Model-Based Techniques for View Deletion in Databases

Chandrabose Aravindan† and Peter Baumgartner‡
Fachbereich Informatik, Universität Koblenz-Landau,
Rheinau 1, D-56075 Koblenz, Germany
{arvind,peter}@informatik.uni-koblenz.de

In this paper, we show how techniques from disjunctive logic program-
ing and classical first-order theorem proving can be used for efficient
deductive) database updates. The key idea is to transform the given
database, together with the update request, into a disjunctive logic pro-
gram and apply disjunctive techniques (such as minimal model reasoning)
to solve the original update problem. We present two variants of our al-
gorithm both of which are of polynomial space complexity. One variant,
which is based on offline preprocessing, is of polynomial time complexity.
We also show that both variants are rational in the sense that they sat-
sify certain rationality postulates stemming from philosophical works on
belief dynamics.

Keywords: database updates, disjunctive logic programming, hyper tableaux,
minimal model reasoning, belief dynamics

1 Introduction

View update in databases is an important problem that has recently attracted attention
of researchers from both deductive and relational fields [AD95, Bry90, Dec90,
Dec96, GL90, GL91, KM90, Tom88, DB82, Kel85, Lan90, for example] ([Abi88] pro-
vides a survey of works in this regard). One crucial aspect of an algorithm for view
update is the satisfaction of certain rationality postulates stemming from philosop-
hical works on rationality of change [Gär92, GR95, for example]. This aspect was
studied in detail in [AD95, Ara95], where an algorithm for database deletion that
satisfies all the rationality postulates was presented. However, a serious drawback of

---

† Funded by the DFG ("Deutsche Forschungsgemeinschaft") under grant Fu 263/3-1
‡ Funded by the DFG within the research programme “Deduction” under grant Fu 263/2-2

A preliminary and shorter version of this paper was published as [AB97]
this and other known rational algorithms (such as the one from Tomasic [Tom88]) is that they are of exponential space and time complexity.

In this paper, we present a radically different approach to rational view updates in databases, resulting in an algorithm of polynomial space complexity. We also show that, with reasonable offline preprocessing, polynomial time complexity can be achieved. For the simplicity of presenting the main ideas, in this paper we restrict our attention to definite datalog programs (note that relational databases can be represented by definite programs) and view deletion only. The approach we present here is very closely related to our diagnosis setup presented in [BFFN97b, BFFN97a], where hyper tableaux calculus [BFN96] was used for efficiently solving model based diagnosis tasks. This close relationship enables us to use our existing, efficient implementation for diagnosis applications for view updates as well.

The basic idea in [BFFN97b, BFFN97a] is to employ the model generation property of hyper tableaux to generate models and read off diagnosis from them. One specific feature of this diagnosis algorithm is the use of semantics (by transforming the system description and the observation using an “initial model” of the correctly working system) in guiding the search for a diagnosis. This semantical guidance by program transformation turns out to be useful for database updates as well. More specifically, we use a (least) Herbrand Model of the given database to transform it along with the update request into a disjunctive logic program in such a way that the models of this transformed program stand for possible updates. Thus known disjunctive logic programming and first-order theorem proving techniques are exploited for efficient and rational view updates.

We discuss two ways of transforming the given database together with the view deletion request into a disjunctive logic program, resulting in two variants of view deletion algorithm. In the first variant, a simple and straightforward transformation is employed. But unfortunately not all models of the transformed program stand for rational deletions. In order to be rational, we show that a rationality axiom itself (strong relevance policy) could be used as a test to filter out models representing non-rational deletions. Interestingly, this test based on a rationality axiom turns out be equivalent to the groundedness test used by Ilkka Niemelä for generating minimal models of disjunctive logic programs [Nie96c]. These two concepts (strong relevance policy and groundedness test) come from two different fields (belief dynamics and minimal model reasoning resp.) and this equivalence provides more insights into the issue (minimization) common to both the fields. Further, this equivalence implies that all minimal models (minimal wrt the EDB atoms) of the transformed program stand for rational deletions. Not surprisingly, all deletions obtained through this algorithm result in minimal change.

The second variant of the algorithm uses the Least Herbrand Model of the given database for the transformation. In fact, what we referred to as offline preprocessing before is exactly this computation of the Least Herbrand Model (which can be done in polynomial space and time). This variant is very meaningful in applications where views are materialized for efficient query answering. The advantage of using the Least Herbrand Model for the transformation is that all models of the transformed disjunctive logic program (not just the minimal ones) stand for rational deletions.
This is a very nice result since a model of a disjunctive logic program can be computed in polynomial space and time.

The rest of the paper is organized as follows: We first briefly recall the rationality of change and the hyper tableaux calculus in Section 2. We then present two variants of our rational and efficient algorithm for view deletion in Section 3. The paper is concluded with some comments on our approach and indications for further work.

2 Background

2.1 Rationality of change

Rationality of change has been studied at an abstract philosophical level by various researchers, resulting in well known AGM Postulates for revision [AGM85, Gär92, GR95, for example]. However, it is not clear how these rationality postulates can be applied in real world problems such as database updates and this issue has been studied in detail by works such as [AD95, Ara95]. In the sequel, we briefly recall the postulates and an algorithm for contraction based on abduction from [AD95, Ara95].

Formally, a knowledge base $KB$ is defined as a finite set of sentences from language $L$, and divided into two parts: an immutable theory $KB_I$, which is the fixed part of the knowledge; and an updatable theory $KB_U$. Because of the duality of revision and contraction, it is enough to consider one, and rationality postulates for contracting a sentence $\alpha$ from $KB_I$ written as $KB-\alpha$ is produced below.

**Definition 2.1**

Let $KB$ be a knowledge base with an immutable part $KB_I$. Let $\alpha$ and $\beta$ be any two sentences. Then, $\alpha$ and $\beta$ are said to be $KB$-equivalent iff the following condition is satisfied: For all set of sentences $E$: $KB_I \cup E \vdash \alpha$ iff $KB_I \cup E \vdash \beta$.

**Definition 2.2 (Rationality Postulates)**

(\text{KB—1}) \text{ (Inclusion)} \hspace{1cm} KB-\alpha \subseteq KB

(\text{KB—2}) \text{ (Immutable-inclusion)} \hspace{1cm} KB_I \subseteq KB-\alpha

(\text{KB—3}) \text{ (Vacuity)} \hspace{1cm} \text{If } \alpha \notin \text{Cn}(KB), \text{ then } KB-\alpha = KB

(\text{KB—4}) \text{ (Immutable-success)} \hspace{1cm} \text{If } KB_I \vdash \alpha, \text{ then } \alpha \notin \text{Cn}(KB-\alpha)

(\text{KB—5}) \text{ (Preservation)} \hspace{1cm} \text{If } \alpha \text{ and } \beta \text{ are } KB\text{-equivalent, then } KB-\alpha = KB-\beta

(\text{KB—6.1}) \text{ (Strong relevance)} \hspace{1cm} \text{If } \beta \in KB \setminus KB-\alpha, \text{ then } \alpha \in \text{Cn}(KB-\alpha \cup \{\beta\})

(\text{KB—6.2}) \text{ (Relevance)} \hspace{1cm} \text{If } \beta \in KB \setminus KB-\alpha, \text{ then } \exists KB' \text{ with } KB-\alpha \subseteq KB' \subseteq KB \text{ s.t. } \alpha \notin \text{Cn}(KB') \text{ and } \alpha \in \text{Cn}(KB' \cup \{\beta\})

(\text{KB—6.3}) \text{ (Weak relevance)} \hspace{1cm} \text{If } \beta \in KB \setminus KB-\alpha, \text{ then } \exists KB' \text{ with } KB' \subseteq KB \text{ s.t. } \alpha \notin \text{Cn}(KB') \text{ and } \alpha \in \text{Cn}(KB' \cup \{\beta\})
Note that we have three variants of the relevant postulate of varying strength. The weaker forms are motivated by various works of Hansson [Han91b, Han91a].

Now we recall an algorithm for contraction based on abduction presented in [AD95, Ara95]. Some basic definitions required for the algorithm are presented first.

**Definition 2.3**
Let $KB$ be a knowledge base and $\alpha$ a sentence. An abductive explanation $\Delta$ for $\alpha$ wrt $KB_I$ is a set of abducibles s.t. $\Delta \cup KB_I \models \alpha$ and $\Delta \cup KB_I$ is consistent. An explanation is *minimal* if no proper subset of it is an explanation. It is said to be *locally minimal*, iff there exists a subset $KB'_I$ of $KB_I$ s.t. $\Delta$ is a minimal abductive explanation of $\alpha$ wrt $KB'_I$. Further, $\Delta$ is said to be *KB-Closed* iff $\Delta \subseteq KB_U$. ■

**Example 2.4**
Consider a knowledge base $KB$ whose immutable part $KB_I = \{p \leftarrow q \wedge r, p \leftarrow r\}$, where $q$ and $r$ are abducibles. Clearly, $\Delta_1 = \{r\}$ is the only minimal abductive explanation for $p$ wrt $KB_I$. $\Delta_2 = \{q, r\}$ is an abductive explanation for $p$ wrt $KB_I$, but not minimal. However, $\Delta_2$ is a locally minimal abductive explanation for $p$ wrt $KB_I$, since it is a minimal explanation for $p$ wrt $\{p \leftarrow q \wedge r\}$ which is a subset of $KB_I$. The concept of a locally minimal abductive explanation is computationally attractive, since a minimal abductive explanation is more expensive to compute. ■

The general contraction algorithm of [AD95, Ara95] is reproduced here as Algorithm 1. The basic idea behind this algorithm is to generate all (locally minimal) explanations for the sentence to be contracted and determine a hitting set for these explanations. Since all (locally minimal) explanations are generated this algorithm is of exponential space and time complexity.

**Definition 2.5 (Hitting set)**
Let $S$ be a set of sets. Then a set $HS$ is a hitting set of $S$ iff $HS \subseteq \bigcup S$ and for every non-empty element $R$ of $S$, $R \cap HS$ is not empty. Further, $HS$ is a minimal hitting set iff no proper subset of it is a hitting set.

**Algorithm 1** General contraction algorithm

Input: A knowledge base $KB = KB_I \cup KB_U$ and a sentence $\alpha$ to be contracted.

Output: A new knowledge base $KB' = KB_I \cup KB'_U$

begin

1. Construct a set $S = \{X \mid X$ is a KB-closed locally minimal abductive explanation for $\alpha$ wrt $KB_I\}$.

2. Determine a hitting set $\sigma(S)$.

3. Produce $KB' = KB_I \cup (KB_U \backslash \sigma(S))$ as a result.

end.
2.2 Hyper Tableaux Calculus

In [BFN96] a variant of clausal normal form tableaux called “hyper tableaux” has been introduced. Since the hyper tableaux calculus constitutes the basis for our view update algorithm, we will briefly recall it. It is sufficient to restrict to the ground version here.

