Incremental Expansion of Database Updates Through Integrity Constraints

M. Dekhtyar * **A. Dikovsky** † **N. Spyratos** ‡

Abstract

Une méthode pratique est proposée de calculer les mises à jour (MàJ) de bases de données en presence de contraintes d'integrité exprimées à l'aide de programmes logiques. Une grande partie des MàJ à effectuer réélement est en fait calculable en temps quadratique par un opérateur d'expansion monotone des MàJ initiales. Cet opérateur sert aussi à l'accélération d'un algorithme standard qui calcule un nouveau état modifié "le moins possible".

1 Introduction

During the last decade much effort went into the investigation of logic programming means for representing data and knowledge which evolve in time. The central problem is to cope with the nonmonotone nature of the change produced to the database or knowledge base state and to find an incremental and logically justified definition for that change. Traditionally logic programs serve as recursive definitions of updates. A logic program applies to the initial state and produces some of its models. In general it involves a negation, therefore its interpretation depends on a chosen semantics. E.g., in [1] fixpoint semantics of $Datalog^{\neg}$ is used, analogous to the fixpoint extensions of the FO logic in [11]. More recent research is based on stable [10], well-founded [18], or some other default rule semantics of logic programs, and follows the minimal change principle ([2, 14, 9]): the result of the update of the initial state I, specified by a program P, is defined as some model I_1 of P minimally deviating from I. Traditionally the *symmetric difference* of database states serves as distance (see [8] for references). Various techniques were found for constructing I_1 from I, e.g. revision programming [16], or inertia axioms [17, 3]. The problem with this tradition is that it does not treat integrity constraints (IC) imposed on

*Dept. of CS, Tver State Univ. 3 Zheljabova str. Tver, Russia, 170013 Email: Michael.Dekhtyar@tversu.ru

†Universite de Nantes, IRIN, UPRES EA N 2157, 2, rue de la Houssiniere BP 92208 F 44322 Nantes cedex 3 France Email: Alexandre.Dikovsky@irin.univ-nantes.fr and Keldysh Institute for Applied Math.

‡Université de Paris-Sud, LRI, U.R.A. 410 du CNRS, Bât. 490 F-91405 Orsay Cedex, France Email: Nicolas.Spyratos@lri.fr

states, i.e. properties of states that remain invariant with respect to updates.

In presence of ICs the update problem can be formulated as follows: *given an initial correct DB state I and an update Δ specifying the facts to add (D^+) and the facts to delete (D^-), one should find a correct DB state I_1 minimally deviating from I, in which Δ is accomplished (i.e. D^+ added and D^- deleted).* It is clear that in general $I_1 \neq (I \cup D^+) \setminus D^-$. One should introduce some additional changes in order to arrive at a model of Φ. This total real change is a partial function of Φ, Δ, and I : $I_1 = \Psi(\Phi, \Delta, I)$. This is illustrated by the following simplified example.

Example 1. Integrity constraints:

$position(X, dept\_chief) \leftarrow salary(X, Y), Y > 8000.$
$position(X, group\_leader) \leftarrow salary(X, Y), 5000 > Y > 3000.$
$position(X, engineer) \leftarrow salary(X, Y), Y \leq 3000.$
$\neg salary(X, Y) \leftarrow salary(X, Z), Y \neq Z.$
$\neg position(X, Y) \leftarrow position(X, Z), X \neq Z.$

database facts:

$salary(peter, 10000).$
$position(peter, dept\_chief).$
$salary(jenny, 4500).$
$position(jenny, group\_leader).$

Suppose that the external update requires that the salary of *peter* be lowered to 3000. This means that the fact $salary(peter, 3000)$ should be added to the state. However, this is only a part of the real change: as a consequence of the integrity constraints the facts $salary(peter, 10000)$ and $position(peter, dept\_chief)$ are to be deleted, and the fact $position$ $(peter, engineer)$ is to be added to the state.

