.t. model equivalence, then its MC is in $\rightsquigarrow$P, and its QA is in $\rightsquigarrow$coNP.*

PROOF. This theorem can be proved in the same manner of Theorem 7. □

THEOREM 25. *(Theorem 9) If $*$ has a compact representation w.r.t. query equivalence, then its MC is in $\rightsquigarrow$NP, and its QA is in $\rightsquigarrow$coNP.*

PROOF. By hypothesis, there is a formula T_1, whose size is polynomial w.r.t. $||T|| + ||P||$, such that $T * P \models Q$ if and only if $T_1 \models Q$, for each formula Q over a set of variables X. The difference w.r.t. the previous theorem is that T_1 is allowed to have variables not in X, while in the problem of model checking and query answering we are only interested in models and formulas over X.

Consider the query answering first. To check whether $T * P \models Q$, one can verify whether $\langle f(T,P), Q\rangle \in S$, where

$$\begin{aligned} f(T,P) &= T_1 \\ S &= \{\langle T', Q'\rangle \mid Q' \text{ is a formula over } X \text{ and } T' \models Q'\} \end{aligned}$$

Notice that no restriction is imposed on the variables of T' in the language S. Note also that f is poly-size and S is a coNP language. As a result, the query answering problem is in $\rightsquigarrow$coNP.

About the model checking: $M \models T * P$ if and only if $T * P \not\models \neg Form(\{M\})$, which is in $\rightsquigarrow$coNP. As a result, the model checking problem is $\rightsquigarrow$NP, since Cadoli, Donini, Liberatore, and Schaerf [1996] proved that co-$\rightsquigarrow$NP=$\rightsquigarrow$coNP. □

D. COMPACT REPRESENTATION OF REVISION

THEOREM 26. *There exists a compact representation w.r.t. both model and query equivalence for GFUV.*

PROOF. We show the existence of a compact representation w.r.t. model equivalence only: query equivalence is always implied by model equivalence.

Let $T = \{\gamma_1, \ldots, \gamma_m\}$ and $P = \delta_1 \wedge \cdots \wedge \delta_k$. One can determine whether $J \in Mod(P)$ is a model of $T *_{GFUV} P$ by determining $T' = \{\gamma_j \mid J \models \gamma_j\}$, then checking whether $T \cup \{\gamma_j, P\}$ is inconsistent for any $\gamma_j \in T \backslash T'$. This can be formalized as a circuit.

Let γ_j be a clause. Checking consistency of $T' \cup \{\gamma_j, P\}$ can be done in polynomial time. As a result, it can be translated to a satisfiability check of a set of Horn clauses. By Lemma 3, such check can in turns be expressed by a polynomial-size circuit. Let INC_j be the circuit whose output is 1 if and only if M does not satisfy γ_j and $T' \cup \{\gamma_j, P\}$ is inconsistent. The circuit representing the result of the revision is:

$$P \wedge \bigwedge_{1 \leq j \leq m} r_j \vee \text{INC}_j$$

The whole circuit has J as model if and only if the set of the clauses of T satisfied by J is a maximal consistent subset of $T \cup \{P\}$. This proves that there exists a compact representation w.r.t. model equivalence for GFUV. □

THEOREM 27. *There exists a compact representation of Winslett's and Borgida's revisions w.r.t. query equivalence.*

PROOF. Since $T *_B P$ is always equivalent either to $T \cup \{P\}$ or $T *_W P$, suffices to prove that Winslett's revision has a compact representation w.r.t. query equivalence.

A model $J \in Mod(P)$ is a model of $T *_W P$ if and only if there exists a model I of T such that for any $K \in Mod(P)$, it holds $Diff(I, K) \subseteq Diff(I, J) \Rightarrow K = J$. We can formalize this by the meta-formula

$$T[X/Y] \wedge P \wedge$$
$$\text{the Horn set } \Big[P[X/Z] \wedge Diff(Y, Z) \subseteq Diff(Y, X) \wedge Z \neq X\Big] \text{ is unsatisfiable}$$

The idea is to represent the Horn set above using a set $B \cup H$, and then adding $\neg\mathsf{sat}$ to the formula $T[X/Y] \wedge P$.