We assume that the reader is familiar with the basic concepts of propositional logic. Clauses, i.e., multisets of literals, are usually written as the disjunction $A_1 \lor \cdots \lor A_m \lor \neg B_1 \lor \cdots \lor \neg B_n$ or as an implication $A_1 \lor \cdots \lor A_m \leftarrow B_1 \land \cdots \land B_n$ ($m \geq 0$, $n \geq 0$). The literals $A_1, \ldots, A_m$ (resp. $B_1, \ldots, B_n$) are called the head (resp. body) of the clause. With $\overline{T}$ we denote the complement of a literal $L$. Two literals $L$ and $K$ are complementary if $\overline{L} = K$.

From now on $D$ always denotes a finite ground clause set, also called database, and $\Sigma$ denotes its signature, i.e. the set of all predicate symbols occurring in it. We consider finite ordered trees $T$ where the nodes, except the root node, are labeled with literals. In the following we will represent a branch $b$ in $T$ by the sequence $b = L_1, \ldots, L_n$ ($n \geq 0$) of its literal labels, where $L_1$ labels an immediate successor of the root node, and $L_n$ labels the leaf of $b$. Concatenation of node sequences is denoted by “;”. So, for instances $(b, L_{n+1})$ denotes the node sequence carrying the respective labels $L_1, \ldots, L_n, L_{n+1}$. By a partial branch through a tableau we mean a sequence of nodes starting from the root to some inner node or leaf node. The same conventions as for branches apply. By the immediate successor (nodes) of partial branch $b$ we mean the set of son nodes of the last node of $b$.

The branch $b = L_1, \ldots, L_n$ is called regular iff $L_i \neq L_j$ for $1 \leq i, j \leq n$ and $i \neq j$; here, equality is understood as equality wrt. the labels (wrt. nodes every branch obviously would be regular). A branch which is not regular is called irregular. The tree $T$ is regular iff every of its branches is regular, otherwise it is irregular. The set of branch literals of $b$ is $\text{lit}(b) = \{L_1, \ldots, L_n\}$. For brevity, we will write expressions like $A \in b$ instead of $A \in \text{lit}(b)$. In order to memorize the fact that a branch contains a contradiction, we allow to label a branch as either open or closed. A tableau is closed if each of its branches is closed, otherwise it is open.

Definition 2.6 (Hyper tableau)

A literal set is called inconsistent iff it contains a pair of complementary literals, otherwise it is called consistent. Hyper tableaux for $D$ are inductively defined as follows:
**Initialization step:** The empty tree, consisting of the root node only, is a hyper tableau for \( D \). Its single branch is marked as “open”.

**Hyper extension step:** If (1) \( T \) is an open hyper tableau for \( D \) with open branch \( b \), and (2) \( C = A_1 \lor \cdots \lor A_m \leftarrow B_1 \land \cdots \land B_n \) is a clause from \( D \) (\( m \geq 0, n \geq 0 \)), called extending clause in this context, and (3) \( \{B_1, \ldots, B_n\} \subseteq b \) (equivalently, we say that \( C \) is applicable to \( b \)) then the tree \( T' \) is a hyper tableau for \( D \), where \( T' \) is obtained from \( T \) by extension of \( b \) by \( C \): replace \( b \) in \( T \) by the new branches

\[
(b, A_1) \ldots, (b, A_m), (b, \neg B_1) \ldots, (b, \neg B_n)
\]

and then mark every inconsistent new branch as “closed”, and the other new branches as “open”.

We say that a branch \( b \) is finished iff it is either closed, or else whenever \( C \) is applicable to \( b \), then extension of \( b \) by \( C \) yields some irregular new branch.

The applicability condition of an extension expresses that all body literals have to be satisfied by the branch to be extended (like in hyper resolution [Rob65]). Unless stated otherwise we will from now on consider only regular hyper tableaux. This restriction guarantees that for finite clause sets no branch can be extended infinitely often. Hence, in particular, no open finished branch can be extended any further. This fact will be made use of below occasionally.

Notice as an immediate consequence of the definition that open branches never contain negative literals.

**Example 2.7 (Hyper Tableaux)**
Consider the following database \( D \):

\[
D : \quad p \lor q \leftarrow t \land r \quad \quad \quad t \leftarrow \\
\quad \quad \quad q \leftarrow p \land t \quad r \leftarrow
\]

The figure on the right contains a hyper tableau for \( D \). For economy of notation, closed branches are not displayed. This tableau is obtained as follows: starting with the empty tree, we can extend with \( t \leftarrow \) and then with \( r \leftarrow \). Then, since \( t \) and \( r \) are now on the branch, we can extend with \( p \lor q \leftarrow t \land r \). The left branch is now finished, because \( q \leftarrow p \land t \) is not applicable. Extension with \( q \leftarrow p \land t \) at the right branch finishes this branch as well.

**Definition 2.8 (Branch Semantics)**
As usual, we represent an interpretation \( \mathcal{I} \) for given domain \( \Sigma \) as the set \( \{ A \in \Sigma \mid \mathcal{I}(A) = \text{true}, A \text{ atom} \} \). Minimality of interpretations is defined via set-inclusion.

Given a tableau with consistent branch \( b \). The branch \( b \) is mapped to the interpretation \( [b]_\Sigma := \text{lit}(b) \). Usually, we write \([b]\) instead of \([b]_\Sigma \) and let \( \Sigma \) be given by the context.
2.2 Hyper Tableaux Calculus

For instance, the semantics of the left (right) branch \( b_1 (b_2) \) in the tableau in Example 2.7 is \([b_1] = \{ t, r, q \}\) \(([b_2] = \{ t, r, p, q \}\).

A refutational completeness result for hyper tableaux was given in [BFN96]. For our purposes of computing database updates, however, we need a (stronger) minimal model completeness result. Further, we are interested in minimal models only with respect to some given subset \( \Gamma \) of the whole signature \( \Sigma \). In the sequel, \( \Gamma \) always denotes some subset of the signature \( \Sigma \), i.e. a subset of the atoms occurring in a clause set under consideration.

**Definition 2.9 (\( \Gamma \)-Minimal Models)**

For any atom set \( M \) define \( M|\Gamma = M \cap \Gamma \). In order to relate atom sets \( M_1 \) and \( M_2 \) define \( M_1 \leq_\Gamma M_2 \) iff \( M_1|\Gamma \subseteq M_2|\Gamma \), and \( M_1 =_\Gamma M_2 \) iff \( M_1|\Gamma = M_2|\Gamma \). As usual, the relation \( M_1 \leq_\Gamma M_2 \) is defined as \( M_1 <_\Gamma M_2 \) or \( M_1 =_\Gamma M_2 \). We say that a model \( I \) for a clause set \( M \) is \( \Gamma \)-minimal (for \( M \)) iff there is no model \( I' \) for \( M \) such that \( I' <_\Gamma I \)

It is easy to see \( \leq_\Gamma \) is a partial order and that \( =_\Gamma \) is an equivalence relation. Notice that the “general” minimal models can simply be expressed by setting \( \Gamma = \Sigma \). Hence, by a minimal model we mean a \( \Sigma \)-minimal one.

Some useful facts are the following:

**Lemma 2 (Useful facts)**

Let \( M_1 \) and \( M_2 \) be atoms sets, and suppose \( \Gamma' \subseteq \Gamma \). Then the following hold:

\[
\begin{align*}
M_1 & \leq_\Gamma M_2 \text{ implies } M_1 \leq_{\Gamma'} M_2 \quad \text{(1)} \\
M_1 & =_\Gamma M_2 \text{ implies } M_1 =_{\Gamma'} M_2 \quad \text{(2)}
\end{align*}
\]

The proof uses simple facts about sets, and it is omitted. Notice that a similar proposition about \( <_\Gamma \) does not hold in general.

We start with a basic lemma. Similar results were given in [Rei87, Nie96a].

**Lemma 3 (Minimization Lemma)**

Suppose \( \Gamma \) is partitioned as \( \Gamma = \Delta \cup \overline{\Delta} \), i.e. \( \Delta \cap \overline{\Delta} = \emptyset \). Let \( M \) be a set of formulas. Then\(^2\)

\[
\begin{align*}
(1a) \ M & \cup \overline{\Delta} \models \Delta \text{ and } (1b) \ M & \cup \overline{\Delta} \text{ is satisfiable} \\
\text{iff } & \quad \text{(2a) } M \cup \overline{\Delta} \cup \Delta \text{ is satisfiable and } (2b) \ \Delta \text{ is } \subseteq_{\text{minimal}} \text{ for this property.}
\end{align*}
\]

Property (2b) using an explicit wording shall mean “there is no partition \( \Gamma = \Delta' \cup \overline{\Delta} \) with \( \Delta' \subset \Delta \) such that \( M \cup \overline{\Delta} \cup \Delta' \) is satisfiable”.

**Proof:** (1) \( \Rightarrow \) (2): Let \( I \) be a model for \( M \cup \overline{\Delta} \), which exists by (1b). By (1a), \( I \) is a model for \( \Delta \) as well. Hence (2a) holds. It remains to show (2b). Suppose, to the contrary, that \( \Delta \) is not a minimal set such that (2a) holds. Hence, there is an

\(^1\)From a circumscriptive point of view, \( \Gamma \) is the set of atoms to be minimized, and \( \Sigma \setminus \Gamma \) varies.

\(^2\)For a set \( \Delta \) of formulas we define \( \neg \Delta := \{ \neg F \mid F \in \Delta \} \). The notation \( M \models \Delta \) means \( M \models F \) for every \( F \in \Delta \).
$A \in \Delta$ such that $M \cup \overline{\Delta} \cup \{\neg A\} \cup \Delta \setminus \{A\}$ is satisfiable. Trivially, $M \cup \overline{\Delta} \cup \{\neg A\}$ is satisfiable as well. But then, $M \cup \overline{\Delta} \not\models A$, and consequently, $M \cup \overline{\Delta} \not\models \Delta$. Contradiction to (1a). Hence the claim follows.

(1) $\iff$ (2) Assume that (2a) and (2b) hold. Hence, (1b) follows trivially. Suppose, to the contrary, that (1a) does not hold. That is, $M \cup \overline{\Delta} \not\models A$, for some $A \in \Delta$. Hence, $X := M \cup \overline{\Delta} \cup \{\neg A\}$ is satisfiable. Let $I$ be a model for $X$. Assume that $I$ is given as the set of true atoms, and atoms not contained in $I$ are false. Hence $I$ implicitly is also an interpretation for $\Delta \setminus \{A\}$. Next define
\[
\Delta' = \{B \in \Delta \setminus \{A\} \mid I \models B\}
\]
\[
\overline{\Delta'} = \overline{\Delta} \cup \{A\} \cup \{B \in \Delta \setminus \{A\} \mid I \models \neg B\}
\]
\[
X' = M \cup \overline{\Delta'} \cup \Delta'.
\]
Since $I \models X$ it follows immediately that $I \models X'$ as well. By construction, we also have $\Gamma = \Delta' \cup \overline{\Delta'}$ and $\Delta' \cap \overline{\Delta'} = \emptyset$. From $\overline{\Delta'} \cap \overline{\Delta}$, we have a contradiction to the minimality of $\Delta$ (2b). Hence the claim follows.