This approach to updates was initiated and developed in [12, 4, 13, 5, 6]. In [4, 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. Unfortunately, the conservative updates are quite complex. We show in [7] that they fall into Σ_p^2 and some problems concerning their aspects are Σ_p^2-complete. Nevertheless, we demonstrate in [7] a new practical method of treating the conservative updates. The 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. We describe several monotone operators which 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 giving an expanded update $(M^+, M^-) \sqsupseteq (D^+, D^-)$ which forms a part of the real change implied by Φ and Δ. This accelerates the search of the model $\Psi(\Phi, \Delta, I)$ because every such model separates M^+ from M^-. In this paper we show how the expansion operators can be used to not only narrow the choice space but also to optimize the choice algorithm itself.

2 Notation

We assume that the reader is familiar with the basic concepts and terminology of logic programming (see [15]).

We fix a finite 1st order signature $\mathbf{S}$ with constants and no other function symbols. It is presumed that the signature never changes. This means in particular that we fix the finite sets $\mathbf{A}$, $\mathbf{L}$, $\mathbf{B}$ and $\mathbf{LB}$ respectively of all atoms, all literals, all ground atoms, and all ground literals in the signature $\mathbf{S}$. In this context a logic program with variables is a succinct representation of its finite ground instant. As all the algorithms we consider, as well as the complexity bounds, apply to the ground instances of logic programs, we shall presume in the sequel that the logic programs are ground.

A literal contrary to a literal l is denoted by $\neg.l$. An IC is an extended logic program Φ, i.e. a finite set of clauses of the form $r = (l \leftarrow l_1, ..., l_n)$ where $n \geq 0$ and (with our assumption) $l, l_i \in \mathbf{LB}$. $\mathbf{IC}$ denotes the set of all integrity constraints in the signature $\mathbf{S}$. For a clause r $head(r)$ denotes its *head*, and $body(r)$ its *body*. We shall treat $body(r)$ as a set of literals. A (total) *interpretation* or a *DB state* is a subset of $\mathbf{B}$. A (ground) clause $r = (l \leftarrow l_1, ..., l_n)$ is *valid in a DB state* I (denoted $I \models r$) if $I \models l$ whenever $I \models l_i$ for each $1 \leq i \leq n$. For a DB state I and for an atom a $I \models a$ means $a \in I$, and for a literal $l = \neg a$ $I \models l$ means $a \notin I$. I is a *model* of IC Φ (denoted $I \models \Phi$) if every clause in Φ is valid in I.

A pair $\Delta = (D^+, D^-)$ where D^+, D^- are subsets of $\mathbf{B}$, and $D^+ \cap D^- = \emptyset$, is called an *update*. Intuitively, the atoms of D^+ are to be added to I, and those of D^- are to be removed from I. $\mathbf{UP}$ will denote the set of all updates in the signature constants in $\mathbf{S}$. We say that $\Delta = (D^+, D^-)$ is *accomplished* in I if $D^+ \subseteq I$ and $D^- \cap I = \emptyset$.

Let Γ be an operator of the type $\Gamma : \mathbf{IC} \times \mathbf{UP} \rightarrow \mathbf{IC} \times \mathbf{UP}$, $\Gamma(\Phi, \Delta) = (\Phi', \Delta')$, 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 $\Gamma^\omega(\Phi, \Delta) = \lim_{n \to \infty} \Gamma^n(\Phi, \Delta)$.

3 Conservative update operators

We start by citing several definitions and facts from [7] and [4].

1. In general an update may contradict a constraint. Their compatibility is formalized as follows.

Definition 1. *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* $Acc(\Phi, \Delta) \neq \emptyset$.

We prove in [7] that the compatibility problem is NP-complete.

2. In [4] we propose the following minimal change criterion:

Definition 2. *Let* I, I_1, I_2 *be three DB states, and* K *be a class of DB states.*
(1) We set $I_1 <_i^I I_2$ *if* $I \cap I_2 \subsetneq I \cap I_1$.
(2) We set $I_1 <_d^I I_2$ *if* $I_1 \setminus I \subsetneq I_2 \setminus I$.
(3) We say that I_1 *is* minimally deviating from I with respect to K *if*
$\forall I' \in K\ (\neg(I' <_i^I I_1)\ \&\ (I' \equiv_i^I I_1 \longrightarrow \neg(I' <_d^I I_1)))$,
where $I_1 \equiv_i^I I_2$ *means that* $I \cap I_1 = I \cap I_2$.

