Monotone Expansion of Updates in Logical Databases *

Michael Dekhtyar[1] , Alexander Dikovsky[2] , Sergey Dudakov[1] and Nicolas Spyratos[3]

[1] Dept. of CS, Tver State Univ. 33 Zheljabova str. Tver, Russia, 170000
Michael.Dekhtyar@tversu.ru, p000104@tversu.ru
[2] Universite de Nantes. IRIN, UPREF, EA No 2157. 2, rue de la Houssiniere BP 92208 F 44322 Nantes cedex 3 France, Alexandre.Dikovsky@irin.univ-nantes.fr
and
Keldysh Institute for Applied Math. 4 Miusskaya sq. Moscow, Russia, 125047
[3] Université de Paris-Sud, LRI, U.R.A. 410 du CNRS, Bât. 490
F-91405 Orsay Cedex, France, Nicolas.Spyratos@lri.fr

Abstract. To find a minimal real change after an update of a database with integrity constraints (IC) expressed by a generalized logic program with explicit negation is proven to be a Σ_2^p-complete problem. We define a class of operators expanding the input updates correctly with respect to the IC. The particular monotone expansion operator we describe is incrementally computed in square time. It provides a practical optimization of the standard complete choice algorithm resolving the update problem.

1 Introduction

Since the early 80ies the database updates have been in the focus of attention of the researchers and the practitioners. There are several mainstreams in this activity. One of the first approaches was that the updates are specified in some algorithmic algebra with iteration or recursion in its signature (cf. [1]). Its main interest is in the expressivity and the complexity analysis. Quite close is the other approach which proposes logical means for verification of the results of updates specified by expressions with operational semantics over primitive updates (see e.g. [3]). Most intensively was developed still another approach to updates which applies to knowledge bases (KB) presented by logic programs. Sometimes KB extensions (states) are distinguished from KB intensions (views), sometimes not. In the first case various methods were developed of updating views through abduction (see e.g. [9, 11, 8]), sometimes enforced by bottom-up hypothesis generation for integrity checking (e.g. [4]). In the second case either the models are updated (in order to indirectly change the knowledge base) (see e.g. [13, 14]), or the logic program specifying the KB is directly changed by another logic program (an update program, [2]). In the case where negative knowledge is treated, both approaches work exclusively with some sort of intended models: for instance, stable or well founded.

* This work was sponsored by the Russian Fundamental Studies Foundation (Grants 97-01-00973, 98-01-00204).

In this paper we develop the approach to updates proposed in [5, 6, 10]. We tackle the following problem: given an extended logic program Φ which formalizes the integrity constraints (IC), a correct initial state $I \models \Phi$, and an external update Δ which specifies the facts D^+ to be added to I and the facts D^- to be deleted from it, one should find the minimal real change $\Psi(I)$ of the state I, sufficient to accomplish Δ and to restore Φ if and when it is violated, i.e. to guarantee that $D^+ \subseteq \Psi(I), \Psi(I) \cap D^- = \emptyset$, *and* $\Psi(I) \models \Phi$. This setting changes substantially the requirements to the models in consideration. Indeed, the program Φ has nothing to do with knowledge definition. It only specifies the conflicts to avoid after the update. So the use of exclusively "intended" models would lead in this case to the loss of information or to unjustified conflict resolution failures. We illustrate this difference by the following simplified example.

Example 1 *The IC Φ below expresses a typical case of an exception from a general rule. It consists of two clauses. The first one expresses the general rule: "children (proposition* children*) can bathe (*bathe*) if with parents (*parents*)". The other one expresses an exception from this rule: "children cannot bathe while the ebb tide (proposition* ebb*) even when with parents":*

$$\begin{array}{l} \mathsf{bathe} \leftarrow \mathsf{children}, \mathsf{parents} \\ \neg\mathsf{bathe} \leftarrow \mathsf{children}, \mathsf{parents}, \mathsf{ebb}. \end{array}$$

*In the state $I = \{$*children,ebb$\}$ *suppose that the parents arrive, which is expressed as the addition of* parents *to I.*

The solution is simple but nontrivial. Inclusion of parents *causes replacement of* ebb *by* bathe*, the result being $I_1 = \{$*children, parents, bathe$\}$*. It seems that the "intended model" methods fail to find this solution. Indeed, since there are no rules with* parents *in the head, there is no direct refutation or abduction proof of the added fact. I is not a stable model of Φ, nor is it I_1. One could propose to add the clauses* children $\leftarrow$ *and* ebb $\leftarrow$ *to Φ and then to update the resulting program Φ' by the update program* {parents $\leftarrow$ }*. In this case I would become a stable model of Φ', but again the inertia rules of [2, 14] would prevent finishing the ebb tide (i.e. prevent to infer $\neg$ebb).*

Moreover, our problem of minimal real change with respect to IC is harder from the complexity point of view than the problem of intended knowledge update. The latter is of the type "guess and check" (which corresponds to NP) whereas, the former is proven in Section 3 of this paper to be Σ_2^p-complete. We present in this section a natural standard complete choice algorithm *D_search* which resolves this problem in linear space and exponential time.

In our earlier papers [5, 6] we introduce a broad class of update operators $\Psi(\Phi, \Delta, I)$ (we call them *conservative*), based on a mixed minimal change criterion which is a *combination of the maximal intersection, and of the minimal symmetric difference* of states. We consider both partial and total models, and we describe nondeterministic and deterministic conservative update algorithms based on conflict resolution techniques.

In this paper we deal with the case of classical total models and we propose a new practical method of speeding up the standard choice algorithm *D_search*. Our method is based on the simple idea that the initial update $\Delta = (D^+, D^-)$ can be incrementally and correctly expanded to a broader update after being iteratively propagated into the IC Φ. In fact, we define in Section 4 a broad class of such *update extension operators* which narrow the choice space of *D_search* and

simultaneously specialize and simplify the IC Φ with respect to the expanded updates. We show that these operators have elegant properties. The particular extension operators we consider propagate positive and negative elementary updates in both directions: from bodies to heads of the clauses and back. The combined fixpoint of the composition of these operators stabilizes after a finite number of steps and is computed in square time. In Section 5 we show how this operator can be used for both, static and dynamic optimizations of *D_search*. This practical method can be applied to databases with IC expressed in the form of logic programs without structures. It also applies to knowledge bases in which the knowledge has an invariable part.

2 Notation

We assume that the reader is familiar with the basic concepts and terminology of logic programming (see [12]).

Language. We fix a 1st order signature $\mathbf{S}$ with an infinite set of constants $\mathbf{C}$ and no other function symbols. A *domain* is a finite subset $\mathbf{D}$ of $\mathbf{C}$. For each domain $\mathbf{D}$ by $\mathbf{A}(\mathbf{D})$, $\mathbf{L}(\mathbf{D})$, $\mathbf{B}(\mathbf{D})$ and $\mathbf{LB}(\mathbf{D})$ we denote respectively the sets of all atoms, all literals, all ground atoms, and all ground literals in the signature $\mathbf{S}$ with constants in $\mathbf{D}$. A literal contrary to a literal l is denoted by $\neg.l$. We set $\neg.M = \{\neg.l \mid l \in M\}$.