For minimal model reasoning, or, more generally, circumscriptive reasoning, Lemma 3 provides an important “tool” for testing whether an open finished branch $b$ represents a minimal model: $\Gamma$ is taken to be either the signature $\Sigma$ (for general minimal model reasoning), or a subset thereof (for circumscriptive reasoning). The set $\Delta$ is the set of positive atoms occurring on $b$. Now, by Lemma 3, $\Delta$ is a minimal model for $M$ iff $M \cup \overline{\Delta} \models \Delta$ (where $\Delta$ is read as conjunction of atoms). Notice that this condition can be tested by a single theorem prover call, which is referred to as the “groundedness test”, by checking $M \cup \overline{\Delta} \cup \{\bigvee_{A \in \Delta} \neg A\}$ for unsatisfiability. Interested readers are referred to [Nie96b, Nie96c] for more information on this technique of generating minimal models.

The following theorem is a strengthening of a result in [BFFN97a].

**Theorem 4 (Model Soundness and $\Gamma$-Minimal Model Completeness)**

Let $T$ be a hyper tableau for $D$. Then, for every finished open branch $b$ in $T$ it holds that $[b]$ is a (not necessarily $\Gamma$-minimal) model for $D$ (Model soundness).

Further, if every open branch in $T$ is finished, then for every $\Gamma$-minimal model $\mathcal{I}$ of $D$ there is an open branch $b$ in $T$ such that $\mathcal{I} \models [b]$ (Model completeness).

**Proof:**

**Soundness direction:** let $b$ in $T$ be an open finished branch. Suppose, to the contrary, that $[b]$ is not a model for $D$. Hence $[b]$ is not a model for some clause $C = A_1 \lor \cdots \lor A_m \leftarrow B_1 \land \cdots \land B_n$. This means that $\{B_1, \ldots, B_n\} \subseteq \text{lit}(b)$ and $\{A_1, \ldots, A_m\} \cap \text{lit}(b) = \emptyset$. Now, $m = 0$ is impossible, because otherwise by finiteness, $b$ would have to be extended by $C$ and it would have been close then. The case $m > 0$ is handled as follows: since $\{A_1, \ldots, A_m\} \cap b = \emptyset$ the extension of $b$ by $C$ does not violate regularity. Hence, $b$ is not finished. Contradiction.

**Completeness direction:** if no $\Gamma$-minimal model for $D$ exists then the theorem holds vacuously. Otherwise let $\mathcal{I}$ be a $\Gamma$-minimal model for $D$. Let $\Delta \subseteq \Gamma$ the true atoms, i.e. $\Delta = \{A \in \Delta \mid \mathcal{I} \models A\}$, and let $\overline{\Delta} = \Gamma \setminus \Delta$. 

In a first step we show that there is an open branch $b$ such that $\Delta \subseteq \llbracket b \rrbracket$. It trivially holds that
\[ D \cup \Delta \cup \neg \Delta \text{ is satisfiable,} \tag{1} \]
where $\neg M := \{ \neg A \mid A \in M \}$. Since $\mathcal{I}$ is a $\Gamma$-minimal model, Lemma 3 is applicable (in the "if" direction) and we conclude $D \cup \neg \Delta \models \Delta$. This holds if and only if
\[ D \cup \neg \Delta \cup \{ \bigvee_{A \in \Delta} \neg A \} \text{ is unsatisfiable.} \tag{2} \]
Hence, by refutational completeness of Hyper tableaux there is a refutation of this clause set. Further, by 1, the subset $D \cup \neg \Delta$ is satisfiable. Hence, in any hyper tableau refutation the clause $\bigvee_{A \in \Delta} \neg A$ must be used once for an extension step, say at branch $b'$. But, by definition of hyper extension step this is possible only if the complementary literals are on the branch $b'$, i.e. $\Delta \subseteq \text{lit}(b')$. We can omit from the refutation all extension steps with $\bigvee_{A \in \Delta} \neg A$, as well as all extension steps with the negative unit clauses from $\neg \Delta$. The result is a hyper derivation from $D$ alone. Now, either the branch $b'$ is finished, and the theorem is proven, or otherwise the derivation can be continued so that at least one open finished branch $b$ with $\text{lit}(b') \subseteq \text{lit}(b)$ comes up. Reason: otherwise every such extension $b$ of $b'$ would be closed, meaning that we could find a refutation of $D \cup \neg \Delta$ alone, which by soundness of hyper tableau contradicts the satisfiability of $D \cup \neg \Delta$. Thus, $b$ is a branch with $\Delta \subseteq \llbracket b \rrbracket$. This concludes the proof of the first step.

Next, we show that for some branch $b$ with $\Delta \subseteq \llbracket b \rrbracket$ we have $\overline{\Delta} \cap \llbracket b \rrbracket = \emptyset$. Suppose, to the contrary, that for every branch $b$ with $\Delta \subseteq \llbracket b \rrbracket$ we have $\overline{\Delta} \cap \llbracket b \rrbracket \neq \emptyset$. \hfill \Box

For example, since in the tableau in Example 2.7 every open branch is finished, one of its branches contains a $\Sigma$-minimal model (the literals $\{ t, r, q \}$ in the left branch constitute a $\Sigma$-minimal model). The $\{ t, r, q \}$-minimal models are $\{ t, r, q \}$ and $\{ t, r, p, q \}$, which both are computed.

The just presented calculus of hyper tableau has been adopted in [BFFN97b, BFFN97a] for model based diagnosis applications (cf. [Rei87]). Further, a semantical approach (by using an "initial model" of the correctly functioning device for transforming the given system description and observation) was used to guide building models. This transformation technique can be successfully used for database updates also, and in the sequel we discuss this in detail.

3 An Algorithm for View Deletion

A definite deductive database $DDB$ consists of two parts: an intensional database $IDB$, a set of definite program clauses; and an extensional database $EDB$, a set of
ground facts. The intuitive meaning of DDB is provided by the Least Herbrand model semantics and all the inferences are carried out through SLD-derivation. The reader is referred to [Llo87, and the references therein], for more information on definite programs, the least Herbrand model semantics, and SLD-derivations. All the predicates that are defined in IDB are referred to as view predicates and those defined in EDB are referred to as base predicates. Extending this notion, an atom with a view predicate is said to be a view atom, and similarly an atom with base predicate is a base atom. Further we assume that IDB does not contain any unit clauses and no predicate defined in a given DDB is both view and base.

Two kinds of view updates can be carried out on a DDB: An atom, that does not currently follow from DDB, can be inserted; or an atom, that currently follows from DDB, can be deleted. In this paper, we consider only deletion of an atom from a DDB. When an atom $A$ is to be deleted, the view update problem is to delete only some relevant EDB facts, so that the modified EDB together with IDB will satisfy the deletion of $A$ from DDB. View update problem, in the context of deductive databases, has been studied by various authors and algorithms based on SLD-trees have been proposed [AD95, Ara95, Bry90, Dec90, Dec96, GL90, GL91, KM90, Tom88, for example].

Note that a DDB can be considered as a knowledge base to be revised. The IDB is the immutable part of the knowledge base, while the EDB forms the updatable part. In general, it is assumed that the language underlying a DDB is fixed and the semantics of DDB is the least Herbrand model over this fixed language. We assume that there are no function symbols implying that the Herbrand Base is finite. Therefore, the IDB is practically a shorthand of its ground instantiation\(^3\), written as IDBG. In the sequel, technically we mean IDBG when we refer simply to IDB. Thus, a DDB represents a knowledge base where the immutable part is given by IDBG and updatable part is the EDB. Hence, the rationality postulates (KB—1), (KB—2), (KB—3), (KB—4), (KB—5), and (KB—6.3) provide an axiomatic characterization for deleting a view atom $A$ from a definite database DDB.

Logic can provide a conceptual level understanding of relational databases, and hence rationality postulates (KB—1), (KB—2), (KB—3), (KB—4), (KB—5), and (KB—6.3) can provide an axiomatic characterization for view deletion in relational databases too. A relational database together with its view definitions can be represented by a definite deductive database (EDB representing tuples in the database and IDB representing the view definitions), and so the same algorithm can be used to delete view extensions from relational and deductive databases. It is then interesting to compare our approach with existing algorithms [DB82, Kel85, Lan90, for example].

An algorithm for view deletion, based on the general contraction algorithm (cf. Algorithm 1) was presented in [AD95, Ara95]. There, given a view atom to be deleted, set of all explanations for that atom has to be generated through a complete SLD-tree and a hitting set of these explanations is then deleted from the EDB. It

\(^3\) a ground instantiation of a definite program $P$ is the set of clauses obtained by substituting terms in the Herbrand Universe for variables in $P$ in all possible ways.
was shown that this algorithm is rational. A serious drawback of this algorithm is 
that all explanations for the view atom to be deleted have to be generated and kept 
in memory (recall that a complete SLD-tree has to be generated). This means that 
this algorithm is of exponential space complexity. The same analysis holds for other 
known rational algorithms such as that of Tomasic [Tom88].

In this paper, we present a radically different approach that runs on polynomial 
space. In contrast to our previous algorithm, this one directly computes a hitting set 
without explicitly generating all the explanations. Moreover, the generation of hitting 
set is carried out through a hyper tableaux calculus that is focussed on the goal.

3.1 An approach based on minimality test

The key idea of the algorithm presented in this paper is to transform the given 
database along with the view deletion request into a disjunctive logic program and 
apply known disjunctive techniques to solve the original view deletion problem. The 
intuition behind the transformation is to obtain a disjunctive logic program in such 
a way that each (minimal) model of this transformed program represent a way of 
deleting the given view atom. We present two variants of our algorithm. The one 
that is discussed in this section employs a trivial transformation procedure but has 
to look for \( \Gamma \)-minimal models, where \( \Gamma \) will consist of the negations of the EDB- 
predicates; \( \Gamma \)-minimal models therefore characterize minimal deletion operations by 
reading the truth of a negative EDB-literal as a deletion.

The other variant (discussed in the next section) performs a costly transformation, 
but dispenses with the requirement of computing the minimal models.

We start presenting our algorithm by first defining precisely how the given database 
is transformed into a disjunctive logic program for view deletion purposes.