3. In terms of this minimal change criterion the *conservative update operators* are defined in [4] 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 is a* conservative update operator *if for each DB state* I *:*
(AC1) $\Psi(I)$ *is a model of* Φ,
(AC2) Δ *is accomplished in* $\Psi(I)$,
(AC3) $\Psi(I)$ *is minimally deviating from* I *with respect to* $Acc(\Phi, \Delta)$.

4. Since DB states and updates are finite, and the domain is closed, there evidently exist exhaustive choice algorithms which implement conservative update operators. It is always possible to impose on the search a certain direction, which conforms to our minimal change criterion. For example, if we choose a successor function $next_X$ for a topological order with respect to set inclusion on subsets of a set of atoms X, and fix a constant $\perp$ so that $next_X(X) = \perp$, then the following simple algorithm described in [7] correctly implements such a directed search.

Algorithm $D{-}search(I, \Phi, \Delta)$
Input: a DB state I, and an update $\Delta = (D^+, D^-)$ compatible with a ground IC Φ.
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}
c, %% boolean, stabilization flag
Output: I_1.
$\tilde{I} := (I \cup D^+) \setminus D^-$; %% search space definition 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.

As we have made it clear in [7], there is no general theoretical method which would optimize this algorithm because even the problem whether a given atom falls into I_1 for fixed input state I and update Δ is Σ_p^2-complete. However, there may exist practical methods of speeding-up the standard search directed by our minimal change criterion. One such efficient method we have proposed in [7]. This method will be reinforced in this paper.

4 Update expansion operators

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.

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

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

These definitions allow to carry over the validity of literals from updates to the interpretations in which the updates are accomplished. Both relations $\models$ and $\not\models$ defined by Table 1 are monotone with respect to updates.

Let us order **IC** by setting $\Phi_1 \preceq \Phi_2$ if $\forall r \in \Phi_1\ \exists r' \in \Phi_2$ $(head(r) = head(r')\ \&\ body(r) \subseteq body(r'))$. This partial order conforms with the following residue operator simplifying logic programs via updates.

Definition 4. *The residue of* Φ *with respect to* $\Delta = (D^+, D^-)$ *is defined as*

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

Any model of Φ, in which Δ is accomplished contains D^+ and does not intersect with D^-. Therefore, it is also a model of the residue of Φ with respect to Δ. Indeed, $res(\Phi,\Delta)$ results from Φ by removing the clauses whose heads are required by Δ, or whose bodies contradict Δ, and then by cancelling the literals required by Δ in the bodies of the resting clauses.

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 $r_2': \neg b \leftarrow \neg a, d$.*

It is clear that for any Φ and Δ $res(\Phi,\Delta) \preceq \Phi$. The following properties of *res* ensure it's incremental computation.

Lemma 1.

(1) For every $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$

$$res(res(\Phi,\Delta),\Delta) = res(\Phi,\Delta).$$

(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 for every $\Phi \in \mathbf{IC}$

$$res(\Phi,\Delta) = res(res(\Phi,\Delta_1),\Delta_2).$$

(3) For an IC Φ let $\Phi = \Phi_1 \cup \Phi_2$ be a partition. Then for any $\Delta \in \mathbf{UP}$

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

We propose in [7] 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 $\Delta = (D^+, D^-) \in \mathbf{UP}$*

- $D^+ \subseteq \Gamma(\Phi,\Delta)^+$, $D^- \subseteq \Gamma(\Phi,\Delta)^-$,
- $\Gamma(\Phi,\Delta)^{ic} \preceq \Phi$,
- $(\Phi,\Delta) \equiv_u \Gamma(\Phi,\Delta)$.

