REVISING PROPOSITIONAL HORN CLAUSE KNOWLEDGE BASES

Foto Afrati[‡]
NTU Athens
afrati@theseas.ntua.gr

Theodoros G. Kavalieros[‡]
NTU Athens
kavalier@theseas.ntua.gr

Dimitri Theodoratos[§]
Rutherford Appleton Laboratory, UK
dth@inf.rl.ac.uk

Abstract

We investigate the problem of revising a propositional knowledge base, viewed as a set of Horn clauses. We present algorithms that revise the knowledge base with a Horn clause according to the Set of Theories [Fagin and Ullman and Vardi] and the When In Doubt Through It Out approaches [Ginsberg]. Under the first approach the problem has shown to be intractable even when further restrictions are applied. We show that the algorithms run in polynomial time in the case where the number of "relevant" negative clauses and rules (definite clauses having at least one negative literal) in the knowledge base is bounded. Intermediate results about resolution deduction of literals from a set of Horn clauses are obtained, which are of interest in their own right.

Keywords: Knowledge bases, Horn clauses, Algorithms, Complexity, Revision

‡Research supported by a grant of the Greek General Secretariat of Research and Technology.
§Research carried out as part of the ERCIM Fellowship Programme and financed by the Commission of the European Communities

1 Introduction

The problem of modifying Knowledge Bases (KBs) is an important one in the domain of AI (belief revision) and of DBs (updating DBs) and has been the subject of considerable research activity recently.

Theory change has also attracted the interest of researchers at the philosophical logic level. Several kinds of modifications have been distinguished and attempts has been made to determine the properties that should characterize this change (see f.i. [1, 12]).

In general, the KB has to be modified in the view of some new knowledge. This new knowledge may concern a change of the world (update operation) or may be some new information about a static world described by the KB (revision operation) [19]. We call here this new knowledge modification. The modification problem is to define and compute the modified KB.

Many approaches have been proposed for defining changes to KBs in the AI and DB literature [10, 2, 25, 5, 24, 27, 11, a.o.]. All these approaches respect the principle of persistence of prior knowledge [5] or minimality of change [10]. This principle states that when replacing an old KB by a new one the change must be minimal (the new KB must be as close as possible to the initial one). We can classify these methods according to various criteria:

The primacy of the update: This criterion as a principle says that the update represents the user's most recent piece of knowledge about the real world. Thus no update is illegal and the updated theory must reflect this new knowledge.

The consistency of the modified KB when the initial KB and the modification are satisfiable.

The irrelevance of syntax: this principle states that the modified theory should not depend on the syntactic form of either the KB or the modification. The approaches that respect this principle (model based) refer only to the models of the theory. On the other hand, in the context of syntax based (or formula based) approaches a formula has either to be retracted as a whole or left in the KB.

Measure of closeness: Different measures of closeness have been adopted by different authors. In order to do so, most of them define a partial order on theories (in the syntax-based approach) or on models (in the model based approach) [18, 20].

The uniqueness of the resulting theory: Some approaches result in a unique KB. When this is not the case and there are many new theories that satisfy the requirements of the definition, the well accepted principle of fairness is adopted stating that one of them should not be arbitrarily chosen. The reason is that the result must reflect the ambiguity that arises [10].

Each of the definitions for modifying KBs can fit well in a particular application domain. Nevertheless, it is widely believed that there is no general purpose method, suitable for any application domain, that will do the right thing in any case [28, 8].

Even though there has been a plethora of proposals for semantics for modifying KBs, it is not clear how to translate a semantic definition into a procedure suitable for implementation. Algorithms have been provided for computing the modified theory in the context of different methods and in restricted cases of the problem [14, 5, 26, 28, 4, 6]. The computational complexity of every approach is an important problem. All methods are clearly intractable [13] in the general case even for propositional languages [28, 23, 8].

In the framework presented above, it seems important to investigate under which restrictions tractability can be obtained for the various approaches proposed and which are the frontiers between tractability and intractability. Relaxing the initial problem with restrictions in order to achieve tractability is a well known technique. This technique is applied in the modification problem in [8] for different modification methods. Another way of approaching intractable problems is by finding approximative solutions that guarantee lower and upper bounds.

In this paper we consider two syntax-based approaches: the set of theories approach [10, 9] and the When In Doubt Throw It Out (WIDTIO) [28] approach [14] in the context of propositional finite languages. Under the first approach the result of a revision is a set of theories. The second approach satisfies the principle of the uniqueness of the resulting theory and it was first introduced as an approximation solution to the set of theories approach providing a lower bound to the formulas deduced by the modified KB. Syntax-based approaches [10, 9, 14, 22, 23] have been very popular because of their conceptual simplicity.

The complexity of the implication problem of the modified KB, given KB and a modification, in the propositional case is addressed, in [8]. It is shown that tractability can be obtained in model-based methods, if combined restrictions are applied simultaneously. Namely that the KB and the update are Horn formulae (conjunctions of Horn clauses) and that the size of the update is bounded by a constant. Nevertheless the problem remains intractable, under these restrictions, in the syntax-based approaches we consider here. Furthermore the problem remains intractable, for the set of theories approach, even in the case where the KB is a set of Horn clauses and the update is a literal. In this paper however, we show that the modification can be done in polynomial time when the KB is a set of Horn clauses and the number of "relevant" negative clauses and rules in the knowledge base is bounded. A rule is here a clause having one positive and at least one negative literal. This result holds also in the case where the size of the modification formula is not bounded. To obtain our result we introduce the notion of the rule goal graph; we define relevant rules and negative clauses on this graph; they are roughly the rules and negative clauses that are involved in some deduction of the modification clause.

Propositional Horn clauses enjoy, of the very interesting property of having a polynomial satisfiability problem (see f.i. [7]). Restricting our set of sentences (seen as describing our world) to Horn clauses has proven, in many cases, to have resulted in lowering considerably the computational complexity. This fact has been recognised for some time now, and a lot of research activity has been devoted to this direction. The approach in [15, 21, 16] and others, is that of trying to approximate any Knowledge Base by a Horn knowledge base (i.e., one that includes solely Horn clauses). Moreover, we believe that Horn clauses represent a natural style for expressing real-life conditions. Many rule based systems are based on them. Logic programming is one indication of their popularity. They also occur in many fields of mathematics. Concerning the restriction on the number of relevant clauses it is a natural one that applies to a wide range of real world cases; imagine for instance a large KB where though, all deductions of the modification formula involve a "small" number of rules and negative clauses.

The rest of the paper is organized as follows. The next section contains definitions, notation and assumptions. In section 3 we prove some lemmata that allow us to specify particular sound and complete deductions of positive and negative literals. In section 4 we provide algorithms for revising KBs with positive literals, negative literals and clauses. We prove their soundness and completeness according to the semantics of the set of theories approach. We also compute their complexity and we show that they are polynomial. In section 5 we modify the results of the previous section to the WIDTIO approach. We prove again that this revision is polynomial as well. Section 6 contains some conclusive remarks and directions for further research.

2 Preliminaries

We consider a propositional language L with finite variable set V.

Every variable in V is an atom. A literal is an atom (positive literal) or a negated atom (negative literal). Small latin letters from the beginning of the alphabet $a, b, c, \ldots$ (subscribed or not) denote atoms and small greek letters $\alpha, \beta, \gamma, \ldots$ denote literals. A clause is a disjunction of literals. A clause is Horn if it contains at most one positive literal. The small latin letters $p, q, r, \ldots$ denote Horn clauses. A (Horn) clause is definite if it contains exactly one positive literal while it is negative if it contains no positive literal. A definite clause containing negative literals is also called a rule while a definite clause having no negative literal is clearly a positive literal. Let r be the rule $\neg a_1 \vee \neg a_2 \vee \ldots \vee \neg a_n \vee a$. We often write r as $a_1 a_2 \ldots a_n \rightarrow a$. According to this notation $head(r)$ is a. An one-literal clause is called unit clause. When a clause contains no literal, we call it empty clause and we denote it by $\Box$.