Definition 3.1 (Renaming)

Given a clause \( C = A \leftarrow B \) where \( A \) (\( B \)) is a disjunction (conjunction) of atoms and 
a set of ground atoms \( S \subseteq \Sigma \). The renaming of \( C \) wrt \( S \) is

\[
C^S = (\bigvee_{A \in A, A \notin S} A) \lor (\bigvee_{B \in B, B \notin S} \neg B) \leftarrow (\bigvee_{B \in B, B \notin S} B) \land (\bigvee_{A \in A, A \notin S} \neg A)
\]

For a clause set, the renaming is defined as the renaming of all its members.

That is, for renaming a clause every atom \( A \) in the body (resp. head) of \( C \) that is 
also in \( S \) is moved to the head (resp. body) as \( \neg A \). Below we will make make use of 
the trivial fact that \( I \models C \) iff \( I \models C^S \).

The careful reader will notice that the renaming does not result in a clause, 
because it leaves us with literals where atoms are expected. However, we will take 
the freedom to refer to a renaming as a clause as well.

We want to apply the hyper tableau calculus to renamed clause sets as well. 
In order to avoid unnecessary changes to the calculus, we can bijectively map a 
renamed clause set to a clause set according to the original definition by taking the 
signature \( \Sigma^S = \{ A \in \Sigma \mid A \notin S \} \cup \{ \neg A \mid A \in S \} \). The second set is to be read 
as a set of atoms containing in their names the negation sign. Henceforth, when
the hyper tableaux calculus is applied to a renamed clause set, always the modified signature $\Sigma^S$ is understood. Hence, also the interpretation $[b]$ associated to a branch $b$ in such a tableau is a model for the renamed clause set, say $M^S$. It can easily be converted into a model for the original clause set $M$ by the following bijection between $\Sigma$-interpretations and $\Sigma^S$-interpretation: for every interpretation $\mathcal{I}$: $\mathcal{I} \models M$ iff $\mathcal{I}^S \models M^S$, where $\mathcal{I}^S(\neg A) = \text{true}$ iff $\mathcal{I}(A) = \text{false}$ for $A \in S$, and $\mathcal{I}^S(A) = \mathcal{I}(A)$ for $A \notin S$.

For instance, take $M = \{A \lor \neg B \lor C\}$, $S = \{A, B\}$, $\mathcal{I} = \{\}$. Hence $M^S = \{-A \lor \neg B \lor C\}$ and $\mathcal{I}^S = \{-A, \neg B\}$ is a model for $M^S$. This example also demonstrates that minimality of models is not preserved\(^4\) Instead the following can be achieved:

**Lemma 5 (Maximal Models by Renaming)**

Let $S$ be a set of atoms such that $\Gamma \subseteq S$, and let $M^S$ be the renaming of satisfiable clause set $M$ wrt. $S$. Let

\[
\begin{align*}
\Delta &= \{A \in \Gamma \mid \mathcal{I}(A) = \text{true}\} \\
\Delta^S &= \{-A \in \Gamma^S \mid \mathcal{I}^S(\neg A) = \text{true}\} \\
&= \{-A \in \Gamma^S \mid \mathcal{I}(A) = \text{false}\}.
\end{align*}
\]

Then a $\Sigma^S$-interpretation $\mathcal{I}^S$ is a $\Gamma^S$-minimal model for $M^S \cup \Delta^S$ iff

$\mathcal{I} \models M \cup \Delta$ and there is no set $\Delta'$ with $\Delta \subset \Delta' \subseteq \Gamma$ such that $M \cup \Delta'$ is satisfiable.

That is, under the stated condition, computing $\Gamma^S$-minimal models is equivalent to maximize a set of atoms $\Delta \subseteq \Gamma$ consistent with $M$. Intentionally $\Delta$ constitutes exactly the true atoms of $\mathcal{I}$ restricted to $\Gamma$. Hence, $\mathcal{I}$ is a “\(\Gamma\)-maximal” model.

**Proof:** The equivalence stated in parenthesis follows from the definition of $\mathcal{I}^S$ and the fact that $\Gamma \subseteq S$. We have

$\mathcal{I}^S$ is a $\Gamma^S$-minimal model for $M^S \cup \Delta^S$

$\iff \mathcal{I}^S \models M^S \cup \{-A \in \Gamma^S \mid \mathcal{I}^S(\neg A) = \text{true}\} \cup \{-\neg A \mid -A \in \Gamma^S \text{ and } \mathcal{I}^S(\neg A) = \text{false}\}$ and the second set is minimal, and hence the third set is maximal

$\iff \mathcal{I} \models M \cup \{-A \in \Gamma \mid \mathcal{I}(A) = \text{false}\} \cup \{A \mid A \in \Gamma \text{ and } \mathcal{I}(A) = \text{true}\}$ and the second set is minimal, and hence the third set is maximal (this follows immediately from the definition of $\mathcal{I}^S$)

$\iff \mathcal{I} \models M \cup \Delta$ as claimed

\[\square\]

**Definition 3.2 \((IDB^*)\) Transformation**

Let $IDB \cup EDB$ be a given database. Let $S_0 = EDB \cup \{A \mid A \text{ is a ground IDB atom}\}$. Then, $IDB^*$ is defined as the renaming of $IDB$ wrt $S_0$.

\[\square\]

\(^4\)One would have to have that $\Gamma \cap S = \emptyset$ to achieve that $\Gamma$-minimality of models is preserved. In the present paper, however, we do not need this result.
3.1 An approach based on minimality test

Remarks 3.3
Note that $IDB^*$ is in general a disjunctive logic program. The negative literals ($\neg A$) appearing in the clauses are intuitively interpreted as deletion of the corresponding atom ($A$) from the database.

Note that there are no facts in $IDB^*$. So when we add a delete request such as $\neg A$ to this, the added request is the only fact and any bottom-up reasoning strategy is fully focused on the goal (here the delete request).

The following example illustrates this transformation idea.

Example 3.4
Consider the following database:

$$IDB : \begin{align*}
p & \leftarrow t \\
p & \leftarrow q \land u \\
q & \leftarrow s \\
u & \leftarrow r
\end{align*}$$

$$EDB : \begin{align*}
t & \leftarrow \\
r & \leftarrow \\
\end{align*}$$

The set $S_0$ is determined by all the $IDB$ atoms and the current $EDB$ atoms and in our case it is \{p, q, u, t, r\}. $IDB^*$ is the transformation of $IDB$ wrt $S_0$ which is given as follows:

$$IDB^* : \begin{align*}
\neg t & \leftarrow \neg p \\
\neg q \lor \neg u & \leftarrow \neg p \\
\ & \leftarrow s \land \neg q \\
\neg r & \leftarrow \neg u
\end{align*}$$

Now, when we have a deletion request for a ground view atom $A$, represented as $\neg A$, the idea is to generate models of $IDB^* \cup \{\neg A\}$ and read the base atoms to be deleted from them. As mentioned in the above remark, $\neg A$ is the only fact and a bottom-up model generation process is fully goal-oriented. We propose to use the hyper tableaux calculus for this, and we state precisely how this is done. Suppose a ground view atom $A$ is to be deleted. Then, a hyper tableau for $IDB^*$ with delete request $\neg A$ is built. The open finished branches give us models for the renamed database. The intuition is that the set of $EDB$ atoms appearing in a model (open branch) constitute a hitting set, and removing this set from $EDB$ should achieve the required view deletion. This is formalized below.

Definition 3.5 (Update Tableaux, Hitting Set)
An update tableau for a database $IDB \cup EDB$ and delete request $\neg A$ is a hyper tableau $T$ for $IDB^* \cup \{\neg A \leftarrow \}$ such that every open branch is finished. For every open finished branch $b$ in $T$ we define the hitting set (of $b$ in $T$) as $HS(b) = \{A \in EDB \mid \neg A \in b\}$.

Remarks 3.6 (Hitting Set and Relation to Diagnosis)
The name "hitting set" is a misnomer here, but we use it in order to compare this approach with previous approaches that generate explanations and a hitting set of
them. This new approach directly generates a “hitting set” without enumerating all the explanations. Also, what we call hitting set here, has been called diagnosis in [BFFN97b, BFFN97a]. In terms of diagnosis concepts from [Rei87], the system description can be considered as IDB, while the non-abnormality of the components form the corresponding EDB. Technically, this means to replace every EDB-atom \( p \) by a clause \( p \leftarrow \neg ab(p) \), where “\( ab(p) \)” means “\( p \) is abnormal”. In order to have a one-to-one mapping between minimal diagnosis (cf. [Rei87]) and minimal database contractions we need the additional axioms \( \neg p \leftarrow ab(p) \) for every EDB-atom \( p \). Together, they mean that a literal \( p \) is abnormal (diagnosis) if and only if \( p \) is deleted from the database.

Axioms of the latter kind are a bit problematic in the approach of [BFFN97a] due to the negative \( ab \)-literals, which were not allowed there. However, in any minimal diagnosis these axioms are a consequence of the other clauses by the following line of reasoning: let a minimal diagnosis be given (i.e. a model which minimizes the extension of the \( ab \)-predicate. Say that \( ab(p) \) holds in that diagnosis. Due to \( p \leftarrow \neg ab(p) \), which is the sole clause containing \( \neg ab(p) \), this can only be the case if \( p \) is false. Hence \( \neg p \leftarrow ab(p) \) holds in this diagnosis.

But now, since we know that \( \neg p \leftarrow ab(p) \) holds in every minimal diagnosis, these clauses can be deleted, and the approach of [BFFN97a] can be used to compute minimal database contractions.

Next we want to assemble all the concepts and results obtained so far in the following theorem:

**Theorem 6 (Completeness of Update tableaux)**

Let \( T \) be a non-trivial update tableau for \( IDB \cup EDB \) and delete request \( \neg A \). Then for every minimal set \( \alpha \subseteq EDB \) such that \( \{\neg A\} \cup IDB \cup (EDB \setminus \alpha) \) is satisfiable there is an open finished branch \( b \) in \( T \) such that \( HS(b) = \alpha \).