From this definitions it follows immediately that the set of all update expansion operators is closed under composition. The particular update expansion operators we use 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 $\Delta = (D^+, D^-) \in \mathbf{UP}$:
$B(\Phi,\Delta)^+ \;=\; D^+ \;\cup\; \{a \in \mathbf{B} \mid \exists r \in \Phi\ (\Delta \not\models head(r)\ \&\ body(r) = \neg a\alpha\ \&\ \Delta \models \alpha)\}$
$B(\Phi,\Delta)^- \;=\; D^- \cup \{a \in \mathbf{B} \mid \exists r \in \Phi\ (\Delta \not\models head(r)\ \&\ body(r) = a\alpha\ \&\ \Delta \models \alpha)\}.$

Forward update expansion Γ_f :

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

Backward update expansion Γ_b :

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

These operators have good properties that guarantee the existence of limits for the directed sets of their iterations. In particular, we prove the following their properties in [7].

Lemma 2.
(1) The operators F and B are monotone on **UP**,
(2) Let $|\Phi|$ be the size of $\Phi \in \mathbf{IC}$ (the number of literals in all clauses of Φ). Then for every $\Delta \in \mathbf{UP}$ $res(\Phi,\Delta) \preceq \Phi$ and therefore $|res(\Phi,\Delta)| \leq |\Phi|$.
(3) For any pair $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$ there is $n \geq 0$ such that $\Gamma_f(\Phi,\Delta) = \gamma_f^n(\Phi,\Delta)$.
(4) For any pair $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$ there is $m \geq 0$ such that $\Gamma_b(\Phi,\Delta) = \gamma_b^m(\Phi,\Delta)$.
(5) For any $\Phi \in \mathbf{IC}$ and $\Delta \in \mathbf{UP}$ the following equalities hold:
$\Gamma_f(\Phi,\Delta)^{ic} = res(\Phi, \Gamma_f(\Phi,\Delta)^{up})$ *and*
$\Gamma_b(\Phi,\Delta)^{ic} = res(\Phi, \Gamma_b(\Phi,\Delta)^{up})$.

Theorem 1. *The operators Γ_f and Γ_b are update expansion operators.*

5 Computation of expansion

The backward expansion operator we use may seem rather weak because the backward operator B is deterministic and it "diagonalizes" only one literal in a clause body. In fact its effect may be quite strong which is illustrated by the following example.

Example 3. *Let Φ be the IC:*

$$\Phi = \left\{ \begin{array}{lcl} r_1: & b & \leftarrow a, c, \neg g \\ r_2: & f & \leftarrow a, d \\ r_3: & \neg d & \leftarrow a, g \\ r_4: & \neg e & \leftarrow \neg a, \neg b, f \\ r_5: & \neg f & \leftarrow a, \neg h \\ r_6: & \neg k & \leftarrow f, h \end{array} \right\}$$

Consider the update $\Delta = (\{a, d, f\}, \{b\})$. Then the ic component of the first degree of $\gamma_b(\Phi, \Delta)$ is the program

$$\gamma_b^1(\Phi, \Delta)^{ic} = \left\{ \begin{array}{lcl} r_1: & b & \leftarrow c, \neg g \\ r_3: & \neg d & \leftarrow g \\ r_5: & \neg f & \leftarrow \neg h \\ r_6: & \neg k & \leftarrow h \end{array} \right\}$$

and the corresponding expanded update is $\gamma_b^1(\Phi, \Delta)^{up} = (\{a, d, f, h\}, \{b, g\})$. The second degree gives the residue program

$$\gamma_b^2(\Phi, \Delta)^{ic} = \left\{ \begin{array}{lcl} r_1: & b & \leftarrow c \\ r_6: & \neg k & \leftarrow \end{array} \right\}$$

and the expanded update $\gamma_b^2(\Phi, \Delta)^{up} = (\{a, d, f, h\}, \{b, g, c\})$. The third degree gives the residue program

$$\gamma_b^3(\Phi, \Delta)^{ic} = \left\{ \begin{array}{lcl} r_6: & \neg k & \leftarrow \end{array} \right\}$$

and the expanded update $\gamma_b^3(\Phi, \Delta)^{up} = \gamma_b^2(\Phi, \Delta)^{up}$, which leads to stabilization at the next step:

$$\Gamma_b(\Phi, \Delta)^{ic} = \left\{ \begin{array}{lcl} r_6: & \neg k & \leftarrow \end{array} \right\}$$

and $\Gamma_b(\Phi, \Delta)^{up} = (\{a, d, f, h\}, \{b, g, c\})$.