A (Horn clause) knowledge base is a finite set of Horn clauses. We use the letters KB and T (standing for theory) to denote a knowledge base. Given a knowledge base KB, $DCLAUSES$ is the set of its definite clauses while $NCLAUSES$ is the set of its negative clauses and conversely: $KB = DCLAUSES \cup NCLAUSES$.

Given a set $DCLAUSES$ of definite clauses, $RULES$ is the set of its rules and $ATOMS$ is the set of its atoms: $DCLAUSES = RULES \cup ATOMS$. As usual, a set of formulas is identified with the conjunction of its formulas.

An empty clause is always interpreted as false. KB logically implies p (written $KB \models p$) if satisfaction of KB entails satisfaction of p. KB is consistent if it is satisfiable. Otherwise it is inconsistent. A clause is valid if it is true by any assignment to its variables. In the following, unless stated otherwise, we consider that KB (or T) is consistent and that clauses are not valid. Also we consider that there are no duplicate literals within clauses.

A resolution (principle) is a rule stating that: if there are two clauses $a \vee p$ and $\neg a \vee q$, construct the clause $p \vee q$. The constructed clause is a resolvent of $a \vee p$ and $\neg a \vee q$ (parent clauses). A unit resolution is a resolution in which the resolvent is obtained by using at least one unit parent clause. Given T, we call its members input clauses. An input resolution is the one in which one of resolvent's parent clauses is an input clause. We use resolution to define deductions: a (resolution) deduction of p from T is a node labelled tree having each leaf labelled by a clause in T, each internal node labelled by the resolvent of the labels of its immediate predecessors and the root labelled by p. Clearly this is a binary tree. A deduction is denoted by the capital letter D. $leaves(D)$ is the set of the labels of the leaves of D. In what follows, we confound nodes with their labels. If there is a deduction of p from T we write $T \vdash p$. A (resolution) refutation of T is a deduction of $\Box$ from T. Unit and input deduction and refutation are defined analogously. The symbol $\vdash_{unit}$ (respectively $\vdash_{input}$) is used for unit (respectively input) deduction.

Capital and small letters cited above as notation are often used in what follows with subscripts or with primes. Minimality and maximality when referred is considered with respect to set inclusion. For a set S, $max(S)$ and $min(S)$ are the sets of maximal and minimal elements of S respectively: $max(S) = \{s \mid s \in S$ and there is no s' in S such that $s \subset s'\}$ and $min(S) = \{s \mid s \in S$ and there is no s' in S such that $s' \subset s\}$.

Resolution is sound. It is also well known that the resolution principle is refutation complete for a set of clauses [3]. This means that if a set of clauses T is unsatisfiable then there exists a resolution refutation of T. Unit resolution and input resolution are refutation equivalent. This means that if there is a unit refutation for a set of clauses T then there is also an input refutation and conversely [3]. Unit resolution (and so input resolution) is refutation complete for a set of horn clauses [17].

3 Deduction in Horn clause KBs

In the last years a lot of research has been done in the domain of theorem proving. Usually, the implication of a formula by a theory is checked by checking the satisfiability of a theory or equivalently by checking for the deduction of the empty clause from a theory. Clearly it is preferable to search for a refutation in a subset than in the set of all the possible deductions. Thus, specific kinds of deductions have been found that are refutation complete for subclasses of formulas.

When we modify a KB, we are interested in the deduction of a formula (the negation of the modification) and not of the empty clause. Moreover, we are interested in finding all the different collections of elements of the KB which are necessary for the deduction of this formula. Thus we need to consider its deductions. Fortunately, we can restrict our searching to any specific (kind of) deduction which is complete for the implication of this formula. The more specific this kind of deduction, the more we restrict the deductions we have to consider.

In this section we use previous results on complete refutations for Horn clauses in order to find a specific complete deduction for the implication of positive and negative literals from a set of Horn clauses. The lemmata presented here are also used in the proof of the soundness and the completeness of the algorithms revising KBs.

Lemma 3.1 *A refutation has exactly one leaf labelled by a negative clause.* $\Box$

Proof: The proof is straightforward if we observe that (a) the resolvent of two definite clauses is always a definite clause and (b) the resolvent of a definite and a negative clause is a negative clause or the empty clause. $\Box$

3.1 Deduction of positive literals

A positive unit deduction (refutation) is a unit deduction (refutation) where one of the parents of each internal node is labelled by a positive unit clause (positive literal). If there is a positive unit deduction of p from T we write $T \vdash_{p.unit} p$. The next lemma shows that positive unit deduction is refutation complete for a set of Horn clauses.

Lemma 3.2 [17] *If T is an inconsistent set of Horn clauses, then* $T \vdash_{p.unit} \Box$. $\Box$

Positive unit deduction is complete for the implication of atoms by a set of Horn clauses. This is shown by the following lemma.

Lemma 3.3 $T \models a$ *if and only if* $T \vdash_{p.unit} a$. $\Box$

Proof: The *if part* comes straightforward from the soundness of the resolution deduction [3].
The *only if part:* If $T \models a$ then $T \cup \{\neg a\}$ is inconsistent; thus $T \cup \{\neg a\} \vdash_{p.unit} \Box$ by lemma 3.2. Since T is consistent, a positive refutation of $T \cup \{\neg a\}$, has $\neg a$ as a leaf. By lemma 3.1 it has mandatory the following form:

$$\neg a \qquad D$$

$$\square$$

where D is a positive unit deduction of a from T. $\square$

The next lemma shows that negative clauses are irrelevant for the implication of atoms. In other words $KB \models a$ if and only if $DCLAUSES \models a$.

Lemma 3.4 *A deduction of an atom, has no leaf labelled by a negative clause.* $\square$

Proof: It suffices to observe that the resolvent of a negative and a definite clause is a negative clause or the empty clause. $\square$

It is evident now that for every atom a, logically implied by T, there is a deduction D of a from T of the following form (positive unit deduction):

$$
\begin{array}{rl}
a_1 a_2 \ldots a_n \rightarrow a & \qquad D_1 \\
a_2 \ldots a_n \rightarrow a & \qquad D_2 \\
a_3 \ldots a_n \rightarrow a & \\
 & \\
a_n \rightarrow a & \qquad D_n \\
a &
\end{array}
$$

where $n \in N_0$ and $D_1, \ldots, D_n$ are deductions of $a_1, \ldots, a_n$ from T, respectively, of the same form.

Remark. Clearly the order of resolution is unimportant in these deductions: if there is a deduction of the form above, we can construct a different deduction of the same atom having the same set of leaves by changing arbitrarily the order of resolution of a rule-leaf with the roots of D_i's. Thus we can consider without loss of generality that every rule is resolved with the atoms of his body, in the order they appear in it (as it is shown in the deduction above).

A further restriction on the deductions we need to consider is provided by the lemma below which disallows to two different definite clauses containing the same positive literal to appear as leaves in a deduction.

Lemma 3.5 *If $T \vdash_{p.unit} a$, then there is a positive unit deduction D of a from T, such that for every pair of distinct rules r_1, r_2 and every atom a' in leaves(D):*

(a) $head(r_1) \neq a'$ and
(b) $head(r_1) \neq head(r_2)$. $\Box$

Proof: Consider the deduction D shown above. Every rule-leaf in D determines a deduction (subtree) of his head from T. Suppose that there is an atom a' and a rule r in $leaves(D)$ such that $head(r) = a'$. We can replace in D every deduction subtree determined by a rule having head a' by a' and obtain a new positive unit deduction of a from T. Suppose now that there are two distinct rules in $leaves(D)$ having the same head a'. Let D' be a deduction subtree in D such that: (a) D' is determined by a rule having head a' and (b) D' is not a subtree of the deduction subtree determined by a rule headed by a'. We can replace in D every deduction subtree determined by a rule having a' as head by D' and obtain again a positive unit deduction of a from T. By repeatedly applying this procedure we end up with a deduction of a from T satisfying the conditions of the lemma. $\Box$

The size of the deductions is restricted by the next lemma which prevents the redundant deduction of the same atom.