Logic programs. We consider *generalized logic programs* in $\mathbf{S}$ *with explicit negation*, i.e. finite sets of clauses of the form $r = (l \leftarrow l_1, ..., l_n)$, where $n \geq 0$ and $l, l_i \in \mathbf{L}(\mathbf{D})$ (note that negative literals are possible in the bodies and in the heads of the clauses). For a clause r $head(r)$ denotes its *head*, and $body(r)$ its *body*. We will treat $body(r)$ as a set of literals. Integrity constraints (IC) are expressed by a logic program Φ of this kind. $\mathbf{IC}(\mathbf{D})$ denotes the set of all *ground* integrity constraints in the signature $\mathbf{S}$ with constants in $\mathbf{D}$. We consider the following *simplification order* on $\mathbf{IC}(\mathbf{D})$: $\Phi_1 \preceq \Phi_2$ if $\forall r \in \Phi_1 \; \exists r' \in \Phi_2 \; (head(r) = head(r') \;\&\; body(r) \subseteq body(r'))$.

Correct DB states. In this paper we consider classical total interpretations of ICs over *closed domains*. This means that a certain domain $\mathbf{D}$ is fixed for each problem. A (total) *interpretation* or a *DB state* I over domain $\mathbf{D}$ is a subset of $\mathbf{B}(\mathbf{D})$. Given an IC $\Phi \in \mathbf{IC}(\mathbf{D})$ and a DB state I over $\mathbf{D}$, a (ground) clause $r = (l \leftarrow l_1, ..., l_n)$ of Φ is *valid* in I (denoted $I \models r$) if $I \models l$ whenever $I \models l_i$ for each $1 \leq i \leq n$. $I \models a$ means $a \in I$, and $I \models \neg.a$ means $a \notin I$. I is *a correct DB state* or a *model* of Φ (denoted $I \models \Phi$) if all clauses of Φ are valid in I.

Updates. A pair $\Delta = (D^+, D^-)$, where D^+, D^- are subsets of $\mathbf{LB}(\mathbf{D})$, and $D^+ \cap D^- = \emptyset$, is called an *update*. Intuitively, the atoms of D^+ are to be added to DB state I, and those of D^- are to be removed from I. $\mathbf{UP}(\mathbf{D})$ will

denote the set of all updates in the signature $\mathbf{S}$ and with constants in $\mathbf{D}$. We say that $\Delta = (D^+, D^-)$ is *accomplished* in I if $D^+ \subseteq I$ and $D^- \cap I = \emptyset$. The updates in $\mathbf{UP}(\mathbf{D})$ will be partially ordered by the componentwise inclusion relation: $\Delta_1 \sqsubseteq \Delta_2$ iff $D_1^+ \subseteq D_2^+$ and $D_1^- \subseteq D_2^-$.

Update operators. Let Γ be an operator of the type $\Gamma : \mathbf{IC}(\mathbf{D}) \times \mathbf{UP}(\mathbf{D}) \to \mathbf{IC}(\mathbf{D}) \times \mathbf{UP}(\mathbf{D})$, $\Gamma(\Phi, \Delta) = (\Phi', \Delta')$, and $\Delta' = (D'^+, D'^-)$. In the definitions to follow Φ', Δ', D'^+, and D'^- are denoted respectively by $\Gamma(\Phi, \Delta)^{ic}$, $\Gamma(\Phi, \Delta)^{up}$, $\Gamma(\Phi, \Delta)^+$, and $\Gamma(\Phi, \Delta)^-$. We denote by Γ^n the n-fold composition of Γ and by Γ^ω the operator $\Gamma^\omega(\Phi, \Delta) = \lim_{n \to \infty} \Gamma^n(\Phi, \Delta)$.

In the sequel we will omit $\mathbf{D}$ when it causes no ambiguity. So when $\mathbf{D}$ is subsumed, in the place of $\mathbf{A}(\mathbf{D})$, $\mathbf{L}(\mathbf{D})$, $\mathbf{B}(\mathbf{D})$, $\mathbf{LB}(\mathbf{D})$ we will use the notation $\mathbf{A}$, $\mathbf{L}$, $\mathbf{B}$, $\mathbf{LB}$.

3 Conservative update operators and their complexity

In general an update may contradict a constraint. So a reasonable definition of an update operator should either contain a requirement of "compatibility" of an update and a constraint, or specify a part of the update "compatible" with the constraint. The requirement of compatibility is easy to formalize.

Definition 1 *For $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$ let us denote by $Acc(\Phi, \Delta)$ the set of all models $I \models \Phi$ where Δ is accomplished. An update Δ is* compatible *with an IC Φ if $Acc(\Phi, \Delta) \neq \emptyset$.*

Proposition 1 *The property $Comp = \{(\Phi, \Delta) |\ \Phi \in \mathbf{IC}\ \&\ \Delta \in \mathbf{UP}\ \&\ (\Delta$ is compatible with $\Phi)\}$ is NP-complete.*

Proof. It is evident that $Comp \in NP$. Now in order to prove NP-hardness of $Comp$ we show that $3{-}SAT \leq_p Comp$. Let $\alpha = \beta_1 \,\&\ldots\&\, \beta_m$ be some $3{-}CNF$, where $\beta_j = l_1^j \vee l_2^j \vee l_3^j$ and $l_i^j \in \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ for all $1 \leq j \leq m$ and $1 \leq i \leq 3$. We consider the set of atoms $\mathbf{B} = \{a, x_1, x_1', \ldots, x_n, x_n'\}$ and construct from α the pair $(\Phi_\alpha, \Delta_\alpha)$, where $\Delta_\alpha = (\{a\}, \emptyset)$ and Φ_α has $(2n + m)$ clauses:

$r_i = (\neg a \leftarrow x_i, x_i'), \quad i = 1, \ldots, n$

$p_i = (\neg a \leftarrow \neg x_i, \neg x_i'), \quad i = 1, \ldots, n$

$b_j = (\neg a \leftarrow \tilde{l}_1^j, \tilde{l}_2^j, \tilde{l}_3^j), \quad j = 1, \ldots, m,$

where $\tilde{l}_i^j = x_k$ if $l_i^j = x_k$, and $\tilde{l}_i^j = x_k'$ if $l_i^j = \neg x_k$.

Suppose that $(\Phi_\alpha, \Delta_\alpha) \in Comp$ and $I \in Acc(\Phi_\alpha, \Delta_\alpha)$. Then $a \in I$ and the clauses r_i and p_i assure that for each $1 \leq i \leq n$ either $x_i \in I$, or $x_i' \in I$, but not both. Let σ be the truth assignment $\sigma(x_i) = true$ iff $x_i' \in I$ and $\sigma(x_i) = false$ iff $x_i \in I$. Since each clause b_j, $1 \leq j \leq m$, is valid in I, there is some $1 \leq i \leq 3$ such that $I \not\models \tilde{l}_i^j$, and hence, $\tilde{l}_i^j \notin I$. So $\sigma(l_i^j) = true$ and therefore, $\sigma(\alpha) = true$.