This effect becomes more clear if we attach to the diagonalized literals their ranks.

Definition 6. *Let $\Phi \in$ **IC** and $\Delta = (D^+, D^-) \in$ **UP**. Then each $l \in D^+$ has the positive b-rank $\rho_b^+(l) = 0$, and each $l \in D^-$ has the negative b-rank $\rho_b^-(l) = 0$. Let $l \leftarrow l_1\alpha$ be a clause in which l and all literals in α have (positive or negative) b-ranks, $r \geq 0$ being the maximum of these ranks, and let l contradict to its rank (i.e. $l = \neg a$ and the atom a has a positive finite rank, or l is an atom and it has a negative finite rank). Then*
$\rho_b^+(l_1) = r + 1$, *if* $l_1 = \neg a,\ a \in \mathbf{B}$, *and*
$\rho_b^-(l_1) = r + 1$, *if* $l_1 = a,\ a \in \mathbf{B}$.
For a literal l which has no finite rank we set $\rho_b^+(l) = \omega$.

For example, in the preceding example $\rho_b^+(h) = 1,\ \ \rho_b^-(g) = 1,\ \ \rho_b^-(c) = 2$. It is not difficult to prove the following statement.

Lemma 3. *For any IC Φ, update Δ, and literal l $\ l \in \Gamma_b(\Phi, \Delta)^+ \cup \Gamma_b(\Phi, \Delta)^-$ iff $\ l$ has a finite b-rank.*

The combination of the operators Γ_f, Γ_b turns out to be surprisingly productive. Meanwhile, both operators are computed in linear time.

The algorithm *f_expand* computing the operator Γ_f
Input: An update $\Delta = (D^+, D^-)$, compatible with a ground IC Φ.
Local variables:
Φ_1 (the resulting IC residue incrementally computed),
M^+ (the expanded positive update incrementally computed),
M^+_{new} (one-step positive update expansion),
M^-, M^-_{new} (same for the negative update expansions),
Δ_1 (the resulting update expansion incrementally computed).
Output: $\Gamma_f(\Phi, \Delta)$.
%% local variables initialization:
$\Phi_1 := \Phi$; $M^+ := D^+$; $M^- := D^-$;
Delete from Φ_1 all clauses r with $head(r) \in M^+$;
Delete from Φ_1 all clauses r with $\neg.head(r) \in M^-$;
Delete from Φ_1 all clauses r 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;
$M^+_{new} := \{a \in \mathbf{B} \mid \exists r \in \Phi_1\ (a = head(r)\ \&\ \Delta \models body(r))\}$;
$M^-_{new} := \{a \in \mathbf{B} \mid \exists r \in \Phi_1\ (\neg.a = head(r)\ \&\ \Delta \models body(r))\}$;
$M^+ := M^+ \cup M^+_{new}$;
$M^- := M^- \cup M^-_{new}$;
$\Delta_1 := (M^+, M^-)$;
%% residue and expansion computation loop:
WHILE $(M^+_{new} \neq \emptyset)\ OR\ (M^-_{new} \neq \emptyset)$ **DO**
Delete from Φ_1 all clauses r with $head(r) \in M^+_{new}$;
Delete from Φ_1 all clauses r with $\neg.head(r) \in M^-_{new}$;
Delete from Φ_1 all clauses r with $body(r) \cap \neg.M^+_{new} \neq \emptyset$;
Delete from Φ_1 all clauses r with $body(r) \cap M^-_{new} \neq \emptyset$;
FOR EACH clause $r \in \Phi_1$ **DO**
$body(r) := body(r) \setminus (M^+_{new} \cup \neg.M^-_{new})$;
END_DO;
$M^+_{new} := \{a \in \mathbf{B} \mid \exists r \in \Phi_1\ (a = head(r)\ \&\ \Delta_1 \models body(r))\}$;
$M^-_{new} := \{a \in \mathbf{B} \mid \exists r \in \Phi_1\ (\neg.a = head(r)\ \&\ \Delta_1 \models body(r))\}$;
$M^+ := M^+ \cup M^+_{new}$;
$M^- := M^- \cup M^-_{new}$;
$\Delta_1 := (M^+, M^-)$;
END_DO;
Output (Φ_1, Δ_1).