Lemma 3.6 *If $T \vdash_{p.unit} a$, then there is a positive unit deduction D of a from T of the type described by the lemma 3.5, such that there are no two nodes on a path from a leaf to the root of D labelled by the same atom.* $\Box$

Proof: Consider a deduction D satisfying the conditions of the previous lemma. These conditions impose that if an atom is deduced more than once then this is the head of the same rule in $leaves(D)$. If there are two nodes on a path from a leaf to the root labelled by the same atom then the path is infinite since the one needs the deduction of the other and conversely, a contradiction. $\Box$

3.2 Deduction of negative literals

The next lemma shows that input deduction is refutation complete for a set of Horn clauses.

Lemma 3.7 [17] *If T is an inconsistent set of Horn clauses, then $T \vdash_{input} \Box$.*

Input deduction is complete for the implication of negative literals by a set of Horn clauses. This is shown by the following lemma.

Lemma 3.8 *$T \models \neg a$ if and only if $T \vdash_{input} \neg a$.* $\Box$

Proof: The *if part* comes straightforward from the soundness of the resolution deduction [3].
The *only if part:* If $T \models \neg a$ then $T \cup \{a\}$ is inconsistent; thus $T \cup \{a\} \vdash_{input} \Box$ by lemma 3.7. Since T is consistent, a refutation of $T \cup \{a\}$, has a as a leaf. Consider

an input refutation of $T \cup \{a\}$, D, and the closest to the root leaf labelled by a. Observe that no rule-leaf headed by a can appear between the root and the leaf labelled by a since otherwise we could remove all the leaves labelled by a from D and construct a refutation of T, a contradiction. Thus this leaf labelled by a is the resolved leaf that ultimately removes $\neg a$ from the internal nodes. Clearly by removing all the leaves labelled by a from D and resolving all the other leaves in the same order we obtain an input deduction of $\neg a$ from T. □

Furthermore every deduction of a negative literal has a particular property:

Lemma 3.9 *A deduction of a negative literal has exactly one leaf labelled by a negative clause.* □

Proof: See lemma 3.1. □

The following lemma allows us to restrict more the kind of deductions we can consider. Every negative literal logicaly implied by KB is deduced by resolving a negative clause with definite clauses from the KB.

Lemma 3.10 *If $T \vdash_{input} \neg a$, then there is an input deduction of $\neg a$ from T of the following form:*

nc_1 $\quad$ dc_1

nc_2

nc_n $\quad$ dc_n

$\neg a$

where dc_i's (nc_i's) are positive (negative) clauses and $n \in N_0$. □

Proof: Consider an input deduction D of $\neg a$ from T. By lemma 3.9 it has only one leaf labelled by a negative clause p. We can prove by induction on the distance of p from the firstly resolved leaves that an input deduction of $\neg a$ from T satisfying the conditions of the lemma can be constructed by D (i.e. the only negative clause leaf is in the beginning of the deduction). □

The presence of rules having the same head or of a rule having the same head as an atom or the repetition of atoms can be constraint in a deduction of a negative literal as the following lemma shows. This provides also a limit in the size of the deduction as no definite clause can appear in it more than once.

Lemma 3.11 *If $T \vdash_{input} \neg a$, then there is an input deduction of $\neg a$ from T satisfying the conditions of lemma 3.10 where no atom appears as a leaf or as a disjunct of a leaf more than once.* $\square$

Proof: Let $T \vdash_{input} \neg a$. Consider a deduction D of $\neg a$ from T satisfying the conditions of lemma 3.10. Suppose that there are two definite clauses $dc_i, dc_j, i < j$ that contain the same atom b (not negated). Then we can construct another deduction D' by taking the sub-deduction of nc_i from D and resolving nc_i with $dc_{i+1}, \ldots, dc_n$ succesively, if this is possible. This procedure does not add extra literals to the resolvent and the empty clause cannot be deduced from T (since T is consistent). Thus D' is a deduction of $\neg a$ from T having a strictly smaller number of leaves containg b non-negated. By repeatedly applying this procedure we clearly end up with a deduction of $\neg a$ from T satisfying the conditions of the lemma. $\square$

Finally lemma 3.12 determines the order of consideration of definite clauses in a deduction: first rules and then atoms.

Lemma 3.12 *If $T \vdash_{input} \neg a$, then there is an input deduction of $\neg a$ from T satisfying the conditions of lemma 3.11 of the following form:*

nc_1 $\quad r_1$

nc_m $\quad r_m$

nc_{m+1} $\quad a_1$

nc_{m+k} $\quad a_k$

$\neg a$

where nc_i's, r_i's and a_i's are negative clauses, rules and atoms respectively and $m, k \in N_0$. $\square$

Proof: It is easy for one to see that we can 'push' atoms down in the deduction. $\square$

Remark The order of consideration of atoms in a deduction of the previous lemma is insignificant.

4 Revision under the Set of Theories semantics

A syntax based approach for KB modification is given in [10]. We call it here the set-of-theories approach. It is also considered in [14, 9, 22, 23]. Modifying a KB under this approach can result in multiple KBs. Thus the principle of the uniqueness of the modified theory is violated. This method is shown to satisfy the postulates for revision [1] as they were specified for propositional finite KBs in [20]. We say that this modification is a revision (operation) and we use the operator $\circ_S$ to denote it.

Given a consistent theory T and a consistent formula p, the possible worlds for p in T is the set:

$$W(p,T) = \{T' \subseteq T : T' \not\models \neg p, T' \subset U \subseteq T \Rightarrow \neg p\}$$

Then the revision of T with respect to the set of theories semantics is:

$$T \circ_S p = \{T' \cup \{p\} : T' \in W(p,T)\}$$

The possible worlds for p in T is the family of maximal subsets of T that do not entail $\neg p$. In order to compute $T \circ_S p$ then, we can compute the complements of the possible worlds for p in T with respect to T. These are the minimal subsets that subtracted by T result in a set not implying $\neg p$. In the following we treat first the revision with literals (negative and positive). These results are extented to Horn clauses.

Let D be a deduction of a literal α from T of a kind specified in the previous section. If we remove a leaf of $leaves(D)$ from T, we are no more able to construct D. We can find all the different deductions of α from T and form the set of all the sets containing one leaf from every deduction. Clearly the minimal elements of this set is what we are looking for for performing the revision of T with α.

In order to handle the revision with a Horn clause p, we compute the minimal sets for every disjunct (literal) of p. The minimal sets for p are then choosed among them.

4.1 Revision with a negative literal

We present now an algorithm which revises a knowledge base with a negative literal. This algorithm takes a consistent KB and $\neg a$ as input and produces as output a familly of knowledge bases $\mathbf{KB}'$. $\mathbf{KB}'$ is the revision of KB with $\neg a$, with respect to the set of theories semantics.

The algorithm proceeds in a bottom-up way (forward chainer). It provides a systematic exhaustive enumeration of all positive unit deductions of atoms from KB of the type described by the lemma 3.6. This specific kind of deduction is proved complete for the deduction of atoms, in subsection 3.1. In a second stage the algorithm provides a systematic exhaustive enumeration of the minimal sets that subtracted from KB disable the deduction of a.