Now suppose that $\alpha \in 3{-}SAT$ and let σ be a truth assignment such that $\sigma(\alpha) = true$. Consider the interpretation $I = \{x_i | \sigma(x_i) = false\} \cup \{x_i' | \sigma(x_i) =$

$true\} \cup \{a\}$. It is easy to check that $I \in Acc(\Phi_\alpha, \Delta_\alpha)$. $\square$

In [5] we propose the following minimal change criterion implementing the intention to keep as much initial facts as possible, and then to add possibly fewer new facts:

Definition 2 *Let I, I_1 be two DB states, and K be a class of DB states. We say that I_1 is* minimally deviating from I with respect to K *if* $\forall I_2 \in K\ (\neg(I \cap I_1 \subsetneq I \cap I_2)\ \&\ ((I \cap I_1 = I \cap I_2)\ \rightarrow\ \neg(I_2 \setminus I \subsetneq I_1 \setminus I)))$.

In terms of this minimal change criterion the *conservative update operators* we consider have been defined in [6] as follows.

Definition 3 *Let Δ be a given update which is compatible with IC Φ. An operator $\Psi = \Psi[\Phi, \Delta]$ on the set of DB states* **UP** *is a* conservative update operator *if for each DB state I :*

- *$\Psi(I)$ is a model of Φ,*
- *Δ is accomplished in $\Psi(I)$,*
- *$\Psi(I)$ is minimally deviating from I with respect to $Acc(\Phi, \Delta)$.*

3.1 Computational complexity of conservative updates

In order to measure the complexity of conservative updates we use two standard algorithmic problems: *Optimistic* and *Pessimistic Fall-Into-Problem* (**OFIP** and respectively **PFIP**) (cf. [7]).
OFIP: Given some $\Delta \in$ **UP** compatible with $\Phi \in$ **IC**, an initial state I, and a literal $l \in$ **LB**, one should check whether there exists a DB state I_1 such that:
(a) $I_1 \in Acc(\Phi, \Delta)$,
(b) I_1 is minimally deviating from I with respect to $Acc(\Phi, \Delta)$, and
(c) $I_1 \models l$.
PFIP: requires (c) be true **for all** models I_1 satisfying (a),(b).

We consider the combined complexity of these problems with respect to problem size evaluated as $N = |\mathbf{D}| + |I| + |\Delta| + |\Phi| + |l|$ ($|\ |$ being the size of constant or literal sets, and of programs in some standard encoding). We denote respectively by **OFIP** and **PFIP** the sets of all solutions (I, Δ, Φ, l) of these problems. It is easy to see that they are in fact co-sets (i.e. $(I, \Delta, \Phi, l) \in$ **PFIP** iff $(I, \Delta, \Phi, \neg.l) \notin$ **OFIP**). The following theorem classifies their complexity depending on nonmonotonic means used: negation in Φ and deletions in Δ.

Theorem 1 [1] *(1) If there are no negations in Φ and deletions in Δ (i.e. Φ is a definite logic program and $D^- = \emptyset$), then both* **OFIP** *and* **PFIP** *belong to P.*
(2) If Φ is a definite logic program, then **OFIP** *is NP-complete and* **PFIP** *is co-NP-complete.*
(3) In the general case **OFIP** *is Σ_2^p-complete and* **PFIP** *is Π_2^p-complete.*

[1] It is worth noting that these complexity bounds are similar to those established in [7] for updates of propositional KBs.

Proof. Point (1) being quite evident, we present some details of the proofs of (2) and (3).

Under conditions of (2), I_1 satisfies **OFIP** (b) iff (b'): I_1 equals the minimal model of $(\Phi \cup D^+ \cup (I_1 \cap I))$. So it is enough to guess I_1, then to check that it satisfies (a) and (b'), and finally, to check that $l \in I_1$. Therefore, **OFIP** $\in NP$. To show the lower bound, we reduce 3-CNF to **OFIP**. Let $\alpha = (l_{11} \vee l_{12} \vee l_{13}) \wedge \cdots \wedge (l_{n1} \vee l_{n2} \vee l_{n3})$ be a 3-CNF and $x_1, \ldots, x_m$ be the variables in α. Then we set $\Delta = (\{c\}, \{a\})$, $I = \{x_1, x'_1, \ldots, x_m, x'_m, a\}$, and define Φ as the set of clauses:

$a \leftarrow x_i, x'_i, \;\; 1 \leq i \leq m;$

$b_i \leftarrow c, \tilde{l}_{ij} \;\; , \; 1 \leq i \leq n, j = 1, 2, 3,$

$b \leftarrow b_1, \ldots, b_n,$

where $\tilde{l}_{ij} = x_i$, if $l_{ij} = x_i$, and $\tilde{l}_{ij} = x'_i$, if $l_{ij} = \neg x_i$.
One can verify that $(I, \Delta, \Phi, b) \in$ **OFIP** iff $\alpha \in$ 3-CNF.

For (3), the upper bound is provided by the following $\exists\forall$-algorithm:
1) Guess an interpretation I_1 such that $I_1 \models l$, and check that $I_1 \in Acc(\Phi, \Delta)$.
2) For each interpretation $I_2 \neq I_1$ check that either $I_2 \notin Acc(\Phi, \Delta)$, or the condition of definition 2 is satisfied.

In order to show the lower bound, we reduce to **OFIP** the Σ_2^p-complete set of valid quantified propositional formulas of the form $\exists x_1 ... \exists x_k \forall y_1 ... \forall y_m \alpha(\bar{x}, \bar{y})$, where α is in 3-DNF. Let ϕ be such a formula with $\alpha(\bar{x}, \bar{y}) = \bigvee_{j=1}^{r} (l_1^j \wedge l_2^j \wedge l_3^j)$, where $l_i^j \in \{x_1, \neg x_1, ..., x_k, \neg x_k, y_1, \neg y_1, ..., y_m, \neg y_m\}$ for al $1 \leq j \leq r$, $1 \leq i \leq 3$. Let $\mathbf{B} = \{a, b, x_1, x'_1, ..., x_k, x'_k, y_1, y'_1, ..., y_m, y'_m\}$. ϕ is encoded by the quadruple (Φ, Δ, I, b), where
$\Delta = (\{a\}, \emptyset)$, $I = \{x_1, x'_1, ..., x_k, x'_k\}$, and Φ has $2k + 4m + r$ clauses:

$rx_i : \; (\neg a \leftarrow x_i, x'_i); \;\; px_i : \; (\neg a \leftarrow \neg x_i, \neg x'_i), \;\; i = 1, ..., k;$

$ry_j : \; (b \leftarrow y_j, y'_j); \;\; py_j : \; (b \leftarrow \neg y_j, \neg y'_j), \;\; j = 1, ..., m;$

$t_j : \; (y_j \leftarrow b); \;\; t'_j : \; (y'_j \leftarrow b), \;\; j = 1, ..., m;$

$b_s : \; (b \leftarrow \tilde{l}_1^s, \tilde{l}_2^s, \tilde{l}_3^s), \;\; s = 1, ..., r, \;$ where $\tilde{l}_i^s = l_i^s, \;$ if $l_i^s \in \{x_1, ..., x_k, y_1, ..., y_m\}$, and $\tilde{l}_i^s = (\neg . l_i^s)', \;$ if $l_i^s \in \{\neg x_1, ..., \neg x_k, \neg y_1, \; ..., \neg y_m\}$.
Then the lower bound of point (3) follows from the next
Claim. ϕ *is valid iff* $(I, \Phi, \Delta, b) \in$ **OFIP**. $\square$

3.2 Directed search implementation of conservative updates

Since DB states and updates are finite, and the domain is closed, the space of DB states resulting from updates is finite as well. For each subset X of this space let us fix the topological order with respect to set inclusion on subsets of X, with the successor function $next_X$. We set $next_X(X) = \perp$ for some constant $\perp$. The following algorithm implements the complete search of a conservative update, directed by this order and conformable with our minimal change criterion.