**Proof:**

We have

\[ \{\neg A\} \cup IDB \cup (EDB \setminus \alpha) \text{ is satisfiable, } \alpha \text{ minimal} \]

**Lemma 5**

For some \( I : I \models \{\neg A\} \cup IDB \cup (EDB \setminus \alpha) \), \( \alpha \text{ minimal} \)

Reason: Recall from Definition 3.2 that \( EDB \subseteq S_0 \); hence Lemma 5 is applicable, setting \( \Delta = (EDB \setminus \alpha), \Delta^S = \neg \alpha \) and \( M = \{\neg A\} \cup IDB \)

there

**Theorem 4**

for some \( I^S_0 : I^S_0 \) is a \( EDB^S_0 \)-minimal model for \( \{\neg A\} \cup IDB^S \)

Reason: every smaller model wrt. \( EDB^S_0 \) would falsify one atom from \( \neg \alpha \), thus contradicting the previous line.

there is an open finished branch \( b \) in \( T \) such that \( [b] =_{EDB^S_0} I^S_0 \). Further, \( I^S_0 =_{EDB^S_0} \neg \alpha \), because \( \neg \alpha = \Delta^S = \{\neg A \in EDB^S_0 \mid I^S_0(\neg A) = \text{true} \)

there is an open finished branch \( b \) in \( T \) such that \( HS(b) = \alpha \)

\( \blacksquare \)
To sum up, we have arrived at a completeness result which allows us to compute minimal contractions by computing one (any) update tableaux. However, the converse is not established yet. That is, given an open finished branch \( b \), it is not guaranteed that \( HS(b) \) is a minimal database contraction. In other words, we will have to establish soundness.

Before discussing the soundness of the intuition given above in this section, we wish to eliminate a trivial form of update tableau:

**Definition 3.7 (Trivial Update Tableaux)**
Let \( T \) be an update tableau. The branch \( b \) in \( T \) is called trivial if \( HS(b) = \emptyset \), otherwise it is called non-trivial. An update tableau is called trivial if some open finished branch is trivial, otherwise it is called non-trivial.

**Example 3.8**
A special case comes up if an update tableau is trivial. For example, if \( IDB = \{ p \leftarrow q \} \) and \( EDB = \{ r \leftarrow \} \) then \( IDB^* = \{ \leftarrow \neg p \land \neg q \} \). The update tableau for \( IDB \cup EDB \) and delete request \( \neg p \) consists of one open branch \( b \), which is labelled with \( \neg p \) (because \( \leftarrow \neg p \land \neg q \) is not applicable to \( \neg p \)). Thus, \( HS(b) = \emptyset \). Intentionally, this means that the delete request can be fullfilled without deleting any \( EDB \)-atoms.

Note that there is no point in continuing an update tableau construction as soon as one open finished trivial branch is derived. This is always an “optimal” case, because it means that the delete request ist compatible to the \( EDB \). This is formalized below.

**Lemma 7 (Trivial update tableaux)**
Let \( T \) be a trivial update tableau for \( IDB \cup EDB \) and delete request \( \neg A \). Then \( IDB \cup EDB \cup \{ \neg A \} \) is consistent.

If \( IDB \cup EDB \cup \{ \neg A \} \) is consistent then every update tableau for \( IDB \cup EDB \) and delete request \( \neg A \) is trivial.

**Proof:** For the first part, let \( T \) be given as stated, and assume, to the contrary, that \( IDB \cup EDB \cup \{ \neg A \} \) is inconsistent. As a property of the renaming, it holds that \( IDB^* \cup EDB^* \cup \{ \neg A \leftarrow \} \) is inconsistent, where \( EDB^* = \{ \leftarrow \neg A \mid A \in EDB \} \). Further, since \( IDB \cup \{ \neg A \} \) is consistent (for syntactical reasons: \( IDB \) contains no facts), \( IDB^* \cup \{ \neg A \leftarrow \} \) is consistent as well. Notice that by definition of update tableau, \( T \) is a hyper tableau for just this set. By soundness of hyper tableaux (Theorem 4), \( T \) must contain open branches. Further, every open branch must close with some clause from \( EDB^* \) (because otherwise, by completeness of hyper tableaux (Theorem 4), we would have a contradiction to the inconsistency of \( IDB^* \cup EDB^* \cup \{ \neg A \leftarrow \} \)). But this means that every open branch contains at least one literal from \( \{ \neg A \leftarrow \mid A \in EDB \} \). But then \( HS(b) \neq \emptyset \) for every open branch \( b \) in \( T \). Consequently, \( T \) is not trivial. Contradiction.

The argumentation for the second part is similar. We will therefore only sketch the proof. Assume, to the contrary, that \( IDB \cup EDB \cup \{ \neg A \} \) is consistent, and that there is a respective non-trivial update tableaux. Let \( T \) be that tableau. It is a tableau for the clause set \( IDB^* \cup \{ \neg A \leftarrow \} \). It must be open, because \( IDB \cup \{ \neg A \} \) is consistent, and consequently \( IDB^* \cup \{ \neg A \leftarrow \} \) is consistent as well. Now, since \( T \)
is non-trivial, every open branch \( b \) in \( T \) contains a literal from \( \{ \neg A \leftarrow \mid A \in EDB \} \). Thus, \( b \) can be closed with some clause from \( EDB^* \) (cf. the first part for a definition of \( EDB^* \)). From this we learn that \( IDB^* \cup EDB^* \cup \{ \neg A \leftarrow \} \) is inconsistent. As a property of the renaming we conclude that \( IDB \cup EDB \cup \{ \neg A \} \) is consistent as well. Contradiction.

In this trivial case, we can establish the following link to the abductive explanations of atom to be deleted:

**Lemma 8**

\( IDB \cup EDB \cup \{ \neg A \} \) is consistent iff there is no \( EDB \)-closed abductive explanation for \( A \) wrt. \( IDB \).

**Proof:** “Only-if”-direction: assume, to the contrary, that \( IDB \cup EDB \cup \{ \neg A \} \) is consistent, and that there is an \( EDB \)-closed abductive explanation for \( A \) wrt. \( IDB \). Thus let \( \Delta \subseteq EDB \) such that \( \Delta \cup IDB \models A \). Equivalently, \( \Delta \cup IDB \cup \{ \neg A \} \) is inconsistent. Thus \( EDB \cup IDB \cup \{ \neg A \} \) is inconsistent. Contradiction.

“If”-direction: assume, to the contrary, that there is no \( EDB \)-closed abductive explanation for \( A \) wrt. \( IDB \) and that \( IDB \cup EDB \cup \{ \neg A \} \) is inconsistent. Equivalently, \( IDB \cup EDB \models A \). Since \( IDB \cup EDB \) is consistent (for syntactical reasons), \( \Delta := EDB \) thus is an abductive explanation. Contradiction.

Now let us turn our attention to the vital question of correctness and rationality of our proposal in the non-trivial case. Unfortunately, reading off updates from models of the transformed program does not result in a rational deletion, as relevance policy may be violated.

**Example 3.9**

Let us continue with example 3.4. Suppose the view atom \( p \) is to be deleted. Then according to the above proposal, an update tableau for \( IDB^* \) and \( \neg p \) is to be built. This is illustrated in the accompanying figure.

As shown, two open branches constitute two hitting sets \( \{ t \} \) and \( \{ t, r \} \). It is not difficult to see that \( \{ t, r \} \) does not satisfy any of the relevance policies (KB—6.1) or (KB—6.2) or (KB—6.3). Hence simple model computation using hyper tableau calculus does not result in rational hitting sets.

So, it is clear that every model of the transformed program does not constitute a rational deletion. This problem could be rectified in two different ways. In this section, we discuss an approach that employs a test for every branch. Yet another approach based on materialized view is presented in the next section.
To filter out only the rational hitting sets, the postulate (KB—6.1) can be used as a test! That is, after constructing a branch, the minimality condition of (KB—6.1) can be checked (which is a theorem proving task). The branch is closed if the corresponding hitting set does not satisfy this strong relevance postulate.

**Definition 3.10 (Minimality Test)**

Let $T$ be an update tableau for $IDB \cup EDB$ and delete request $\neg A$. We say that open finished branch $b$ in $T$ satisfies the strong minimality test iff $\forall s \in HS(b) : IDB \cup EDB \setminus HS(b) \cup \{s\} \vdash A$.

**Definition 3.11 (Update Tableau Satisfying Strong Minimality)**

An update tableau for given $IDB \cup EDB$ and delete request $\neg A$ is transformed into an update tableau satisfying strong minimality by marking every open finished branch as closed which does not satisfy strong minimality.

**Example 3.12**

Continuing with the same example, after constructing the branch corresponding to the hitting set $\{t, r\}$, the strong minimality test is carried out as follows: It is checked if the resulting database with each member of hitting set implies the deleted atom $p$ or not. For example, $IDB \cup EDB \setminus \{t, r\} \cup \{t\} \vdash p$. But the same does not hold for $r$, i.e. $IDB \cup EDB \setminus \{t, r\} \cup \{r\} \not\vdash p$, and hence this branch fails the strong minimality test.

Interestingly, this minimality test is equivalent to the groundedness test used by Ilkka Niemelä for generating minimal models of disjunctive logic programs [Nie96b, Nie96c]. The key idea of the groundedness test is to check if the members in the model are implied by the program together with the negation of the atoms not present in the model.

In the following we will formulate this groundedness test technique formally and establish the equivalence results to the minimality test technique.

**Definition 3.13 (Groundedness Test)**

Let $T$ be an update tableau for $IDB \cup EDB$ and delete request $\neg A$. We say that open finished branch $b$ in $T$ satisfies the groundedness test iff $\forall s \in HS(b) : IDB^* \cup \{\neg B \mid B \in EDB \setminus HS(b)\} \cup \{\neg A \leftarrow \} \vdash \neg s$.

A notable difference between the groundedness test and the minimality test is that the minimality test is carried out wrt. the original database $IDB$, whereas the groundedness test is carried out wrt. the renamed database $IDB^*$. We will show that the minimal models computed by using the groundedness test achieve a minimization of the deletion of $EDB$-atoms such that consistency with a given delete request is recovered.

**Proposition 9**

Let $T$ be a hyper tableau for $IDB^* \cup \{\neg A \leftarrow \}$ and $b$ be an open finished branch in $T$. Then the following are equivalent:
(a) \( b \) satisfies the groundedness test

(b) \( b \) satisfies the strong minimality test

(c) \([b]\) is an \( EDB^*\)-minimal model for \( IDB^* \cup \{ \neg A \} \), where \( EDB^* = \{ \neg C \mid C \in EDB \} \)

This means that every \( EDB^*\)-minimal model of \( IDB^* \cup \{ \neg A \} \) provides a minimal hitting set for deleting the ground view atom \( A \). Candidate branches can be checked for this property by the groundedness test.

**Proof:** (a) iff (b): We have

Branch \( b \) satisfies the groundedness test

\[ \iff IDB^* \cup \{ \leftarrow B \mid B \in EDB \backslash HS(b) \} \cup \{ \neg A \leftarrow \} \vdash \{ \neg s \mid s \in HS(b) \} \quad (*) \]

\[ \iff IDB \cup \{ B \leftarrow \mid B \in EDB \backslash HS(b) \} \cup \{ \leftarrow A \} \vdash \{ \leftarrow s \mid s \in HS(b) \} \]

Reason: by property of renaming: for any pair of interpretations \( I^S \) for the previous line and \( I \) for this line: a clause occurring in the previous line (either in the premise or conclusion) is true in \( I^S \) iff the corresponding clause in this line is true in \( I \).

\[ \iff \forall s \in HS(b) : IDB \cup EDB \backslash HS(b) \cup \{ s \} \vdash A \]

\[ \iff b \] satisfies the strong minimality test.