We start by computing the set $DED.PLITS$. In $DED.PLITS$ we store all the positive literals (atoms) that can be deduced by KB and all the definite clauses of KB that get involved in the deduction of each of them (remember that negative clauses are irrelevant for the implication of atoms). Formally, $DED.PLITS$ is a set of pairs. The first component of a pair is a deduced atom, say a. The second component is the set of leaves (definite clauses of KB) of a deduction of a from KB. We use the variables $PRE.PLITS$ and $CUR.PLITS$ to store intermediate computed pairs. $PRE.PLITS$ keeps the pairs formed in the previous iteration of the computation procedure while $CUR.PLITS$ keeps the pairs being formed in the current iteration.

Since we are interested in disabling KB from implying a, the algorithm selects from $DED.PLITS$ the sets of leaves of the deductions of a and stores them in $DED.OF.A$. It takes then one leaf from every set, in all the different ways and forms the collection of sets $DIS.DED.A$. Every set of this collection subtracted from KB disables the deduction of a. The minimal elements of $DIS.DED.A$ with respect to set inclusion are stored in $DIS.DED.A_{min}$. The output set $\mathbf{KB'}$ is provided by succesively subtracting every element of $DIS.DED.A_{min}$ by KB and adding $\neg a$.

Algorithm UPNEGLIT

Input: $KB, \neg a$ $\{KB$ is $RULES, ATOMS$ and $NCLAUSES\}$
Output: $\mathbf{KB'}$

begin

$\{$*initialisation of variables*$\}$

$DED.PLITS := \{(a, \{a\}) \mid a \in ATOMS\}$;
$PRE.PLITS := DED.PLITS$;

$\{$*computation of the deductions of a* $\}$
repeat

$\{$*for every rule in KB we check if it can be used to produce a new deduction of an atom from KB*$\}$

CUR.PLITS := $\emptyset$;
for every rule $a_1 \dots a_n \rightarrow a$ in $RULES$ do
for every collection $(a_1, S_1), \dots, (a_n, S_n)$ from $DED.PLITS$ s.t.
some $(a_i, S_i) \in PRE.PLITS$ and

there are no distinct rules r_1, r_2 or a rule r_1 and an atom a'
in $\{a_1 \ldots a_n \to a\} \cup (\bigcup_{j=1}^{n} S_j)$ s.t.
$head(r_1) = head(r_2)$ or $head(r_1) = a'$ do

{*a new deduction of the head of the rule can be constructed. Its leaves will form the second component of a new pair in DED.PLITS*}

$\mathcal{S} := \{a_1 \ldots a_n \to a\} \cup (\bigcup_{j=1}^{n} S_j)$;
$CUR.PLITS := CUR.PLITS \cup \{(a, \mathcal{S})\}$
endfor
endfor;
$PRE.PLITS := CUR.PLITS$;
$DED.PLITS := DED.PLITS \cup CUR.PLITS$;
until $CUR.PLITS := \emptyset$;

{*We compute now the revised base* $\mathbf{KB}'$ *from the deductions in* $DED.PLITS$}

$DED.OF.A := \{S \mid (a, S) \in DED.PLITS\}$;
$DIS.DED.A := \{\ S \mid S = \{p_1\} \cup \ldots \cup \{p_k\}, DED.OF.A = \{S_1, \ldots, S_k\}$
and $p_i \in S_i, i = 1, \ldots, k\}$;
$DIS.DED.A_{min} := min(DIS.DED.A)$;
$\mathbf{KB}' := \{KB' \mid KB' = (KB - S) \cup \{\neg a\},\ S \in DIS.DED.A_{min}\}$
end. □

Example 4.1 Let KB be a consistent KB where $ATOMS = \{e, f, b\}$ and $RULES = \{b \to a,\ de \to b,\ f \to d\}$. Suppose we want to revise KB with $\neg b$. We will follow the steps of our algorithm. First the set $DED.PLITS$ is constructed as follows:

Initialization
$DED.PLITS = \{(e, \{e\})\} \cup \{(f, \{f\})\} \cup \{(b, \{b\})\}$.
$PRE.PLITS = DED.PLITS$

First repeat-until Iteration
The rule $b \to a$ can be activated:
$\mathcal{S} = \{b \to a\} \cup \{b\}, \quad CUR.PLITS = \{(a, \{b \to a, b\})\}$.
The rule $f \to d$ can be activated:
$\mathcal{S} = \{f \to d\} \cup \{f\}, \quad CUR.PLITS = \{(a, \{b \to a, b\})\} \cup \{(d, \{f \to d, f\})\}$.
Thus at the end of this iteration:
$CUR.PLITS = \{\ (a, \{b \to a, b\}),\ (d, \{f \to d, f\})\}$,
$DED.PLITS = \{\ (e, \{e\}),\ (f, \{f\}),\ (b, \{b\}),\ (a, \{b \to a, b\}),$
$(d, \{f \to d, f\})\ \}$.

Second repeat-until Iteration
The rule $de \to b$ can be activated:
$\mathcal{S} = \{de \to b\} \cup \{e\} \cup \{f \to d, f\}$,
$CUR.PLITS = \{(b, \{de \to b, e, f \to d, f\})\}$.
Thus at the end of this iteration:
$PRE.PLITS = \{(b, \{de \to b, e, f \to d, f\})\}$.
$DED.PLITS = \{\ (e, \{e\}),\ (f, \{f\}),\ (b, \{b\}),\ (a, \{b \to a, b\}),$
$(d, \{f \to d, f\}),\ (b, \{de \to b, e, f \to d, f\})\ \}$.

Third repeat-until Iteration
The rule $b \to a$ can be activated again:
$\mathcal{S} = \{b \to a\} \cup \{de \to b, e, f \to d, f\}$,
$CUR.PLITS = \{(a, \{b \to a, de \to b, e, f \to d, f\})\}$.
Thus at the end of this iteration:
$PRE.PLITS = \{(a, \{b \to a, de \to b, e, f \to d, f\})\}$
$DED.PLITS = \{\ (e, \{e\}),\ (f, \{f\}),\ (b, \{b\}),\ (a, \{b \to a, b\}),$
$(d, \{f \to d, f\}), (b, \{de \to b, e, f \to d, f\}), (a, \{b \to a, de \to b, e, f \to d, f\})\}$.

Fourth repeat-until Iteration
No rule can be activated. Thus at the end of this iteration $CUR.PLITS = \emptyset$ and the function returns the last value of $DED.PLITS$.

The algorithm proceeds now as follows:
$ALL.DED = \{\ (e, \{e\}),\ (f, \{f\}),\ (b, \{b\}),\ (a, \{b \to a, b\}),\ (d, \{f \to d, f\}),$
$(b, \{de \to b, e, f \to d, f\}),\ (a, \{b \to a, de \to b, e, f \to d, f\})\}$.
Since we are interested in the deductions of b:
$DED.OF.A = \{\ \{b\},\ \{de \to b, e, f \to d, f\}\ \}$.
Thus we can have four possible sets that subtracted by KB will disable the deduction of b from KB:
$DIS.DED.A = \{\ \{b, de \to b\},\ \{b, e\},\ \{b, f \to d\},\ \{b, f\}\}$.
We are looking for the minimal sets among them. Since none of those sets is subset of another, $DIS.DED.A_{min} = DIS.DED.A$. Thus we have four knowledge bases as a result of the revision of KB with $\neg b$, stored in $\mathbf{KB'} = \{KB_1, KB_2, KB_3, KB_4\}$.
$KB_1 = \{e, f, b \to a, f \to d\} \cup \{\neg b\}$
$KB_2 = \{f, b \to a, de \to b, f \to d\} \cup \{\neg b\}$.
$KB_3 = \{e, f, b \to a, de \to b\} \cup \{\neg b\}$.
$KB_4 = \{e, b \to a, de \to b, f \to d\} \cup \{\neg b\}$. □