Algorithm $D\_search(I, \Phi, \Delta)$
Input: a DB state I, and an update $\Delta = (D^+, D^-)$ compatible with $\Phi \in$ **IC**.
Local variables: $\tilde{I}$ (initial state update), H_{add} (positive one-step state update), H_{del}

(negative one-step state update), H^+ (search space for positive updates H_{add}), H^- (search space for negative updates H_{del}), b, c (boolean stabilization flags).
Output: I_1.
$\tilde{I} := (I \cup D^+) \setminus D^-$; % search space for the current state updates H_{del} and H_{add}:
$H^- := \tilde{I} \setminus D^+$; % search space for H_{del}
$H^+ := \mathbf{B} \setminus (\tilde{I} \cup D^-)$; % search space for H_{add}
$H_{del} := \emptyset$; $b := false$;
WHILE $\neg b$ **AND** $H_{del} \neq \ \perp$ **DO**
  $H_{add} := \emptyset$; $c := false$;
  WHILE $\neg c$ **AND** $H_{add} \neq \ \perp$ **DO**
    $I_1 := (\tilde{I} \setminus H_{del}) \cup H_{add}$;
    IF there is a clause $r \in \Phi$ such that $I_1 \not\models r$
    THEN $H_{add} := next_{H^+}(H_{add})$ **ELSE** $c := true$
    END_IF
  END_DO
  IF c **THEN** $b := c$ **ELSE** $H_{del} := next_{H^-}(H_{del})$
  END_IF
END_DO
Output I_1.

Theorem 2 *Algorithm D_search implements conservative update operators in linear space and in time* $O(2^{d+a}N)$, *where* $N = |\Phi| + |\Delta|$, d *is the size (the number of literals) of* H^-, *and* a *is the size of* H^+.

Proof. It is easy to check that for every $I' \in Acc(\Phi, \Delta)$ there are such $H'_{del} \subseteq H^-$ and $H'_{add} \subseteq H^+$ that $I' = (\tilde{I} \setminus H'_{del}) \cup H'_{add}$. Since $Acc(\Phi, \Delta) \neq \emptyset$ and algorithm $D{-}search(I, \Phi, \Delta)$ tries all pairs (H_{del}, H_{add}) such that $H_{del} \subseteq H^-$ and $H_{add} \subseteq H^+$, then it eventually stops and outputs some model $I_1 \in Acc(\Phi, \Delta)$. The order on pairs (H_{del}, H_{add}) induced by functions $next_{H^-}$ and $next_{H^+}$ insures that I_1 is minimally deviating from I with respect to $Acc(\Phi, \Delta)$. The complexity bounds are easy to verify. □

Theorem 1 shows that the nature of conservative update operators does not leave a hope to substantially optimize this algorithm in general. However, there may exist practical methods of speeding-up this standard search directed by our minimal change criterion. Below we propose one such efficient method. Our idea is to reduce the search space H^+, H^- by expanding correctly the sets D^+, D^-, propagating them into Φ.

4 Update expansion operators

In this section we define several operators on constraints and updates which correctly expand the sets of added and deleted facts according to the IC implemented by a logic program Φ, and simultaneously specialize and simplify Φ according to the expanded updates.

Each update $\Delta = (D^+, D^-)$ induces the set of literals l which it requires

$(\Delta \models l)$, and those to which it contradicts $(\Delta \not\models l)$. The idea behind our expansion operators is that this primary positive and negative information can be incrementally augmented being propagated into Φ. Table 1 below describes the primary relations $\models$ and $\not\models$ between Δ and ground atoms $a \in D^+$ and $a \in D^-$. These relations can be extended to conjunctions of ground literals $l_1, ..., l_k \in \mathbf{BL}$: $\Delta \models l_1, ..., l_k$ if $\forall j\ (\Delta \models l_j)$, and $\Delta \not\models l_1, ..., l_k$ if $\exists j\ (\Delta \not\models l_j)$. In particular, $\Delta \models \emptyset$, and $\Delta \not\models \emptyset$ is not true.

Table 1. Relations $\Delta \models l$ and $\Delta \not\models l$.

$\in$	D^+	D^-
a	$\Delta \models a,\ \Delta \not\models \neg a$	$\Delta \models \neg a,\ \Delta \not\models a$

These definitions allow to carry over the validity of literals from updates to the interpretations in which the updates are accomplished. Namely, the following evident property holds.

Lemma 1 *Let Δ be accomplished in I. Then:*
(1) if $\Delta \models l_1, ..., l_k$, then $I \models l_1, ...l_k$, and if $\Delta \not\models l_1, ..., l_k$, then $\neg(I \models l_1, ...l_k)$;
(2) if $I \models l_1, ...l_k$, then it is not the case that $\Delta \not\models l_1, ..., l_k$.

Evidently, relations $\models$ and $\not\models$ defined by Table 1 are monotone with respect to updates.

Lemma 2 *For any two updates $\Delta_1, \Delta_2 \in \mathbf{UP}$ such that $\Delta_1 \sqsubseteq \Delta_2$, and for any set $\{l_1, ..., l_k\} \subseteq \mathbf{BL}$*
(1) if $\Delta_1 \models l_1, ..., l_k$, then $\Delta_2 \models l_1, ...l_k$;
(2) if $\Delta_1 \not\models l_1, ..., l_k$, then $\Delta_2 \not\models l_1, ...l_k$.

The simplification order on programs we use conforms with the following residue operator simplifying logic programs via updates.

Definition 4 *The residue of $\Phi \in \mathbf{IC}$ with respect to Δ is defined as:*

$$res(\Phi, \Delta) = \{l \leftarrow \alpha \mid \exists r \in \Phi\ (head(r) = l\ \&\ \neg(\Delta \models l)\ \&\ \neg(\Delta \not\models body(r))\ \&\ \alpha \subseteq body(r)\ \&\ body(r) \setminus \alpha = max\{\beta \subseteq body(r) \mid \Delta \models \beta\})\}.$$

Example 2 *For the IC Φ consisting of clauses: $r_1 : a \leftarrow b, \neg c$, $r_2 : \neg b \leftarrow \neg a, d$, $r_3 : c \leftarrow \neg b, \neg d$, and $r_4 : \neg c \leftarrow b, d$, and the update $\Delta = (\{b\}, \{c\})$ their residue $res(\Phi, \Delta)$ has two clauses: $r_1' : a \leftarrow$ and $r_2' : \neg b \leftarrow \neg a, d$.*

The following evident properties of res ensure its incremental computation.