(a) iff (c): we want to apply Lemma 3 to conclude that line (*) holds iff

\[ IDB^* \cup \{ \leftarrow B \mid B \in EDB \backslash HS(b) \} \cup \{ \neg A \leftarrow \} \cup \{ \neg s \leftarrow \mid s \in HS(b) \} \]

is satisfiable, and \( HS(b) \) is minimal. Since this is only slight reformulation of (c) the claim would be proven.

Hence we show by the following line of reasoning that Lemma 3 is applicable: we set there \( M = IDB^* \cup \{ \neg A \leftarrow \} \), \( \Gamma = \{ \neg C \mid C \in EDB \} \), \( \Delta = \{ \neg s \mid s \in HS(b) \} \) and \( \Sigma = \{ \neg B \mid B \in EDB \backslash HS(b) \} \). In order to apply Lemma 3 we have to show that \( \tilde{N} = IDB^* \cup \{ \neg B \mid B \in EDB \backslash HS(b) \} \cup \{ \neg A \leftarrow \} \) is satisfiable: clearly, \( IDB^* \cup \{ \neg A \leftarrow \} \) is satisfiable, because we are given a tableaux with open saturated branch \( b \) for it. No clause \( \leftarrow B \) such that \( B \in EDB \backslash HS(b) \) can be used to close \( b \), because then we would have to have \( \neg B \in b \), which implies \( B \in HS(b) \). But \( B \) cannot be in both \( EDB \backslash HS(b) \) and \( HS(b) \). Thus, \( b \) remains an open branch when \( \{ \leftarrow \neg B \mid B \in EDB \backslash HS(b) \} \) would be added to \( IDB^* \cup \{ \neg A \leftarrow \} \). Further, \( b \) remains finished, because adding negative clauses does not enable other extension steps than those closing branches. Thus, together, \( \tilde{N} \) is satisfiable, because \( b \) constitutes a model for it.

Now we are in a position to formally present our algorithm. Given a database and a view atom to be deleted, we first transform the database into a disjunctive logic program and use hyper tableaux calculus to generate models of this transformed program. Models that do not represent rational deletions are filtered out using the strong minimality test. This is formalized in Algorithm 2.

To show the correctness of this algorithm (wrt the rationality analysis), we could study if all the rationality postulates are satisfied. It is obvious that (KB—6.1) is satisfied and checking other postulates should not be difficult. The completeness of this algorithm (wrt the rationality analysis) could be studied following the lines of
Algorithm 2 View deletion algorithm based on minimality test

Input: A database $IDB \cup EDB$ and a ground view atom $A$ to be deleted.

Output: A new database $IDB \cup EDB'$

begin

1. Construct a branch $b$ of an update tableau satisfying strong minimality (cf. Definition 3.11) for $IDB \cup EDB$ and delete request $\neg A$.

2. Produce $IDB \cup EDB\setminus HS(b)$ as a result. ($HS(b)$ of a branch $b$ of an update tableau is defined in Definition 3.5)

end.

[BFFN97a], where a similar exercise was carried out for minimal diagnosis completeness.

However, we choose an alternative way to show the rationality of this approach in order to gain more insight. In particular, we analyse how this is related to the previous approach presented in the last section, i.e. generating explanations and computing hitting sets of these explanations. To better understand the relationship it is imperative to find where the explanations are in the hyper tableau approach. We first define the notion of cut in this direction.

Definition 3.14 ($EDB$-cut)
Let $T$ be an update tableau with open branches $b_1, \ldots, b_n$. A set $S = \{A_1, \ldots, A_n\} \subseteq EDB$ is said to be a $EDB$-cut of $T$ iff $\neg A_i \in b_i$, for $1 \leq i \leq n$.

That is, an $EDB$-cut is obtained from $T$ by picking exactly one negated $EDB$-atom from each open branch in $T$. Notice that for non-trivial update tableau an $EDB$-cut always exist, as every open branch contains a negated $EDB$-atom.

A careful reader would have already realized that an $EDB$-cut across the tableau constitutes an explanation for the view atom being deleted. More precisely:

Lemma 10
Let $T$ be a non-trivial update tableau for $IDB \cup EDB$ and delete request $\neg A$. Let $S$ be the set of all $EDB$-closed minimal abductive explanations for $A$ wrt. $IDB$. Let $S'$ be the set of all $EDB$-cuts of $T$. Then the following hold:

- $S \subseteq S'$
- $\forall \Delta' \in S' : \exists \Delta \in S \text{ s.t. } \Delta \subseteq \Delta'$

Proof: For the first item, let $EDB \supseteq \Delta \in S$ be a given minimal abductive explanation for $A$. We have to show that there is an $EDB$-cut of $T$ which consists of the same literals as $\Delta$. First, note that $IDB \cup \Delta \cup \{\neg A\}$ is inconsistent, and that $IDB \cup \Delta' \cup \{\neg A\}$ is consistent, for each $\Delta' \subset \Delta$ (*). Since the renaming transformation preserves satisfiability, it holds that $IDB* \cup \Delta* \cup \{\neg A \leftarrow \}$ is inconsistent as well, where $\Delta* := \{ \leftarrow \neg A | A \in \Delta \}$. Further, since $IDB \cup \{\neg A\}$ is consistent (for
syntactical reasons: \( IDB \) contains no facts), \( IDB^* \cup \{ \neg A \leftarrow \} \) is consistent as well. Notice that by definition of update facts, \( T \) is a hyper tableau for just this set.

By soundness of hyper tableaux (Theorem 4), \( T \) must contain open branches. Further, every open branch \( b_i \) in \( T \), for \( 1 \leq i \leq n \), must close with some (negative unit) clause of the form \( \neg A_i \in \Delta^* \). This holds, because otherwise, by completeness of hyper tableaux (Theorem 4), we would have a contradiction to the inconsistency of \( IDB^* \cup \Delta^* \cup \{ \neg A \leftarrow \} \). Furthermore, each of the negative unit clauses in \( \Delta^* \) must be used (at least once) to close a branch, because otherwise we would have together with soundness of hyper tableaux a contradiction to the minimality of \( \Delta \), as stated in (*) Thus, in other words, collecting the respective \( EDB \)-atoms from the \( b_i \)'s gives the \( EDB \)-cut \( \Delta \) of \( T \).

The argumentation for the second item is similar. Let \( \Delta' \in S' \) be arbitrarily. By definition of \( EDB \)-cut, for each \( A \in \Delta' \) there is an open branch in \( T \) containing \( \neg A \), and vice versa. Hence, the additional negative unit clauses \( \Delta^* := \{ \neg A \mid A \in \Delta^* \} \) can be used to close \( T \). Recall that \( T \) is a hyper tableau for the clause set \( IDB^* \cup \{ \neg A \leftarrow \} \). Thus, by soundness of Hyper tableaux, \( IDB^* \cup \Delta^* \cup \{ \neg A \leftarrow \} \) is inconsistent. Renaming does not affect consistency. Hence \( IDB \cup \Delta' \cup \{ \neg A \} \) is inconsistent as well. Equivalently, \( IDB \cup \Delta' \models A \). Since \( IDB \cup \Delta' \) is consistent for syntactical reasons (this set contains no negative clauses), \( \Delta' \) is an \( (EDB \)-closed\) abductive explanation for \( A \). Trivially \( \Delta' \) contains a minimal such explanation \( \Delta \). That is, \( \Delta \in S \) as desired.

\[ \textbf{Remarks 3.15} \]

Unfortunately, and contrary to what one might expect, it is not possible to strengthen the second item in this lemma towards

\[ \forall \Delta' \in S' : \exists \Delta \in S \text{ s.t. } \Delta = \Delta' , \]

even if we restrict to update tableaux satisfying strong minimality. This is the canonical counterexample:

\[
\begin{align*}
IDB: & \quad p \leftarrow q \land r \\
p & \leftarrow t \\
EDB: & \quad q \leftarrow r \\
r & \leftarrow t \\
t & \leftarrow
\end{align*}
\]

Now consider \( p \). There are two minimal abductive explanations for \( p \), which are \( \{ t \} \) and \( \{ q, r \} \). An update tableau (even satisfying strong minimality), however, will also admit the \( EDB \)-cuts \( \{ q, t \} \) and \( \{ r, t \} \). This example shows us that computing minimal hitting sets and computing minimal abductive explanations are rather different things.

The above lemma precisely characterizes what explanations are generated by an update tableau. It is obvious then that a branch cuts through all the explanations and constitutes a hitting set for all the generated explanations. This is formalized below.
Lemma 11 ([AD95, Ara95])

Let $S$ and $S'$ be sets of sets s.t. $S \subseteq S'$ and every member of $S' \setminus S$ contains an element of $S$. Then, a set $H$ is a minimal hitting set for $S$ iff it is a minimal hitting set for $S'$.

Lemma 12

Let $T$ be a non-trivial update tableau for $IDB \cup EDB$ and a delete request $\neg A$ that satisfies the strong minimality test. Then, for every open finished branch $b$ in $T$, $HS(b)$ is a minimal hitting set for all the abductive explanations of $A$.

Proof: Follows from Lemma 10 and Lemma 11.

So, Algorithm 2 generates a minimal hitting set (in polynomial space) of all $EDB$-closed locally minimal abductive explanations of the view atom to be deleted. From the belief dynamics results recalled in section 2, it immediately follows that Algorithm 2 is rational.

Theorem 13 (Main Theorem)

Algorithm 2 is rational, in the sense that it satisfies all the rationality postulates (KB—1), (KB—2), (KB—3), (KB—4), (KB—5), and the strong relevance postulate (KB—6.1). Further, any deletion that satisfies these postulates can be computed by this algorithm.

3.2 An approach based on materialized view

In many real world applications, the view is materialized, i.e. the least Herbrand Model is computed and kept, for efficient query answering. In such a situation, rational hitting sets (i.e. deletions satisfying weak relevance) can be computed without performing any minimality test. The idea is to transform the given $IDB$ wrt the materialized view and to make a little change to the calculus. We will first describe the new transformation, and then motivate why the calculus has to be changed as well.

Definition 3.16 ($IDB^+$ Transformation)

Let $IDB \cup EDB$ be a given database. Let $S_1$ be the Least Herbrand Model of this database. Then, $IDB^+$ is defined as the transformation of $IDB$ wrt $S_1$.