Theorem 4.1 *Algorithm UPNEGLIT correctly computes the revision of a consistent knowledge base KB with a negative literal $\neg a$, with respect to the set of* *theories semantics, when KB and $\neg a$ are given as input.* $\square$

Proof: Algorithm $UPNEGLIT$ serially produces deductions of atoms from KB, exactly those satisfying the conditions of lemma 3.6. By lemmata 3.3 and 3.6, these deductions are sound and complete for the implication of atoms. Since at least one pair (a_i, S_i) in the construction of a new deduction is chosen from the pairs computed in the previous iteration over $RULES$, (PRE.PLITS), the same deduction is not encountered more than once. Lemma 3.5 subsumes lemma 3.6. Thus the condition on the non-existence of two distinct definit clauses with the same postive literal in the leaves of a derivation guarantees that the same atom will not label a node on a path from a leaf to the root more than once in the deductions considered. This guarantees also that the algorithm will terminate. Further, the algorithm for each deduction of a, stores its set of leaves. It is then easy to see that in the next steps it computes exactly $KB \circ_S \neg a$ as it is defined in the beginning of the current section. $\square$

Given KB we define the rule-goal graph G_{KB} as follows: The set of nodes of G_{KB} correspond one-to-one to the union of two sets: (a) the set of variables V of our language and (b) the set $RULES$ of the rules of KB . There is an arc (a, r) if and only if the head of rule r identifies with a and there is an arc (r, b) if and only if b identifies with an atom in the body of r (subgoal of rule r). Now given an atom a, a rule is relevant to a if it is reachable by a in G_{KB}. Clearly we can decide in polynomial time the number of rules relevant to an atom a. The following theorem can be shown:

Theorem 4.2 *Let KB be a consistent knowledge base and a be an atom. Suppose that the number of rules relevant to a in G_{KB} is bounded by a constant. Then the revision of KB with $\neg a$ can be done in polynomial time.* $\square$

Proof: Let $RULES_a$ be the set of rules relevant to a in G_{KB}. Clearly no other rules from $RULES$ are involved in any deduction of a from KB. We can then apply algorithm $UPNEGLIT$ on $ATOMS$ and $RULES_a$ instead of $ATOMS$ and $RULES$. Let n_a be the number of atoms in $ATOMS$. Suppose that the number of rules in $RULES_a$ is bounded by a constant c. Let $|r|$ be the greatest number of atoms in the body of a rule in $RULES_a$.
The repeat-until loop is repeated at most c times since no deduction can contain more than c distinct rules. $DED.PLITS$ can have at most $\mathcal{O}(c^c)$ elements. Every S_i has at most $n_a + c$ definite clauses (atoms and rules).
The construction of $DIS.DED.A$ is actually equivalent to computing cross-products. Since every deduction can have at most $(n_a + c)$ leaves and there are at most $\mathcal{O}(c^c)$ deductions the complexity of this step is at most $\mathcal{O}((n_a + c)^{c^c})$. The construction of the set of minimal elements costs at most $\mathcal{O}((n_a + c)^{c^{2c}}|r|^2)$. This determines also un upper bound for the complexity of the algorithm. $\square$

Algorithm $UPNEGLIT$ can be improved. First by refining the deductions we consider. We handle then a smaller class of complete deductions (for the implication of atoms) and so we have to construct fewer of them. Second by using appropriate implementation techniques and structures for the data. This study is beyond the scope of this paper, our main concern being here to highlight the key role of rules for the revision operation with negative literals.

4.2 Revision with a positive literal

We present in this subection an algorithm which revises a knowledge base with a positive literal a. The algorithm takes a consistent KB and a as input and produces as output a familly of knowledge bases $\mathbf{KB'}$. $\mathbf{KB'}$ is the revision of KB with a, with respect to the set of theories semantics.

This algorithm proceeds as the previous one in a bottom-up way (forward chainer). Practically, it provides in a first stage a systematic exhaustive enumeration of all the input deductions of $\neg a$ from KB of the type described by lemma 3.12. In a second stage it provides a systematic exhaustive enumeration of all the minimal sets of clauses that subtracted by KB invalidate all the deductions of $\neg a$. In this specific kind of deduction a negative clause is first resolved with rules and then with atoms. An atom cannot appear twice as a positive clause or as a disjunct of a definite clause in it. As it is shown in subsection 3.2, this deduction is complete for the implication of negative literals.

Information about the deductions is kept as in the previous algorithm in the form of pairs. The first component of a pair is the deduced negative clause say p. The second component is a set containing the negative clause (only one), the rules and the atoms from KB involved in the deduction of p. Pairs corresponding to negative clauses deduced using only rules (and an initial negative clause) are stored temporaly in $CUR.NCLAUSES$ and $PRE.NCLAUSES$. In $CUR.NCLAUSES$ are kept pairs being computed in the current iteration of the deductions construction procedure while in $PRE.NCLAUSES$ are kept pairs computed in the previous iteration. Once a negative clause is deduced using rules we try to resolve it with as many atoms as possible in any order. If $\neg a$ is deduced the second component of the corresponding pair (set of leaves of the deduction) is forwarded to the set $DED.OF.NOTA$. Thus $DED.OF.NOTA$ ends up by containing the sets of leaves of all the deductions of $\neg a$ from KB.

We want to disable KB from implying $\neg a$. We follow the same way as for the revision with negative clauses: the algorithm takes one leaf from every set of $DED.OF.NOTA$, in all the different ways and forms the collection of sets $DIS.DED.NOTA$. Every set of this collection subtracted from KB invalidates the deduction of $\neg a$. The minimal elements of $DIS.DED.NOTA$ with respect to set inclusion are stored in $DIS.DED.NOTA_{min}$. The output set $\mathbf{KB'}$ is provided by succesively subtracting every element of $DIS.DED.NOTA_{min}$ by KB and adding a.

Algorithm UPPOSLIT

Input: KB, a $\{KB$ is $RULES, ATOMS$ and $NCLAUSES\}$
Output: $\mathbf{KB'}$

```
begin

{we compute all the deductions of ¬a}

  DED.OF.NOTA := ∅;
  for every q in NCLAUSES do
    CUR.NCLAUSES := {(q, {q})};
    PRE.NCLAUSES := CUR.NCLAUSES − {(¬a, {¬a})};
    repeat

      {we resolve a deduced negative clause with
      as many atoms as possible}

      for every (p, P) in CUR.NCLAUSES do
        (p', P') := (p, P);
        for every a in ATOMS do
          if p' and a can be resolved together then
            p' becomes the resolvent of p' and a;
            P' := P' ∪ {a};
          endif;
        endfor;
        if p' is ¬a then
          DED.OF.NOTA := DED.OF.NOTA ∪ {P'}
        endif;
      endfor;

      {we resolve a deduced negative clause with rules
      in all the different ways}

      CUR.NCLAUSES = ∅;
      for every (p, P) in PRE.NCLAUSES do
        for every r in RULES do
          if p and r can be resolved together and
          there is no rule r' in P s.t.
          head(r) = head(r') then
```

construct the resolvent p' of p and r;
$P' := P \cup \{r\}$;
$CUR.NCLAUSES := CUR.NCLAUSES \cup \{(p', P')\}$;
endif;
endfor;
endfor;
$PRE.NCLAUSES :=$
$CUR.NCLAUSES - \{(\neg a, S) \in CUR.NCLAUSES\}$;
until $CUR.NCLAUSES = \emptyset$;
endfor;

{*We compute the revised knowledge base* $\mathbf{KB}'$}

$DIS.DED.NOTA := \{S \mid S = \{p_1\} \cup \ldots \cup \{p_k\}, DED.OF.NOTA = \{S_1, \ldots, S_k\}$ and $p_i \in S_i, i = 1, \ldots, k\}$;
$DIS.DED.NOTA_{min} := min(DIS.DED.NOTA)$;
$\mathbf{KB}' := \{KB' \mid KB' = (KB - S) \cup \{a\},\ S \in DIS.DED.NOTA_{min}\}$
end. $\square$

Example 4.2 Consider a KB where $RULES = \{c \rightarrow b, adf \rightarrow c\}$, $ATOMS = \{d, f\}$ and $NCLAUSES = \{\neg a \vee \neg d, \neg a \vee \neg b \vee \neg f\}$. Suppose we want to revise KB with a. We will follow the steps of our algorithm.