Lemma 3

(1) $Acc(\Phi, \Delta) = Acc(res(\Phi, \Delta), \Delta)$ for all $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$.

(2) Let an update $\Delta = (D^+, D^-)$ be partitioned into two updates $\Delta_1 = (D_1^+, D_1^-)$ and $\Delta_2 = (D_2^+, D_2^-)$, where $D^+ = D_1^+ \cup D_2^+$ and $D^- = D_1^- \cup D_2^-$. Then $res(\Phi, \Delta) = res(res(\Phi, \Delta_1), \Delta_2)$ for all $\Phi \in \mathbf{IC}$.

(3) Let $\Phi = \Phi_1 \cup \Phi_2$ be an IC partition. Then

$$res(\Phi, \Delta) = res(\Phi_1, \Delta) \cup res(\Phi_2, \Delta)$$

for all $\Delta \in \mathbf{UP}$.

(4) Let $|\Phi|$ be the size of $\Phi \in \mathbf{IC}$ (the number of literals in all clauses of Φ). Then $res(\Phi, \Delta) \preceq \Phi$, and therefore $|res(\Phi, \Delta)| \leq |\Phi|$ for all $\Delta \in \mathbf{UP}$.

We propose the following general definition of expansion operators.

Definition 5

(1) Let Φ, Φ' be ICs, and Δ, Δ' be updates. For the pairs (Φ, Δ) and (Φ', Δ') we say that they are update-equivalent *(notation: $(\Phi, \Delta) \equiv_u (\Phi', \Delta')$) if $Acc(\Phi, \Delta) = Acc(\Phi', \Delta')$.*

(2) An operator $\Gamma : \mathbf{IC} \times \mathbf{UP} \to \mathbf{IC} \times \mathbf{UP}$ is an update expansion operator *if for all compatible $\Phi \in \mathbf{IC}$ and $\Delta = (D^+, D^-) \in \mathbf{UP}$*

- $\Delta \sqsubseteq \Gamma(\Phi, \Delta)^{up}$.
- $\Gamma(\Phi, \Delta)^{ic} \preceq \Phi$,
- $(\Phi, \Delta) \equiv_u \Gamma(\Phi, \Delta)$.

From this definition it follows immediately that the set of all update expansion operators is closed under composition. The particular update expansion operators we propose are defined through operators which propagate the relations $\Delta \models l$ and $\Delta \not\models l$ in opposite directions: from bodies to heads and back.

Forward operator F on $\Phi \in \mathbf{IC}$ and $\Delta = (D^+, D^-) \in \mathbf{UP}$:
$F(\Phi, \Delta)^+ = D^+ \cup \{a \in \mathbf{B} \mid \exists r \in \Phi\ (a = head(r)\ \&\ \Delta \models body(r))\}$
$F(\Phi, \Delta)^- = D^- \cup \{a \in \mathbf{B} \mid \exists r \in \Phi\ (\neg a = head(r)\ \&\ \Delta \models body(r))\}$.

Backward operator B on $\Phi \in \mathbf{IC}$ and $\Delta = (D^+, D^-) \in \mathbf{UP}$:
$B(\Phi, \Delta)^+ = D^+ \cup \{a \in \mathbf{B} \mid \exists r \in \Phi\ (\Delta \not\models head(r)\ \&\ body(r) = \neg a\alpha\ \&\ \Delta \models \alpha)\}$
$B(\Phi, \Delta)^- = D^- \cup \{a \in \mathbf{B} \mid \exists r \in \Phi\ (\Delta \not\models head(r)\ \&\ body(r) = a\alpha\ \&\ \Delta \models \alpha)\}$.

Clearly, both operators F and B are monotone. They are also invariant with respect to the residue operator res and do not change models.

Lemma 4 *For any $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$ the following equalities hold:*
(1) $F(res(\Phi, \Delta), \Delta) = F(\Phi, \Delta)$, $B(res(\Phi, \Delta), \Delta) = B(\Phi, \Delta)$.
(2) $Acc(\Phi, \Delta) = Acc(\Phi, F(\Phi, \Delta))$, $Acc(\Phi, \Delta) = Acc(\Phi, B(\Phi, \Delta))$.

We now use these operators to define the forward and backward update expansions.

Forward update expansion Γ_f :

$$\begin{array}{l} \gamma_f^0(\Phi, \Delta) = (\Phi, \Delta) \\ \gamma_f(\Phi, \Delta) = (res(\Phi, \Delta), \\ \qquad\qquad F(res(\Phi, \Delta), \Delta)) \\ \gamma_f^{n+1}(\Phi, \Delta) = \gamma_f(\gamma_f^n(\Phi, \Delta)) \\ \Gamma_f(\Phi, \Delta) = \lim\limits_{n \to \infty} \gamma_f^n(\Phi, \Delta). \end{array}$$

Backward update expansion Γ_b :

$$\begin{array}{l} \gamma_b^0(\Phi, \Delta) = (\Phi, \Delta) \\ \gamma_b(\Phi, \Delta) = (res(\Phi, \Delta), \\ \qquad\qquad B(res(\Phi, \Delta), \Delta)) \\ \gamma_b^{n+1}(\Phi, \Delta) = \gamma_b(\gamma_b^n(\Phi, \Delta)) \\ \Gamma_b(\Phi, \Delta) = \lim\limits_{n \to \infty} \gamma_b^n(\Phi, \Delta). \end{array}$$

The operators F, B, and res we have introduced, have the properties that guarantee the existence of limits for the directed sets of their iterations.

Lemma 5 *For any pair $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$*
(1) there is $n \geq 0$ such that $\Gamma_f(\Phi, \Delta) = \gamma_f^n(\Phi, \Delta)$,
(2) there is $m \geq 0$ such that $\Gamma_b(\Phi, \Delta) = \gamma_b^m(\Phi, \Delta)$.

Proof. In order to establish point (1) let us consider for each pair (Φ, Δ), $\Delta = (D^+, D^-)$, the number $N(\Phi, \Delta) = |\Phi| + |\mathbf{LB}| - |D^+| - |D^-| \geq 0$. From Lemma 3 and the definition of γ_f it follows that for every n, if $\gamma_f^{n+1}(\Phi, \Delta) \neq \gamma_f^n(\Phi, \Delta)$, then $N(\gamma_f^{n+1}(\Phi, \Delta)) < N(\gamma_f^n(\Phi, \Delta))$. Therefore, for some $m \leq N(\Phi, \Delta)$ the equality $N(\gamma_f^{m+1}(\Phi, \Delta)) = N(\gamma_f^m(\Phi, \Delta))$ holds and $\Gamma_f(\Phi, \Delta) = \gamma_f^m(\Phi, \Delta)$.
A similar argument proves the point (2). □

Theorem 3
(1) Γ_f, Γ_b, and $\Gamma_{lim} = (\Gamma_f \circ \Gamma_b)^\omega$ are update expansion operators.
(2) There are algorithms f_expand and b_expand which correctly compute the operators Γ_f and Γ_b in linear time.
(3) Γ_{lim} is computable in square time.