Example 3.17

Consider Example 3.4 again. It is not difficult to see that the Least Herbrand model of the database $IDB \cup EDB$ given there is $S_1 = \{t, p, r, u\}$. Hence:

$$IDB^+ : \begin{align*}
- t & \leftarrow - p \\
- u & \leftarrow - p \land q \\
q & \leftarrow s \\
- r & \leftarrow - u
\end{align*}$$

Note that $IDB^+$ differs from $IDB^*$ in the treatment of $q$, which is an $IDB$-predicate, but $q$ is false in $S_1$. 

A first idea to compute rational hitting sets would be to compute an update tableaux for a delete request \( \neg A \), where \( IDB^* \) is replaced by \( IDB^+ \) (cf. Def. 3.5). Indeed, this would work for Example 3.17 if we take \( EDB = \{ t \leftarrow, r \leftarrow \} \) and delete request \( \neg p \). The single hitting set computed by this method is just \( \{ t \} \). Deleting \( t \leftarrow \) from \( IDB \) a correct solution (in the sense of the weak relevance postulate). However, as the following examples shows, in general there is a completeness problem with this approach:

**Example 3.18**
Consider the following database:

\[
IDB: \quad p \leftarrow q \land r \\
    p \leftarrow r \\
EDB: \quad q \leftarrow \\
    r \leftarrow
\]

The Least Herbrand Model of this database is \( \{ p, q, r \} \) and the transformed database based on this model is given as:

\[
IDB^+: \quad \neg r \lor \neg q \leftarrow \neg p \\
    \neg r \leftarrow \neg p
\]

To delete the view atom \( p \), the hyper tableau calculus is applied on \( IDB^+ \cup \{ \neg p \leftarrow \} \). There exist two different update tableaux, which are the following:

This tableau is obtained by using the first rule of \( IDB^+ \) first, and then using the second rule on the left branch. On the right branch the second rule is not applicable due to regularity. The two open branches provide the two models of \( IDB^+ \cup \{ \neg p \} \) which stand for the hitting sets \( \{ q, r \} \) and \( \{ r \} \). Clearly, \( \{ q, r \} \) is not minimal, and the left branch would be closed by the minimality test.

Notice that there are two possible contractions of \( EDB \) satisfying weak relevance: the one is deletion of \( r \) (\( r \) is an \( EDB \)-closed locally minimal abductive explanation for \( p \) by virtue of the second rule of \( IDB \)), and the other is deletion of \( q \) and \( r \) (take the first rule of \( IDB \)). However, the latter contraction could not be read off from either tableaux. One idea would be to drop the minimality test and consider the hitting sets along the open branches for deletion. However, as the right tableau shows, this would not help either, because it misses the hitting set \( \{ q, r \} \).

Of course, we do not want to search the space of all possible tableaux. Hence, we will weaken the calculus below such that both tableaux can be extended to compute all solutions.
The reason for the failure to compute both rational hitting sets lies in the regularity condition: it is too strong, and thus disables the application of clauses necessary to compute all rational hitting sets. However, with a weaker version of regularity we obtain a complete calculus. In the weaker, strict hyper tableaux we forbid to allow extension with the same clause more than once along each branch. This can be formalized equivalently as follows:

**Definition 3.19 (Occurrences, Strict Hyper Tableaux)**

Given a finite ground clause set $D$. For comparing two literals $L_1$ and $L_2$ occurring in clauses in $D$ we define $L_1 =_{occ} L_2$ iff $L_1$ is the same occurrence\(^5\) as $L_2$ in $D$. A branch $b = L_1, \ldots, L_n$ is called strict\(^6\) iff $L_i \neq_{occ} L_j$ for $1 \leq i, j \leq n$ and $i \neq j$. “non-strict” means “not strict”. These definitions are extended towards tableaux in the same way as the definition of regularity (cf. Section 2.2). Adapting Definition 2.6, we say that a branch $b$ in a strict hyper tableau is finished iff it is either closed, or else whenever $C$ is applicable to $b$, then extension of $b$ by $C$ yields some non-strict new branch.

Notice that every regular hyper tableau is also strict, but not necessarily vice versa. But as in regular hyper tableau, in strict hyper tableaux no branch can be extended infinitely often.

Without proof we note that strict hyper tableaux are sound and complete. In particular, completeness holds because the same model construction for open hyper tableaux can be carried out as in the regular case.

**Definition 3.20 (Update Tableau based on Materialized View)**

An update tableau based on materialized view for a database $IDB \cup EDB$ and delete request $\neg A$ is a strict hyper tableau $T$ for $IDB^+ \cup \{ \neg A \leftarrow \}$ such that every open branch is finished. The notion of trivial update tableau is similar to that of Definition 3.7.

**Example 3.21**

Consider the database in Example 3.18 again. Two strict update tableau for delete request $\neg p$ exist, which are the following:

![Update tableau 1](image1)

Update tableau 1.

![Update tableau 2](image2)

Update tableaux 2.

Notice that both tableaux now carry along the open branches the two rational hitting sets as desired, which are $\{q, r\}$ and $\{r\}$.

Below we will show the general result, namely that the hitting sets of the open branches of any update tableau based on materialized view satisfy weak relevance. Before we are going to prove this, let us first give the algorithm (Algorithm 3):
Algorithm 3 View deletion algorithm based on materialized view

Input: A database $IDB \cup EDB$ and a ground view atom $A$ to be deleted.

Output: A new database $IDB \cup EDB'$

begin

1. Construct a branch $b$ of an update tableau based on materialized view (cf. Definition 3.20) for $IDB \cup EDB$ and delete request $\neg A$.

2. Produce $IDB \cup EDB \setminus HS(b)$ as a result. ($HS(b)$ of a branch $b$ of an update tableau is defined in Definition 3.5)

end.

Example 3.22
Consider this database:

$$
IDB: \quad ans(X, Y) \leftarrow par(X, Y) \\
\quad ans(X, Y) \leftarrow par(X, Z) \land ans(Z, Y) \\

EDB: \quad par(a, b) \\
\quad par(b, c) \\
\quad par(b, d) \\
\quad par(d, e)
$$

To delete $ans(a, e)$, hyper tableau calculus is applied on $IDB^+ \cup \{\neg ans(a, e) \leftarrow\}$ as follows:

$$
\neg ans(a, e) \\
\quad \neg par(a, b) \quad \neg ans(b, e) \\
\quad \quad \neg par(b, d) \quad \neg ans(d, e) \\
\quad \quad \quad \neg par(d, e)
$$

Each branch constitutes a hitting set: $\{par(a, b)\}, \{par(b, d)\}, \{par(d, e)\}$. Any one of these can be selected and removed from $EDB$. Note that $\{par(a, b), par(b, d), par(d, e)\}$ is an $EDB$-cut across the tableau. Also note that it is the unique $EDB$-closed locally minimal abductive explanation for $ans(a, e)$.

Like the approach with minimality test, this algorithm runs on polynomial space. Unlike the previous one, this does not require a minimality test and hence of polynomial time complexity too. But, this requires some offline pre-processing of computing the Least Herbrand Model. Note that, unlike the other approach based on minimality test, this method may generate a non-minimal (but rational) hitting set.

So, this approach for view deletion may not satisfy (KB—6.1) in general. But, as shown in the sequel, conformation to (KB—6.3) is guaranteed and thus this approach results in rational deletion. Furthermore, any deletion conforming to (KB—6.3) can be computed.
3.2 An approach based on materialized view

We start with the following lemma which corresponds to Lemma 10 of the approach based on the minimality test. The first item states completeness of the approach. To prove this, we need the relaxation of regular towards strict hyper tableaux. The second item expresses soundness, and the third item states that the EDB-cuts consist only of atoms which are part of some EDB-closed locally minimal abductive explanation for \( A \).

**Lemma 14**

Let \( T \) be a non-trivial update tableau based on materialized view for \( IDB \cup EDB \) and delete request \( \neg A \). Let \( S \) be the set of all EDB-closed locally minimal abductive explanations for \( A \) wrt \( IDB \). Let \( S' \) be the set of all EDB-cuts of \( T \). Then the following hold:

- \( S \subseteq S' \).
- \( \forall \Delta' \in S' : \exists \Delta \in S \text{s.t. } \Delta \subseteq \Delta' \)
- \( \forall \Delta' \in S' : \Delta' \subseteq \bigcup S \)

**Proof:** The proof of the first item is similar to the corresponding one in Lemma 10. However, we will give it explicitly in order to make clear why strictness is necessary. Thus, let \( EDB \supseteq \Delta \in S \) be a given locally minimal abductive explanation for \( A \). By definition of “locally minimal” (Def. 2.3), there is some subset \( M \subseteq IDB \) such that \( \Delta \) is a minimal abductive explanation for \( A \) wrt. \( M \).

We have to show that there is an EDB-cut of \( T \) which consists of the same literals as \( \Delta \). First, note that \( IDB \cup \Delta \cup \{ \neg A \} \) is inconsistent (though not necessarily minimal inconsistent). As a property of the renaming, it holds that \( IDB^+ \cup \Delta^+ \cup \{ \neg A \leftarrow \} \) is inconsistent, where \( \Delta^+ := \{ \leftarrow \neg A \mid A \in \Delta \} \). For later use, we let \( M^+ \subseteq IDB^+ \) denote the transformation of \( M \subseteq IDB \) wrt. the least Herbrand model of \( IDB \cup \Delta \), and note that \( M^+ \cup \Delta^+ \cup \{ \neg A \leftarrow \} \) is minimally inconsistent (by the satisfiability preservation of the transformation, and by the fact that \( \Delta \) is a minimal abductive explanation for \( A \) wrt. \( M \)).

Similarly, since \( IDB \cup \{ \neg A \} \) is consistent (for syntactical reasons: IDB contains no facts), both \( IDB^+ \cup \{ \neg A \leftarrow \} \) and \( M^+ \cup \{ \neg A \leftarrow \} \) are consistent as well. Now let \( T_M \) be a strict hyper tableaux for \( M^+ \cup \{ \neg A \leftarrow \} \) such that every open branch is finished. Notice that \( T_M \) in general is not an update tableau based on materialized view for \( IDB \cup EDB \) and delete request \( A \), because possibly \( M^+ \subseteq IDB^+ \).