Computation of $DED.OF.NOTA$ *(the deductions of* $\neg a$*):*

First the negative clause $\neg a \vee \neg b$ *in* $NCLAUSES$ *is considered:*
After resolution with atoms from $ATOMS$:
$DED.OF.NOTA = \{\{\neg a \vee \neg d, d\}\}$
No rule can resolved with $\neg a \vee \neg b$.

Then the negative clause $\neg a \vee \neg b \vee \neg f$ *is considered.*
At the end of the first repeat-until iteration:
$CUR.NCLAUSES = \{(\neg a \vee \neg c \vee \neg f, \{\neg a \vee \neg b \vee \neg f, c \rightarrow b\})\}$
At the end of the second repeat-until iteration:
$CUR.NCLAUSES = \{(\neg a \vee \neg d \vee \neg f, \{\neg a \vee \neg b \vee \neg f, c \rightarrow b, adf \rightarrow c\,d)\}$
Now $\neg a$ can be deduced by resolving $\neg a \vee \neg d \vee \neg f$ with atoms. Thus: $DED.OF.NOTA = DED.OF.NOTA \cup \{\{\neg a \vee \neg b \vee \neg f, c \rightarrow b, adf \rightarrow c, d, f\}\}$
No rule can be applied at the third and last repeat-until iteration.
Thus $DED.OF.NOTA$ keeps its previous value.

There are no other negative clauses in $NCLAUSES$ *to consider.*
We have so $DED.OF.NOTA =$

$\{ \{\neg a \vee \neg d, d\},\ \{\neg a \vee \neg b \vee \neg f, c \rightarrow b, adf \rightarrow c, d, f\} \}$

Computation of the sets that subtracted from KB invalidate the deduction of $\neg a$ *and computation of the modified KB,* $\mathbf{KB'}$*:*

$DIS.DED.NOTA = \{ \{\neg a \vee \neg d, \neg a \vee \neg b \vee \neg f\},\ \{\neg a \vee \neg d, f\},$
$\{\neg a \vee \neg d, c \rightarrow b\},\ \{\neg a \vee \neg d, adf \rightarrow c\},\ \{\neg a \vee \neg d, d\},$
$\{d, \neg a \vee \neg b \vee \neg f\},\ \{d, f\},\ \{d, c \rightarrow b\},\ \{d, adf \rightarrow c\},\ \{d\}\}.$
We are looking for the minimal sets among them. Consequently:
$DIS.DED.A_{min} = \{ \{\neg a \vee \neg d, \neg a \vee \neg b \vee \neg f\},\ \{\neg a \vee \neg d, f\},$
$\{\neg a \vee \neg d, c \rightarrow b\},\ \{\neg a \vee \neg d, adf \rightarrow c\},\ \{d\}\}.$
Thus we have five knowledge bases as a result of the revision of KB with a, stored in $\mathbf{KB'} = \{KB_1, KB_2, KB_3, KB_4, KB_5\}$.
$KB_1 = \{f, d, c \rightarrow b, adf \rightarrow c\} \cup \{a\}$
$KB_2 = \{d, c \rightarrow b, adf \rightarrow c, \neg a \vee \neg b \vee \neg f\} \cup \{a\}$
$KB_3 = \{f, d, adf \rightarrow c, \neg a \vee \neg b \vee \neg f\} \cup \{a\}$
$KB_4 = \{f, d, c \rightarrow b, \neg a \vee \neg b \vee \neg f\} \cup \{a\}$
$KB_5 = \{f, c \rightarrow b, adf \rightarrow c, \neg a \vee \neg b \vee \neg f, \neg a \vee \neg d\} \cup \{a\}$. □

Theorem 4.3 *Algorithm UPPOSLIT correctly computes the revision of a consistent knowledge base* KB *with a positive literal* a*, with respect to the set of theories semantics, when* KB *and* a *are given as input.* □

Proof: Algorithm $UPNPOSLIT$ serially produces in a first stage deductions of $\neg a$ from KB at least all those satisfying the conditions of lemma 3.12. By lemmata 3.8 and 3.12, these deductions are sound and complete for the implication of negative literals. The condition on the non-existence of two rules in a deduction having the same head guarantees that the algorithm will terminate. For every deduction constructed the set of leaves is kept. It is then easy to see that in the next stage the algorithm computes exactly $KB \circ_S a$ as it is defined in the beginning of the current section. □

Given KB, let $RULES'$ be the set of rules constructed as follows: for every negative clause p in KB we put in $RULES'$ the rule $p \vee w$ where w is a variable not appearing in our variable set V. We define the rule goal graph G'_{KB} as follows: The set of nodes of G_{KB} correspond one-to-one to the union of two sets: (a) the set of variables appearing in $KB \cup \{w\}$, (b) $RULES \cup RULES'$. There is an arc (a, r) if and only if the head of rule r identifies with a and there is an arc (r, b) if and only if b identifies with an atom in the body of r (subgoal of rule r). Clearly rules in $RULES'$ correspond to the negative clauses of KB and have only outgoing arcs. Now, given a variable a, let $R(a)$ be the subset of rules in $RULES'$ which have the additional property that they reach in G'_{KB} the node labelled a. We say that a rule is w-relevant to a if it is reachable in G'_{KB} by $R(a)$ (rules in R(a) are also counted). We can decide in polynomial time the number of rules w-relevant to a. The following theorem can be shown:

Theorem 4.4 *Let KB be a consistent knowledge base and a be an atom. Suppose that the number of rules w-relevant to a in G'_{KB} is bounded by a constant. Then the revision of KB with a can be done in polynomial time.* □

Proof: Let $RULES_a$ be the w-relevant to a rules in KB and $NCLAUSES_a$ be the negative clauses in KB that correspond to the rules in $R(a)$. It is not difficult to see that no other rules and negative clauses from KB are involved in an input deduction of $\neg a$ from KB. We can then apply algorithm $UPPOSLIT$ on $ATOMS, RULES_a$ and $NCLAUSES_a$ instead of $ATOMS, RULES$ and $NCLAUSES$. Suppose that the number of clauses in $RULES_a$ and $NCLAUSES_a$ is bounded by a constant c. Let $l = max(|r|, |n|)$ where $|n|$ is the greatest number of literals in a negative clause in $NCLAUSES_a$.
The external for loop is repeated once for each negative clause in $NCLAUSES_a$. Since no deduction of a negative literal can have more rules than in $RULES_a$, the repeat-until loop is repeated at most $|RULES_a|$ times. $PRE.NCLAUSES$ can have at most c^c elements while $DED.OF.NOTA$ can have at most $\mathcal{O}(c^{c+1})$ elements.
The construction of $DIS.DED.NOTA$ is equivalent to computing cross-products. Every deduction can have at most $n_a + c$ leaves. There are at most $\mathcal{O}(c^{c+1})$ deductions. Thus the complexity of this step is at most $\mathcal{O}((n_a + c)^{c^{c+1}})$. The construction of the set of minimal elements costs at most $\mathcal{O}((n_a + c)^{c^{2c+2}} l^2)$. This provides also un upper bound for the complexity of the algorithm. □

Algorithm $UPPOSLIT$ can be improved. Different implementation improvements can also be performed. Again this is out of the scope of this paper and can be the subject of a further work.