Proof. 1. Let us show that Γ_f is an expansion operator. From Lemma 2 it follows that operators γ_f and γ_b are monotone with respect to updates. Then by standard induction on n we prove that so are operators γ_f^n and γ_b^n. Therefore, by Lemma 5 Γ_f and Γ_b are also monotone with respect to updates.

Now, in order to prove that Γ_f is an update expansion operator we show that

$$Acc(\Phi, \Delta) = Acc(\Gamma_f(\Phi, \Delta)).$$

In fact, it suffice to establish the equality

$$Acc(\Phi, \Delta) = Acc(\gamma_f(\Phi, \Delta)).$$

Then the previous equality will follow from Lemma 5 (2) by evident induction on m.
IF-part: $Acc(\Phi, \Delta) \subseteq Acc(\gamma_f(\Phi, \Delta))$. Suppose that $I \in Acc(\Phi, \Delta)$. By definition, $\gamma_f(\Phi, \Delta) = (res(\Phi, \Delta), F(res(\Phi, \Delta), \Delta))$.

First, we show that $I \models res(\Phi, \Delta)$. Let $r' = l \leftarrow \alpha$ be a clause in $res(\Phi, \Delta)$. Then by definition of res, there are a clause $r \in \Phi$ and a set of literals β such that $head(r) = l, body(r) = \alpha \cup \beta$, and $\Delta \models \beta$. By Lemma 1, $I \models \beta$. Now, if $I \models \alpha$, then $I \models body(r)$ and $I \models l$ (since $I \models r$). So $I \models r'$, and $I \models res(\Phi, \Delta)$.

Next, we show that the update $\gamma_f(\Phi, \Delta)^{up} = F(res(\Phi, \Delta), \Delta)$ is accomplished in I. Let a be any atom from $F(res(\Phi, \Delta), \Delta)^+$. Then either $a \in D^+$, or $a = head(r)$ for some clause $r \in res(\Phi, \Delta)$ such that $\Delta \models body(r)$. In the first case, $a \in I$ since Δ is accomplished in I. In the second case, by Lemma 1 $I \models body(r)$ and $I \models a$ since $I \models res(\Phi, \Delta)$. Therefore, $a \in I$ and $F(res(\Phi, \Delta), \Delta)^+ \subseteq I$. If $a \in F(res(\Phi, \Delta), \Delta)^-$, then either $a \in D^-$, or $\neg a = head(r)$ for some clause $r \in res(\Phi, \Delta)$ such that $\Delta \models body(r)$. Again we conclude that $I \models head(r) = \neg a$ and therefore, $a \notin I$. Then $I \cap F(res(\Phi, \Delta), \Delta)^- = \emptyset$. Hence, the update $\gamma_f(\Phi, \Delta)^{up}$ is accomplished in I.
ONLY-IF-part: $Acc(\gamma_f(\Phi, \Delta)) \subseteq Acc(\Phi, \Delta)$.
Suppose that $I \in Acc(\gamma_f(\Phi, \Delta))$. Since F is monotone with respect to updates, and $F(res(\Phi, \Delta), \Delta)$ is accomplished in I, then Δ is accomplished in I as well. Now let r be a clause of Φ with $head(r) = l$. Suppose that $I \models body(r)$. Then by Lemma 1(2), it is not the case that $\Delta \not\models body(r)$. If $\Delta \models l$, then by Lemma 1(1), $I \models l$ and $I \models r$. Otherwise, there is a clause $r' = l \leftarrow \alpha \in res(\Phi, \Delta)$ such that for some set of literals β $body(r) = \alpha \cup \beta$, and $\Delta \models \beta$. Since $I \models body(r)$, then $I \models \alpha$, and $I \models l$ because $I \models r'$. Therefore, $I \models r$. So $I \models \Phi$ and $I \in Acc(\Phi, \Delta)$.

This ends the proof that Γ_f is an update expansion operator.

2. Let us show that Γ_b is an expansion operator. A standard induction argument shows that it is enough to establish this fact for γ_b. So we should prove the equality

$$Acc(\Phi, \Delta) = Acc(\gamma_b(\Phi, \Delta)).$$

IF-part. $Acc(\Phi, \Delta) \subseteq Acc(\gamma_b(\Phi, \Delta))$.
Suppose that $I \in Acc(\Phi, \Delta)$. By definition, $\gamma_b(\Phi, \Delta) = (res(\Phi, \Delta), B(res(\Phi, \Delta), \Delta))$. Let us check that $\gamma_b(\Phi, \Delta)^{up} = B(res(\Phi, \Delta), \Delta)$ is accomplished in I.

Let $a \in B(res(\Phi, \Delta), \Delta)^+$. If $a \in D^+$ then $I \models a$. Let $a \notin D^+$. Since Δ is accomplished in I, there is a clause $r \in res(\Phi, \Delta)$ such that $r = l_0 \leftarrow \neg a\alpha, \Delta \not\models l_0$, and $\Delta \models \alpha$. Then by Lemma 1, $I \not\models l_0$, and $I \models \alpha$. Since $I \models r$ and $I \not\models l_0$, then $I \not\models body(r)$. Therefore, $I \not\models \neg a$ and $a \in I$. So $B(res(\Phi, \Delta), \Delta)^+ \subseteq I$.

Let $a \in B(res(\Phi, \Delta), \Delta)^-$. If $a \in D^-$, then $a \notin I$. Let $a \notin D^-$. Since Δ is accomplished in I, there is a clause $r \in res(\Phi, \Delta)$ such that $r = l_0 \leftarrow a\alpha, \Delta \not\models l_0$, and $\Delta \models \alpha$. Then by Lemma 1, $I \not\models l_0$ and $I \models \alpha$. Since $I \models r$ and $I \not\models l_0$, $I \not\models body(r)$ is true. Hence, $I \not\models a$, and $a \notin I$. So $B(res(\Phi, \Delta), \Delta)^- \cap I = \emptyset$ and therefore, $\gamma_b(\Phi, \Delta)^{up}$ is accomplished in I. Together with $I \models res(\Phi, \Delta)$, which was proven above, this implies $Acc(\Phi, \Delta) \subseteq Acc(\gamma_b(\Phi, \Delta))$.

The proof of the *ONLY-IF-part* is similar to that of the operator γ_f.

3. By a standard induction argument we infer that Γ_{lim} is an update expansion operator from points 1 and 2, and from Lemma 5.

We omit here quite a standard algorithm *f_expand* for Γ_f, and we show a sequential computation in linear time for Γ_b. It keeps track of the set BR of the clauses whose heads contradict the current update, and places into the queue Q those clauses in this set, whose bodies have only one literal (which therefore, also contradicts Δ). In the course of the main loop one clause in Q is invoked and the literal in it's body contradicting to Δ is used to expand the current update (M^+, M^-) and to reduce the current IC Φ_1.