Now, the formula obtained as a model $J \cup I \subseteq X \cup Y$ if and only $T *_S P$ has J as a model. As a result, these formulas are query equivalent w.r.t. formulas that contains only letters in X. □

THEOREM 28. *There exists a compact representation of Satoh's revision w.r.t. query equivalence.*

PROOF. The proof is similar to that of Winslett's revision. A model $J \in Mod(P)$ is a model of $T *_S P$ if and only if there exists a model I of T such that for any other pair $I' \in Mod(T)$ and $J' \in Mod(P)$, if $Diff(I', J') \subseteq Diff(I, J)$ then $Diff(I', J') = Diff(I, J)$. This can be expressed as

$$\begin{array}{l} T[X/Y] \wedge P \wedge \\ \text{the Horn set } \Big[T[X/Z] \wedge P[X/W] \wedge Diff(Z,W) \subseteq Diff(Y,X) \\ \wedge Diff(Z,W) \neq Diff(Y,X)\Big] \text{ is unsatisfiable} \end{array}$$

We represent the Horn set with two arrays of variables $B \cup H$. The compact representation is given by

$$T[X/Y] \wedge P \wedge \mathsf{sat} \wedge \text{REPR}$$

where REPR is the formula that express the fact that $B \cup H$ represents the above Horn set. □

REFERENCES

ALCHOURRÓN, C. E., GÄRDENFORS, P., AND MAKINSON, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic 50*, 510–530.

BOPPANA, R. AND SIPSER, M. 1990. The complexity of finite functions. In J. VAN LEEUWEN Ed., *Handbook of Theoretical Computer Science*, Volume A, Chapter 14, pp. 757–804. Elsevier Science Publishers (North-Holland), Amsterdam.

BORGIDA, A. 1985. Language features for flexible handling of exceptions in information systems. *ACM Transactions on Database Systems 10*, 563–603.

CADOLI, M., DONINI, F. M., LIBERATORE, P., AND SCHAERF, M. 1996. Feasibility and unfeasibility of off-line processing. In *Proceedings of the Fourth Israeli Symposium on Theory of Computing and Systems (ISTCS'96)* (1996), pp. 100–109. IEEE Computer Society Press. URL = `ftp://ftp.dis.uniroma1.it/PUB/AI/papers/cado-etal-96.ps.gz`.

CADOLI, M., DONINI, F. M., LIBERATORE, P., AND SCHAERF, M. 1999. The size of a revised knowledge base. *Artificial Intelligence 115*, 1, 25–64.

CADOLI, M., DONINI, F. M., AND SCHAERF, M. 1996. Is intractability of non-monotonic reasoning a real drawback? *Artificial Intelligence 88*, 1–2, 215–251.

CADOLI, M., DONINI, F. M., SCHAERF, M., AND SILVESTRI, R. 1997. On compact representations of propositional circumscription. *Theoretical Computer Science 182*, 183–202.

DALAL, M. 1988. Investigations into a theory of knowledge base revision: Preliminary report. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI'88)* (1988), pp. 475–479.

DOWLING, W. P. AND GALLIER, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming 1*, 3, 267–284.

EITER, T. AND GOTTLOB, G. 1992. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence 57*, 227–270.

FAGIN, R., ULLMAN, J. D., AND VARDI, M. Y. 1983. On the semantics of updates in databases. In *Proceedings of the Second ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS'83)* (1983), pp. 352–365.

FORBUS, K. D. 1989. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)* (1989), pp. 1273–1278.

GÄRDENFORS, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books, MIT Press, Cambridge, MA.

Ginsberg, M. L. 1986. Conterfactuals. *Artificial Intelligence 30*, 35–79.

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

Halpern, J. Y. and Vardi, M. Y. 1991. Model checking vs. theorem proving: A manifesto. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR'91)* (1991). Also in Lifshitz V. *Artificial Intelligence and Mathematical Theory of Computation. Papers in Honor of John McCarthy*, Academic Press, San Diego, 1991.

Johnson, D. S. 1990. A catalog of complexity classes. In J. van Leeuwen Ed., *Handbook of Theoretical Computer Science*, Volume A, Chapter 2, pp. 67–161. Elsevier Science Publishers (North-Holland), Amsterdam.

Katsuno, H. and Mendelzon, A. O. 1991. Propositional knowledge base revision and minimal change. *Artificial Intelligence 52*, 263–294.

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

Liberatore, P. 1995. Compact representation of revision of Horn clauses. In X. Yao Ed., *Proceedings of the Eighth Australian Joint Artificial Intelligence Conference (AI'95)* (1995), pp. 347–354. World Scientific.

Liberatore, P. 1998. On the compilability of diagnosis, planning, reasoning about actions, belief revision, etc. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)* (1998), pp. 144–155.

Liberatore, P. and Schaerf, M. 2000. Belief revision and update: Complexity of model checking. *Journal of Computer and System Sciences*. To appear.

Satoh, K. 1988. Nonmonotonic reasoning by minimal belief revision. In *Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'88)* (1988), pp. 455–462.