The next algorithm implements a sequential computation of the operator 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 each execution 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 ground IC Φ.
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).
%% local variables initialization:
$\Phi_1 := \Phi$; *nhYY OcTicOOcOcTnOcTfnXTrfoHYshYTjrcfhDYsrNnOcX,dcOOcX.cOOcD jrcTicOOOcTsYsnOcTpγOcTic*

END_DO
END_DO
$\Delta_1 := (M^+, M^-)$;
Output (Φ_1, Δ_1).

6 Practical optimization of the directed search

As we have shown in [7], the iteration of the composition of forward and backward expand operators $(\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta)$ is again an expand operator and it is computed in square time. This fact has given the following optimized version of the standard directed search algorithm:

$$\Psi(I) = D\!-\!search(I, (\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta)).$$

Here we present even more optimized version of the computation of $\Psi(I)$. The new idea is to optimize the inner WHILE-loop of this algorithm. Indeed, after the new subtracted subset H_{del} has been fixed at the current step of the outer WHILE-loop, all the sets tested in the course of the inner WHILE-loop for being a model of the IC are supersets of $\tilde{D}^+ = (M^+ \cup I) \setminus H_{del}$ where $M^+ = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta))^+$ and I is the initial state. At the same time $\tilde{D}^+$ does not intersect with $\tilde{D}^- = M^- \cup H_{del}$, where $M^- = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta))^-$. So we can apply the expansion algorithm to the pair $(\tilde{D}^+, \tilde{D}^-)$ and construct the residue $\tilde{\Phi}$ of the IC Φ with respect to this greater update. $\tilde{\Phi}$ is simpler than $((\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta))^{ic}$ and this is the first source of the optimization. The other source is that $(\tilde{D}^+, \tilde{D}^-)$ is a part of the real change caused by $(D^+, D^- \cup H_{del})$. Therefore, if this expansion of the update (M^+, M^-) is not contradictory, it still narrows the choice space. If it is contradictory, then we discover at the early phase that this choice of H_{del} is wrong and we cut the inner WHILE-loop. Here is the optimized algorithm.

Algorithm $\mathcal{CU}$ computing a conservative update operator
Input: a DB state I, an update $\Delta = (D^+, D^-)$ compatible with an IC Φ.
Output: A DB state $I_1 = \Psi[\Phi, \Delta](I)$.
%% Precomputation of $D\!-\!search(I, (\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta))$:
$M^+ = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta))^+$;
$M^- = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta))^-$;
$\Phi_\omega = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi, \Delta))^{ic}$;
$\tilde{I} := (I \cup M^+) \setminus M^-$;
%% search space definition for the current state updates H_{del} and H_{add}:
$H^- := \tilde{I} \setminus M^+$; %% search space for H_{del}
$H^+ := \mathbf{B} \setminus (\tilde{I} \cup M^-)$; %% search space for H_{add}
%% The outer loop on H_{del} in the search space H^- narrowed by the precomputation:
$H_{del} := \emptyset$; $b := false$;
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}; if correct, it is good for all choices of H^{add}:
$D^+_{del} := \tilde{I} \setminus H_{del}$;
$D^-_{del} := M^- \cup H_{del}$;
%% Precomputation of $D{-}search(I, (\Gamma_f \circ \Gamma_b)^\omega(\Phi_\omega, (D^+_{del}, D^-_{del})))$:
$M^+_{del} = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi_\omega, (D^+_{del}, D^-_{del})))^+$;
$M^-_{del} = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi_\omega, (D^+_{del}, D^-_{del})))^-$;
$\Phi_{del} = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi_\omega, (D^+_{del}, D^-_{del})))^{ic}$;
IF $(M^+_{del} \cap M^-_{del} = \emptyset)$ %% checking non contradictory choice of H_{del}
THEN %% non contradictory choice
%% 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}:
$D^+_{add} := M^+_{del} \cup H_{add}$;
$D^-_{add} := M^-_{del}$;
%% Computation of the new DB state through $D{-}search(I, (\Gamma_f \circ \Gamma_b)^\omega(\Phi_\omega, (D^+_{add}, D^-_{add})))$:
$M^+_{add} = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi_{del}, (D^+_{add}, D^-_{add})))^+$;
$M^-_{add} = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi_{del}, (D^+_{add}, D^-_{add})))^-$;
$\Phi_{add} = ((\Gamma_f \circ \Gamma_b)^\omega(\Phi_{del}, (D^+_{add}, D^-_{add})))^{ic}$;
%% 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 %% minimal correct state found
$I_1 := M^+_{add}$;
$c := true$
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 2. *For any DB state I, and compatible Φ and Δ the algorithm $\mathcal{CU}$ finds a DB state I_1 such that:*
(1) $I_1 \in Acc(\Phi, \Delta)$.
(2) I_1 is minimally deviating from I with respect to $Acc(\Phi, \Delta)$.