4.3 Revision with a Horn clause

We combine in this subsection the methods for revision with literals in the revision with Horn clauses. Let $p = \alpha_1 \vee \ldots \vee \alpha_n$ be a Horn clause. (α_i's are literals.) $W(\alpha_i, T)$ are the possible worlds for α_i in T. We define $\mathcal{W} = \bigcup_{i=1}^{n} W(\alpha_i, T)$.

Lemma 4.1 *$W(p, T) = max(\mathcal{W})$.* □

Proof: We prove that $max(\mathcal{W}) \subseteq W(p, T)$:
It is not difficult to see for a clause q that if there exists $X \subseteq T$ such that $X \models \neg q$ then there exists $Y \in W(q, T)$ such that $X \subseteq Y$. Let now $X \in max(\mathcal{W})$. Then there exists $k \in [1, n]$ such that $X \in W(a_k, T)$ and X is a maximal element of W. Since $X \in W(\alpha_k, T)$, we have that $X \subseteq T, X \not\models \neg\alpha_k$ and

$\forall U : X \subset U \subseteq T \Rightarrow U \models \neg\alpha_k$. Thus $X \not\models \neg p$. Suppose that there exists U such that $X \subset U \subseteq T$ and $U \not\models \neg p$. i.e. $U \not\models \neg\alpha_1 \wedge \ldots \wedge \neg\alpha_n$. Then there exists $i \in [1, n], i \neq k$, such that $U \not\models \neg\alpha_i$. Therefore there exists $Y \in W(\alpha_i, T)$ such that $U \subseteq Y$ and so $X \subset Y$. Since $W(\alpha_i, T) \subseteq \mathcal{W}$, we have that $Y \in \mathcal{W}$. This is a contadiction since $max(\mathcal{W})$ contains only maximal elements of $\mathcal{W}$. Consequently $\forall U : X \subset U \subseteq T \Rightarrow U \models \neg p$. Thus $X \in W(p, T)$.
We prove that $W(p, T) \subseteq max(\mathcal{W})$:
Let $X \in W(p, T)$. Then $X \subseteq T, X \not\models \neg\alpha_1 \wedge \ldots \wedge \neg\alpha_n$ and $\forall U : X \subset U \subseteq T \Rightarrow U \models \neg\alpha_1 \wedge \ldots \wedge \neg\alpha_n$. Therefore there exists $k \in [1, n]$ such that $X \not\models \neg\alpha_k$ and $\forall U : X \subset U \subseteq T \Rightarrow U \models \neg\alpha_k$. Thus $X \in W(\alpha_k, T)$ and so $X \in \mathcal{W}$. By the definition of W(p,T), there is no $Y \in W(p, T)$ such that $X \subset Y$. Suppose now that $X \notin max(\mathcal{W})$. Since $X \in \mathcal{W}$, there exists $Y \in max(\mathcal{W})$ such that $X \subset Y$. Since $max(\mathcal{W}) \subseteq W(p, T)$ we have that $Y \in W(p, T)$, a contradiction. Thus $X \in max(\mathcal{W})$. □

The previous lemma suggests a straightforward way to compute the revision of a KB with a Horn clause. The algorithm we give below takes as input a consistent knowledge base KB and a Horn clause p and outputs the revision of KB with p, $\mathbf{KB'}$, with respect to the set of theories semantics.

This algorithm uses algorithms $UPNEGLIT$ and $UPPOSLIT$ to compute, for every literal α of the input clause, the complements of the worlds of α in T with respect to T (and possibly some supersets of them). It assembles these sets in $DIS.DED$. It selects then the minimal sets of $DIS.DED$. The output is provided by succesively subtracting these minimal sets from KB and adding the input clause p.

Algorithm UPCLAUSE

Input: $KB,\ p = \alpha_1 \vee \ldots \vee \alpha_n \quad \{KB \text{ is } RULES, ATOMS \text{ and } NCLAUSES\}$
Output: $\mathbf{KB'}$

begin
  $DIS.DED := \emptyset$;
  for $i = 1$ to n do
    use algorithm **UPNEGLIT** on KB, α_i if α_i is a negative literal to compute the set $DIS.DED.NOTA$ or **UPPOSLIT** on KB, α_i if α_i is a positive literal to compute the set $DIS.DED.A$;
    add the computed set to $DIS.DED$
  endfor;
  $DIS.DED_{min} := min(DIS.DED)$;
  $\mathbf{KB'} := \{KB' \mid KB' = (KB - S) \cup \{p\},\ S \in DIS.DED_{min}\}$
end. □

Example 4.3 Consider a KB, KB, having $ATOMS = \{d, e\}$, $RULES = \{d \rightarrow a, e \rightarrow a, d \rightarrow b\}$ and $NCLAUSES = \{\neg c \vee \neg e\}$. Suppose we want to revise KB with $ab \rightarrow c$. We apply algorithm $UPCLAUSE$:

$DIS.DED := \emptyset$
After applying algorithm $UPNEGLIT$ on $KB, \neg a$:
$DIS.DED = \{\{d, e\},\ \{d, e \rightarrow a\},\ \{d \rightarrow a, e\},\ \{d \rightarrow a, e \rightarrow a\}\}$.
After applying algorithm $UPNEGLIT$ on $KB, \neg b$:
$DIS.DED = DIS.DED \cup\ \{\{d\},\ \{d \rightarrow b\}\}$.
After applying algorithm $UPPOSLIT$ on KB, c:
$DIS.DED = DIS.DED \cup\ \{\{\neg c \vee \neg e\},\ \{e\}\}$.

Then $DIS.DED_{min} = \{\{d \rightarrow a, e \rightarrow a\}, \{d\}, \{d \rightarrow b\}, \{\neg c \vee \neg e\}, \{e\}\}$.
The output knowledge base is $\mathbf{KB'} = \{KB_1, KB_2, KB_3, KB_4, KB_5\}$.
$KB_1 = \{d, e, d \rightarrow b, \neg c \vee \neg e\} \cup \{ab \rightarrow c\}$
$KB_2 = \{e, d \rightarrow a, e \rightarrow a, d \rightarrow b, \neg c \vee \neg e\} \cup \{ab \rightarrow c\}$
$KB_3 = \{d, e, d \rightarrow a, e \rightarrow a, \neg c \vee \neg e\} \cup \{ab \rightarrow c\}$
$KB_4 = \{d, e, d \rightarrow a, e \rightarrow a, d \rightarrow b\} \cup \{ab \rightarrow c\}$
$KB_5 = \{d, d \rightarrow a, e \rightarrow a, d \rightarrow b, \neg c \vee \neg e\} \cup \{ab \rightarrow c\}$. □

Given KB let G'_{KB} be the rule-goal graph described ib section 4.2. We can show the following theorem:

Theorem 4.5 *Algorithm UPCLAUSE correctly computes the revision of a consistent knowledge base KB with a Horn clause p, with respect to the set of theories*

semantics, when KB and p are given as input.
Let $p = a \vee \neg a_1 \vee \ldots \vee \neg n$. Suppose that the number of rules relevant to a and the number of rules w-relevant to a_i, $i = 1, \ldots, n$ in G'_{KB} is bounded by a constant. Then the revision of KB with p can be done in polynomial time. $\square$

Proof: When algorithm $UPPOSLIT$ ($UPNEGLIT$) is executed on input KB, a ($\neg a_i$) the set $DIS.DED.NOTA$ ($DIS.DED.A$) contains the complements of the possible worlds for a ($\neg a_i$) in KB with respect to KB and possibly some supersets of them. Then algorithm $UPCLAUSE$ implements straightforward lemma 4.1 and so it computes exactly $KB \circ_S a$ as it is defined in the beginning of this section.