Algorithm *b_expand* **computing the backward update expansion**
Input: Some compatible update $\Delta = (D^+, D^-)$ and $\Phi \in$ **IC**.
Local variables: BR (clauses whose heads contradict the current update expansion), Φ_1 (the resulting IC residue incrementally computed), M^+ (the expanded positive update incrementally computed), M^- (the expanded negative update incrementally computed), Δ_1 (the resulting update expansion incrementally computed), Q (the queue of clauses in BR with single literal bodies), r (ranges over the clauses).
Output: (Φ_1, Δ_1).
$\Phi_1 := \Phi$; $M^- := D^-$; $M^+ := D^+$;
Delete from Φ_1 all clauses r with $head(r) \in (M^+ \cup \neg.M^-)$ or with $body(r) \cap (\neg.M^+ \cup M^-) \neq \emptyset$;
FOR EACH clause $r \in \Phi_1$ **DO**
    $body(r) := body(r) \setminus (M^+ \cup \neg.M^-)$
END_DO
$BR := \{r \in \Phi_1 \mid head(r) = l \ \& \ \Delta \not\models l\}$;
FOR EACH clause $r = (l' \leftarrow l'_1) \in BR$ **DO**
    $Q := ENQUEUE(Q, r)$; $BR := BR \setminus \{r\}$
END_DO

% residue and expansion computation loop:
WHILE Q is not empty **DO**
$r := FRONT(Q)$; let $r = l \leftarrow l_1$; $Q := DEQUEUE(Q, r)$;
IF $l_1 \in \mathbf{B}$
THEN $M^- := M^- \cup \{l_1\}$ **ELSE** $M^+ := M^+ \cup \{\neg.l_1\}$
END_IF;
Delete from Φ_1 and from BR all clauses with l_1 in their bodies;
Delete from Φ_1 all clauses with $\neg.l_1$ in their heads;
Delete $\neg.l_1$ from bodies of clauses in Φ_1 and BR;
$BR := BR \cup \{r \in \Phi_1 \mid head(r) = l_1\}$
FOR EACH clause $r = (l' \leftarrow l'_1) \in BR$ **DO**
$Q := ENQUEUE(Q, r)$; $BR := BR \setminus \{r\}$
END_DO
END_DO
$\Delta_1 := (M^+, M^-)$;
Output (Φ_1, Δ_1).

Γ_{lim} is computed by an evident square time algorithm with the loop in which successive calls of f_expand and b_expand are effected till stabilization. □

As it turns out, the limit operator Γ_{lim} is more powerful then both operators Γ_f and Γ_b, and than their compositions.

Theorem 4 *For each $n \geq 1$ there exist compatible IC Φ and update Δ such that $(\Gamma_f \circ \Gamma_b)^{n+1}(\Phi, \Delta) \neq (\Gamma_f \circ \Gamma_b)^n(\Phi, \Delta)$.*

Proof. Let us consider the following IC Φ consisting of $2(n+1)$ clauses:
$r_1 :\ a_1 \leftarrow b_0$
$s_1 : \neg b_0 \leftarrow a_1, \neg b_1$
. . .
$r_{i+1} :\ a_{i+1} \leftarrow b_i$
$s_{i+1} :\ \neg a_i \leftarrow a_{i+1}, \neg b_{i+1}$
. . .
$r_{n+1} :\ a_{n+1} \leftarrow b_n$
$s_{n+1} :\ \neg a_n \leftarrow a_{n+1}, \neg b_{n+1}$.
Let $\Delta = (\{b_0\}, \emptyset)$. Then it is easy to check that $(\Gamma_f \circ \Gamma_b)^i(\Phi, \Delta) = (\Phi_i, \Delta_i)$, where $\Phi_i = \{r_{i+1}, s_{i+1}, ..., r_{n+1}, s_{n+1}\}$ and $\Delta_i = (\{b_0, a_1, b_1, ..., a_i, b_i\}, \emptyset)$ for all $1 \leq i \leq n+1$. Therefore,
$(\Gamma_f \circ \Gamma_b)^{n+1}(\Phi, \Delta) =$
$(\emptyset, (\{b_0, a_1, b_1, ..., a_{n+1}, b_{n+1}\}, \emptyset))$
$\neq (\Gamma_f \circ \Gamma_b(\Phi, \Delta))^n =$
$(\{r_{n+1}, s_{n+1}\}, (\{b_0, a_1, b_1, ..., a_n, b_n\}, \emptyset))$. □

5 Speeding-up the directed search

The simplest way to speed-up the standard algorithm of directed search is to reduce its search space by using the IC and the update optimized by Γ_{lim} in the

place of initial IC and update. The resulting operator

$$\Psi(I) = D\_search(I, \Gamma_{lim}(\Phi, \Delta))$$

is a conservative update operator, and is more efficient. Meanwhile, we can optimize D_search itself by applying extension operators inside its loops. The idea is to optimize the inner WHILE-loop of this algorithm. We apply Γ_{lim} twice in this loop: first time, after the new choice of H_{del}, and second time, after the new choice of H_{add}. The first application allows to precompute in advance the consequences of the choice of H_{del}. If the choice is correct, this simplifies Φ and still narrows search space H^+ for all H_{add} in the loop. If the choice is not correct, we cut the inner WHILE-loop at this early phase. The second application can narrow the search space H^+ for H_{add} or give the result immediately.

Algorithm $\mathcal{CU}$ computing a conservative update operator
Input: a DB state I, an update $\Delta = (D^+, D^-)$ compatible with an IC Φ.
Output: DB state I_1.

% Initial update expansion and IC simplification:
$(\Phi_\omega, (M^+, M^-)) := \Gamma_{lim}(\Phi, \Delta)$;
$\check{I} := (I \cup M^+) \setminus M^-$;
% New search space for H_{del} :
$H^- := \check{I} \setminus M^+$;
% The outer loop on H_{del} in the narrowed search space H^-:
$H_{del} := \emptyset$; $b := false$;
WHILE $\neg b$ **AND** $H_{del} \neq \ \perp$ **DO**
$H_{add} := \emptyset$; $c := false$;
% Computation of the initial state update determined by the choice of H_{del}; when correct, it is good for all choices of H_{add} in the inner loop:
$D^+_{del} := \check{I} \setminus H_{del}$; $D^-_{del} := M^- \cup H_{del}$;
% Precomputation of $D\_search(I, \Gamma_{lim}(\Phi_\omega, (D^+_{del}, D^-_{del}))$:
$(\Phi_{del}, (M^+_{del}, M^-_{del})) := \Gamma_{lim}(\Phi_\omega, (D^+_{del}, D^-_{del}))$;
IF $(M^+_{del} \cap M^-_{del} = \emptyset)$ % checking non contradictory choice of H_{del}
THEN % non contradictory choice
% search space definition for the current state updates H_{add} :
$H^+ := \mathbf{B} \setminus (M^+_{del} \cup M^-_{del})$;
% the inner loop on H_{add} in the choice space reduced by the H_{del} precomputation of the residue Φ_{del} and the expansion (M^+_{del}, M^-_{del}):
WHILE $\neg c$ **AND** $H_{add} \neq \ \perp$ **DO**
% The new update (D^+_{add}, D^-_{add}) determined by the choice of H_{add}:
$D^+_{add} := M^+_{del} \cup H_{add}$; $D^-_{add} := M^-_{del}$;
% An attempt to compute a new DB state through Γ_{lim}.
$(M^+_{add}, M^-_{add})) := \Gamma_{lim}(\Phi_{del}, (D^+_{add}, D^-_{add}))^{up}$;
% contradiction check:
IF $(M^+_{add} \cap M^-_{add} \neq \emptyset$ **OR** $M^+_{add} \not\models \Phi_{add})$
THEN % contradictory choice of H_{add}
$H_{add} := next_{H^+}(H_{add})$ % H_{add} abandoned
ELSE IF $M^+_{add} \setminus D^+_{add} \neq \emptyset$
THEN % Decreasing search space for H_{add}:

$H^+ := M^+_{add} \setminus M^+_{del}$:
$H_{add} := next_{H^+}(H_{add})$
ELSE % minimal correct state found
$I_1 := M^+_{add}$;
$c := true$
END_IF
END_IF
END_DO
END_IF
IF c **THEN** $b := c$
ELSE % H_{del} abandoned and the inner loop cut
$H_{del} := next_{H^-}(H_{del})$
END_IF
END_DO
Output I_1.

Theorem 5 *The algorithm $\mathcal{CU}$ implements a conservative update operator on compatible ICs and updates.*

The following artificial example illustrates this algorithm.

Example 3 *Let Φ have the clauses:*
$r_1 : \ b \leftarrow a, e, f;$
$r_2 : \ d \leftarrow \neg e, \neg g, \neg i;$
$r_3 : \neg a \leftarrow \neg e, \neg h, i;$
$r_4 : g \leftarrow \neg d, h;$
$r_5 : \ c \leftarrow a, \neg b;$
$r_6 : b \leftarrow c, d,$
and update $\Delta = (\{a\}, \{b\})$ be applied to DB state $I = \{b, d, e, f\}$.

Then $\mathcal{CU}$ runs as follows. Before the main loop Γ_f removes the rule r_5, and Γ_b removes r_6. The result is: $M^+ = \{a, c\}$, $M^- = \{b, d\}$, and $\Phi_\omega = \{ \ r'_1 : \ b \leftarrow e, f;$ $r_2 : \ d \leftarrow \neg e, \neg g, \neg i;$ $r_3 : \neg a \leftarrow \neg e, \neg h, i;$ $r'_4 : g \leftarrow h\}$. Then $\tilde{I} = \{a, c, e, f\}$ and $H^- = \{e, f\}$. The first loop **WHILE** *is started with $H_{del} = \emptyset$, $D^+_{del} = \tilde{I}$, $D^- = \{b, d\}$. Then, due to r'_1, Γ_f adds b to M^+_{del}. Hence, the condition of the first* **IF** *is false, so the new H_{del} is $next_{H^-}(\emptyset) = \{e\}$. Now in the main loop we get $M^+_{del} = \{a, c, f\}$, $M^-_{del} = \{b, d, e\}$, and $\Phi_{del} = \{ \ r'_2 : \ d \leftarrow \neg g, \neg i;$ $r'_3 : \neg a \leftarrow \neg h, i;$ $r'_4 : g \leftarrow h\}$. Therefore, $H^+ = \{g, h, i\}$ and for $H_{add} = \emptyset$ the first execution of the inner loop* **WHILE** *discovers the contradiction: $M^+_{add} = M^+_{del} \not\models \Phi_{add}$. Now $H_{add} = next_{H^+}(\emptyset) = \{g\}$ is considered, and the course of the second execution of the inner loop we obtain $D^+_{add} = \{a, c, f, g\} = M^+_{add}$, $\Phi_{add} = \{r'_3, r'_4\}$. So $M^+_{add} \models \Phi_{add}$. Here $\mathcal{CU}$ finishes with the result $I_1 = \{a, c, f, g\}$.*

6 Conclusion

Even though we have proven that the conservative updates are intractable in general, the method we propose extracts their reasonably large part which is incrementally computed in square time. It provides both, static and dynamic optimizations of the complete search. The next step would be to find more

powerful, nevertheless tractable expansion operators. E.g., we could use a more powerful backward operator if we abandoned the claim that a *single* literal in the body is refuted. However, the resulting operator would be nondeterministic. It is an interesting theoretical problem to implement efficiently some sort of such disjunctive backward operators.

References

1. Abiteboul, S.:Updates a new Frontier. In: *Proc. of the Second International Conference on the Theory of Databases, ICDT'88.* LNCS **326** (1988) 1-18.
2. Alferes, J.J.,Pereira, L.M.: Update-Programs Can Update Programs. In J.Dix, L.M. Pereira, T.C. Przymusinski, editors: *Second International Workshop, NMELP'96. Selected Papers.* LNCS **1216** (1997) 110-131.
3. Bonner, A.J., Kifer, M.: An Overview of Transaction Logic. *Theoretical Computer Science,* **133**(2) (1994), 2-5-265.
4. Decker H.: An extension of SLD by abduction and integrity maintenance for view updating in deductive databases. In: *Proc. of the 1996 International Conference on Logic Programming.* MIT Press, (1996), 157-169.
5. Dekhtyar, M., Dikovsky, A., Spyratos, N.: On Conservative Enforced Updates. In: Dix, J., Furbach, U., Nerode, A., editors: *Proceedings of 4th International Conference, LPNMR'97.* Dagstuhl Castle, Germany, LNCS **1265** (1997) 244-257.
6. Dekhtyar, M., Dikovsky, A., Spyratos, N.: On Logically Justified Updates. In: J. Jaffar, editor: *Proc. of the 1998 Joint International Conference and Symposium on Logic Programming.* MIT Press, (1998), 250-264.
7. Eiter, T., Gottlob, G.: On the complexity of propositional knowledge base revision, updates, and counterfactuals. Artificial Intelligence **57** (1992) 227-270.
8. Eshghi, K., Kowalski, R. A.: Abduction Compared with Negation by Failure. In: *Proc. of the 1989 International Conference on Logic Programming.* (1989)
9. Guessoum A., Lloyd J.W.: Updating knowledge bases. New Generation Computing, **8** (1990), 71-89.
10. Halfeld Ferrari Alves, M., Laurent, D., Spyratos, N., Stamate, D.: Update rules and revision programs. Rapport de Recherche Université de Paris-Sud, Centre d'Orsay, LRI **1010** (12 / 1995).
11. Kakas A.C., Mancarella P.: Database updates through abduction. IN: *Proc. 16th VLBD Conference.* (1990) 650-661.
12. Lloyd, J.W., Foundations of Logic Programming. Second, Extended Edition. Springer-Verlag. (1993)
13. Marek, V.W., Truszczyński, M.: Revision programming, database updates and integrity constraints. In: *International Conference on Data Base theory, ICDT.* LNCS **893** (1995) 368-382.
14. Przymusinski, T.C., Turner, H.: Update by Means of Inference Rules. In: V.W.Marek, A.Nerode, M.Truszczyński, editors, *Logic Programming and Nonmonotonic Reasoning.* Proc. of the Third Int. Conf. LPNMR'95, Lexington, KY, USA (1995) 166-174.