7 Conclusion

The expansion operator $(\Gamma_f \circ \Gamma_b)^\omega$ introduced in [7] and used in this paper is interesting for itself, because it constructs a large part of the real change implied by the given update relative to the fixed IC, and it is effectively computed. It is defined through two update propagation operators: the forward operator F and the backward operator B, which are very natural. F is analogous to the immediate consequence operator. As it concerns the backward operator, we use a particular operator B which is *deterministic*. We could use a more powerful 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.

Acknowledgement. We gratefully acknowledge the stimulating critics provided by the anonymous referees.

References

[1] Abiteboul, S., Vianu, V.: Fixpoint Extensions of First-Order Logic and Datalog-like Languages. In: *Proc. of the 4th Symp. on Logic in Comput¿ Sci.* IEEE, (1989), 71-78.

[2] Alchourón, C.E., Gärdenfors, P., and Makinson, D.: On the logic of theory change: partial meet contraction and revision functions. *J. Symbolic Logic.* **50** (1985) 510-530.

[3] 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** (1996) 110-131.

[4] 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.

[5] Dekhtyar, M., Dikovsky, A., Spyratos, N.: Enforcement of Integrity Constraints by Means of Minimal Sufficient Changes. In: A. Dikovsky, editor: *Programming and Computer Software. Special issue. "Application of Logics in Programming".* (1998) N 2, 27-37.

[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] Dekhtyar, M., Dikovsky, A., Spyratos, N.: Monotone Expansion of Updates in Logical Databases. Submitted for publication.

[8] Eiter, T., Gottlob, G.: On the complexity of propositional knowledge base revision, updates, and counterfactuals. Artificial Intelligence **57** (1992) 227-270.

[9] Fagin, R., Kuper, G., Ullman, J., and Vardi, M.Y.: Updating Logical Databases. In: P. Kanellakis, editor, *Advances in Computing Research*, JAI Press **3** (1986) 1-18.

[10] Gelfond, M., Lfschitz, V.: The stable semantics for logic programs. In: R.Kovalsky and K.Bowen, editors, *Proc. of the 5th Intern. Symp. on Logic Programming.* Cambridge, MA, MIT Press (1988) 1070-1080.

[11] Gurevich, Y., Shelah, S.: Fixed-Point Extensions of First Order Logic. In: *Proc. of the 26th Symp. on Found. of Comp. Sci.* (1985), 346-353.

[12] 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).

[13] Laurent, D., Spyratos, N., Stamate, D.: Deterministic enforcement of constraints. In: A. Dikovsky, editor: *Programming and Computer Software. Special issue. "Application of Logics in Programming".* (1998) N 2, 38-57.

[14] Katsuno, H., Mendelzon, A. O.: Propositional knowledge base revision and minimal change. Artificial Intelligence **52** (1991) 253-294.

[15] Lloyd, J.W., Foundations of Logic Programming. Second, Extended Edition. Springer-Verlag. (1993)

[16] Marek, V.W., Truszciński, M.: Revision programming, database updates and integrity constraints. In: *International Conference on Data Base theory, ICDT.* LNCS **893** (1995) 368-382.

[17] 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.

[18] Van Gelder, A., Ross, K.A., and Schlipf, J.S.: The Well-Founded Semantics for General Logic Programs. Journal of the ACM **38** (1991) 620-650.