Now let h be the number of literals in the modification Horn clause and c be the bounding constant. The complexity of the construction of $DIS.DED$ is $\mathcal{O}((h-1)(n_a+c)^{c^c} + (n_a+c)^{c^{c+1}})$. Considering the second factor more important the complexity of computing the minimal elements is at most $\mathcal{O}((n_a+c)^{c^{2c+2}} l^2)$ which gives un upper bound for the complexity of the algorithm. $\square$

5 Revision under WIDTIO semantics

Another syntax-based approach, the When In Doubt Throw It Out (WIDTIO) [28] was first introduced in [14] as an approximation solution to the modification problem. It has the advantage that it results in a unique theory. We use here the operator $\circ_W$ to denote this modification operation.

Given a consistent theory T and a satisfiable formula p, the revision of T with respect to the WIDTIO semantics is defined as:

$$T \circ_W p = \{T' \cup \{p\} : \bigcap_{W \in W(p,T)} W\}$$

According to this definition and in order to perform a revision we need to compute the possible worlds of p in T and take their intersection. The following procedure takes as input a consistent knowledge base KB and a Horn clause p and outputs the revision of KB with p, KB', with respect to the WIDTIO semantics. It uses algorithm $UPCLAUSE$ to compute the minimal sets of clauses that subtracted from KB result in possible worlds of p in KB.

Algorithm WIDTIO.UPCLAUSE

Input: KB, p $\{KB$ is $RULES, ATOMS$ and $NCLAUSES\}$
Output: KB'

begin
use algorithm **UPCLAUSE** on KB, p to compute the set $DIS.DED_{min}$;

$W' := \bigcup_{S \in DIS.DED_{min}} S$;
$KB' := KB - S$
end. □

Example 5.1 Consider the KB KB of the previous example 4.3. Suppose that we want to perform the revision of KB with $b \rightarrow c$ uder the WIDTIO semantics. We apply algorithm $WIDTIO.UPCLAUSE$:
Algorithm $UPCLAUSE$ applied on KB, $b \rightarrow c$ computes the following value for the set $DIS.DED_{min}$:
$DIS.DED_{min} = \{\{d\}, \{d \rightarrow b\}, \{\neg c \vee \neg e\}, \{e\}\}$.
Then $W' = \{d, d \rightarrow b, \neg c \vee \neg e, e\}$.
Thus the output knowledge base is $KB' = \{d \rightarrow a, e \rightarrow a\}$. □

Theorem 5.1 *Algorithm WIDTIO.UPCLAUSE correctly computes the revision of a consistent knowledge base KB with a Horn clause p, with respect to the WIDTIO semantics, when KB and p are given as input. Let $p = a \vee a_1 \vee \ldots \vee n$. Suppose that the number of rules relevant to a and the number of rules w-relevant to a_i, $i = 1, \ldots, n$ in G'_{KB} is bounded by a constant. Then this revision of KB with p can be done in polynomial time.* □

Proof: When algorithm $WIDTIO.UPCLAUSE$ is executed on input KB, p , the set $DID.DED_{min}$ contains exactly the complements of the possible worlds of p in KB with respect to KB. Then Algorithm $WIDTIO.UPCLAUSE$ is a straightforward implementation of the definition of $KB \circ_W p$ above.
Un upper bound for the complexity of this procedure is given by the complexity of algorithm $UPCLAUSE$. □

6 Conclusion

We have investigated the computational complexity of the problem of revision on Horn clause Knowledge Bases, for the two syntactic approaches: a) revision under the set of theories semantincs defined by Fagin, Ullman and Vardi [10], and b) revision according to the When In Doubt Throw It Out (WIDTIO) [28] approach defined by Ginsberg [14]. We have shown the problem to be polynomial when the number of definite clauses having at least one negative literal (rules) in the Knowledge Base is bounded.

There are several interesting related upon complexity questions, primarily concerning the gap among our results and the result in [8]. It would be also interesting to see to what extend the results obtained here can be tranfered to restricted cases of first order Knowledge Bases.

References

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *J. Symbolic Logic*, 50:510–530, 1985.

[2] A. Borgida. Language features for flexible handling of exceptions in information systems. *ACM Trans. Database Syst.*, 10:563–603, 1985.

[3] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.

[4] T. S.-C. Chou and M. Winslett. Immortal: a model-based belief revision system. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 99 – 110, 1991.

[5] M. Dalal. Updates in propositional databases. Technical Report DCS-TR-222, Department of Computer Science, Rutgers Univ., Feb. 1988.

[6] A. Del Val. Computing knowledge base updates. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 740–750, 1992.

[7] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiablity of propositional Horn formulae. *J. Logic Programming*, 3:267–284, 1984.

[8] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell.*, 57:227–270, 1992.

[9] R. Fagin, G. M. Kuper, J. Ullman, and M. Vardi. Updating logical databases. In P. Kanellakis and F. Preparata, editors, *Advances in computing Research*, volume 3, pages 1–18. JAI Press, Greenwhich, CT, 1986.

[10] R. Fagin, J. Ullman, and M. Vardi. On the semantics of updates in databases. In *Proceedings Second ACM Symposium on Principles of Database Systems,* Atlanta, GA, pages 352–365, 1983.

[11] K. D. Forbus. Introducing actions into qualitative simulation. In *Proceedings IJCAI-89*, pages 1273–1278, 1989.

[12] P. Gärdenfors. *Knowledge in flux: Modeling the Dynamics of Epistemic states*. Bradford Books/MIT Press, Cambridge, MA, 1988.

[13] M. Garey and D. S. Jonhson. *Computers and intractability - A Guide to the Theory of NP-completness*. Freeman, New York, 1979.

[14] M. L. Ginsberg. Counterfactuals. *Artif. Intell.*, 30:35–79, 1986.

[15] H.A.Kautz and B.Selman. Knowledge compilation using horn approximations. In *Proceedings AAAI*, 1991.

[16] H.A.Kautz, M.J.Kearns, and B.Selman. Horn approximations of empirical data. Artificial Intelligence,to appear.

[17] L. Henschen and L. Wos. Unit Refutations and Horn Sets. *Journal of the ACM*, 21(4):590–605, 1974.

[18] H. Katsuno and A. O. Mendelson. A unified view of propositional knowledge base updates. In *Proceedings IJCAI,* Detroit, MI, pages 1413–1419, 1989.

[19] H. Katsuno and A. O. Mendelson. On the difference between updating a knowledge base and revising it. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 387–394, 1991.

[20] H. Katsuno and A. O. Mendelson. Propositional knowledge base revision and minimal change. *Artif. Intell.*, 52:263–294, 1991.

[21] D. Kavvadias, C. Papadimitriou, and M. Sideri. On horn envelopes and hypergraph transversals. In *ISAAC*, 1993.

[22] B. Nebel. A knowledge level analysis of belief revision. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning,* Toronto, Ontario, pages 301–311, 1989.

[23] B. Nebel. Belief revision and default reasoning: syntax based approaches. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 417–428, 1991.

[24] K. Satoh. Nonmonotonic reasoning by minimal belief revision. In *Proceedings International Conference on Fifth Generation Computer Systems*, pages 455–462, 1988.

[25] A. Weber. Updating propositional formulas. In *Proceedings First Conference on Expert Database Systems*, pages 487–500, 1986.

[26] M. Winslett Wilkins. A model-theoretic approach to updating logical databases. In *Proceedings Fifth ACM Symposium on Principles of Database Systems*, pages 224–234, Apr. 1986.

[27] M. Winslett Wilkins. Reasoning about action using a possible models approach. In *Proceedings AAAI-88,* St. Paul, MN, pages 89–93, 1988.

[28] M. Winslett Wilkins. *Updating logical databases.* Cambridge University Press, Cambridge, England, 1990.