By soundness of hyper tableaux (Theorem 4), \( T_M \) must contain open branches. Further, every open branch \( b_i \) in \( T_M \), for \( 1 \leq i \leq n \), must close with some (negative unit) clause of the form \( \leftarrow \neg A_i \in \Delta^+ \). This holds, because otherwise, by completeness of hyper tableaux (Theorem 4), we would have a contradiction to the inconsistency of \( M^+ \cup \Delta^+ \cup \{ \neg A \leftarrow \} \). Furthermore, each of the negative unit clauses in \( \Delta^+ \) must be used (at least once) to close some branch \( b_i \), because otherwise we would have together with soundness of hyper tableaux a contradiction to the minimal unsatisfiability of \( M^+ \cup \Delta^+ \cup \{ \neg A \leftarrow \} \). Thus, in other words, collecting the respective EDB-atoms from the \( b_i \)'s gives the EDB-cut \( \Delta \) of \( T_M \).
Next, we will argue that $T$ contains the same \emph{EDB}-cut $\Delta$ as $T_M$. First, $T_M$ can be extended to an update tableau based on materialized view for \emph{IDB \cup EDB} and delete request $A$, by extending as long as possible (i.e. until every open branch is finished) with the clauses from $\emph{IDB}^+ \supseteq M^+$. Let $T'_M$ be the resulting hyper tableau. Notice that trivially all $\emph{EDB}$-cuts through $T_M$ are preserved. Next, apply Proposition 15 to conclude that $T$ consists of the same open branches as $T'_M$, when viewed as a set of occurrences. Hence, also their $\emph{EDB}$-cuts are the same. Thus, $\Delta$ is an $\emph{EDB}$-cut of $T$ as desired.

The argumentation for the second item is much the same as in the proof of the second item of Lemma 10. Therefore, we will not repeat it here. The crucial point is that the interpretation used for renaming includes all of the $\emph{EDB}$-atoms, which is the case.

The third item can be shown by proving the following proposition: $\forall r \in \Delta : r \in \bigcup S$.  

Hyper tableaux, like most tableau calculi, enjoy the property of \emph{proof confluence}: every derived tableau so far can be continued to a refutation, if one exists at all. Proof confluence is a most important property because it enables the design of proof procedures which do not backtrack over the generated tableaux. For our purposes we need a stronger result, saying that the models computed in any two hyper tableaux are the same:

**Proposition 15 (Model Confluence of Strict Hyper Tableaux)**

Let $T_1$ and $T_2$ be strict hypertableaux for some clause set $M$ such that in both $T_1$ and $T_2$ every open branch is finished. Then the open branches of $T_1$ and $T_2$ are the same wrt. occurrences of literals. More formally,

\[
\text{Branches}(T_1) = \text{Branches}(T_2)
\]

where

\[
\text{Branches}(T) = \{\text{OccSet}(b) \mid b\ \text{branch in } T\},
\]

\[
\text{OccSet}(b) = \{L \mid L\ \text{is a literal occurrence in } b\}.
\]

For illustration consider the two strict tableaux in Example 3.21, and notice that the branch sets of occurrences are the same. On the other side, the tableaux in Figure 3.18 give a counterexample why Proposition 15 does not hold for regular hyper tableaux.

\textbf{Proof:} We will define a strictness preserving transformation $\tau$ on hyper tableaux where every open branch is finished, such that $\text{Branches}(T_1) = \text{Branches}(T)$ (let us say that $\tau$ \emph{preserves branches}), and such that every open branch in $\tau(T)$ is finished as well. Furthermore, $\tau$ does not increase the depth of $T$.

Using $\tau$, we will stepwisely transform $T_2$ into $T_1$. Since $\tau$ preserves branches, this immediately proves the proposition.

Now for the application of $\tau$: either $T_2 = T_1$ and we are done. Equality here means that $T_1 = T_2$ iff there is a bijection between the nodes (resp. edges) of $T_1$ and $T_2$ such that corresponding nodes are labelled with the same literal occurrence.

If $T_1 \neq T_2$ then find two partial branches (cf. Section 2.2) $b_1$ and $b_2$ through $T_1$ and $T_2$, respectively, such that $b_1 = b_2$. Here, two branches are equal iff they are
3.2 An approach based on materialized view

equal as sequences of literal occurrences. The branches $b_1$ and $b_2$ can be selected such that they are of maximal length wrt. this property, and that the last node is not a leaf. Pictorially, we climb down $T_1$ and $T_2$ as long as possible until we find a difference.

More formally, there exist partial branches $b_1$ in $T_1$ and $b_2$ in $T_2$ such that $b_1 = b_2$ and $(b_1, L_1) \neq (b_2, L_2)$ for every partial branch $(b_1, L_1)$ in $T_1$ and every partial branch $(b_2, L_2)$ in $T_2$. Such $b_1$ and $b_2$ must exist because $T_1 \neq T_2$. Further, it is impossible that $b_1$ is a prefix of $b_2$, i.e. $b_2 = (b_1, b)$ for some node sequence $b$. This holds, because otherwise a hyper extension step would be applicable to $b_1$ (namely the one which was applied to $b_2$), contradicting the given assumption that every open branch in $T_1$ is finished. For the same reason it is impossible that $b_2$ is a prefix of $b_1$. Hence, the partial branches $(b_1, L_1)$ and $(b_2, L_2)$ indeed exist as suggested.

The transformation $\tau$ will be applied to $T_2$ at branch $b_2$ in such a way that after the transformation the immediate successor nodes of $b_2$ are labelled with the same literal occurrences as the immediate successor nodes of $b_1$. Since $\tau$ acts local to the subtree below $b_2$, the transformed tableaux is strictly more in accordance with $T_1$, in the sense that the partial branches $b_1$ and $b_2$ can both be extended by one node and are still equal. Further, since $\tau$ does not increase the depth of the tableaux, repeated application of $\tau$ necessarily terminates after finitely many steps.

Hence it remains to define the transformation $\tau$, proper. It is shown in Figure 1. Suppose that the immediate successor nodes of $b_1$ being labeled with positive literal occurrences are $C_1, \ldots, C_{n_C}$, i.e. $b_1$ was extended with some clause $C$ with these head literals (topmost tree in Figure 1). Similarly, suppose that $b_2$ was extended with a different clause $B$ with head literals $B_1, \ldots, B_{n_B}$ (middle tree). From $T_1$ we learn that $C$ is applicable to $b_1$, and since $b_1 = b_2$, $C$ is applicable to $b_2$ as well. Since in $T_2$ every open branch is finished, $C$ must have been used for an hyper extension step in every open branch in the subtree below $b_2$ (i.e. at least once). This is also indicated in the middle tree in Figure 1. $C$ can occur in each subtree below $B_i$ (for $1 \leq i \leq n_B$) several times (only for ease of presentation, these occurrences are drawn at the same level). Further, below each occurrence of a literal $C_j$ ($1 \leq j \leq n_C$) there is a subtree, where $k$ further names the occurrences of the subtree below $C_j$ passing through $B_i$.

The bottommost tree in Figure 1 shows the result of the $\tau$-transformation. First, we extend $b_2$ with $C$ instead of $B$. As can be learned from $T_1$, $C$ is applicable to $b_2$. Then, each branch $(b_2, C_j)$ is extended with $B$. Clearly, $B$ is applicable to these branches, because $B$ is even applicable to $b_2$ alone (let us say that “context has increased” in such situations). Notice that through this exhaustive extension both $C$ and $B$ appear in every open branch below $b_2$.

Next, the extension steps producing the subtrees $T^{B_i}$ in $T_2$ are repeated below the branches $(b_2, C_j, B_i)$ in $\tau(T_2)$. Again, this is possible because context has increased. Note that there is no problem with strictness, because $C$ does not occur in $T^{B_i}$.

Finally, the branches in the repeated extension steps of $T^{B_i}$ in $\tau(T_2)$ have to be extended by repeating the extension steps leading to $\tau^{B_i}$ in $T_2$.

By this transformation the depth of the tree stays the same, as even for the longest branch only a different order of literal occurrences is produced. Further, by construction, for each open branch in $\tau(T_2)$ which finally passes through $\tau^{B_i}$ there
is an open branch in $T_2$ being labeled with the same literal occurrences, only in a
different order. In sum, this transformation achieves the desired property above,
namely that $\tau(T_2)$ is one level more in accordance with $T_1$ than $T_2$.

Lemma 16
Let $IDB \cup EDB$ be a database and $-A$ a delete request. Let $S$ be the set of all $EDB$-
closed locally minimal abductive explanations for $A$ wrt $IDB$. Then there exists an
update tableau based on materialized view $T$ for $IDB \cup EDB$ and $-A$ s.t. $S \subseteq S'$,
where $S'$ is the set of all $EDB$-cuts of $T$.

Lemma 17 ([AD95, Ara95])
Let $S$ and $S'$ be sets of sets s.t. $S \subseteq S'$ and for every member $X$ of $S' \setminus S$: $X$ contains
a member of $S$ and $X$ is contained in $\bigcup S$. Then, a set $H$ is a hitting set for $S$ iff it
is a hitting set for $S'$.

Lemma 18
Let $T$ as in Lemma 14. Then $HS(b)$ is a rational hitting set for $A$, for every open
finished branch $b$ in $T$.

Proof: Follows from Lemma 14 and Lemma 17.

Now the following main theorem, stating the rationality of Algorithm 3, follows
immediately.

Theorem 19 (Main Theorem)
The above algorithm is rational, in the sense that it satisfies all the rationality pos-
tulates (KB—1), (KB—2), (KB—3), (KB—4), (KB—5), and (KB—6.3).

4 Concluding Remarks
We have presented two variants of an algorithm for deleting a view atom from a
definite database. The key idea of this approach is to transform the given database
into a disjunctive logic program in such a way that updates can be read off from
the models of this transformed program. In contrast to the previous approaches, this
algorithm is of polynomial space complexity. One variant based on materialized views
is of polynomial time complexity. Moreover, we have also shown that this algorithm
is rational in the sense that it satisfies the rationality postulates that are justified
from philosophical angle.

As mentioned before, this algorithm is based on a diagnosis algorithm presented
in [BFFN97b, BFFN97a]. An implementation exists for this diagnosis algorithm and
has been tested extensively on real world examples. This implementation can be
easily adopted for view updates as well and we are working on that now.

In the second variant, where materialized view is used for the transformation,
after generating a hitting set and removing corresponding $EDB$ atoms, how do we
easily move to the new materialized view? An obvious way is to recompute the view
from scratch using the new $EDB$ (i.e. compute the Least Herbrand Model of the
Figure 1: The $\tau$-transformation.
new updated database from scratch), but it is certainly interesting to look for more
efficient methods. A reasonable answer for this question will greatly increase the
significance of this approach. This is indeed the view maintenance problem studied
by various researchers [GM95, for example].

Our approach works on the assumption that the EDB is available and the com-
plete EDB is indeed used for the transformation. It is interesting to study whether
this approach can be effectively used in situations where EDB is very huge or not
completely known. It should not be difficult to work with only that part of the EDB
upon which the current view update request depends on, but a formal study in this
regard is necessary.

We are also exploring how this approach can be extended for disjunctive databases,
where there is generally no unique minimal model.

Acknowledgements

The authors would like to thank all the members of the Artificial Intelligence Re-
search Group at the University of Koblenz, Germany, for stimulating discussions on
this topic. This research is a part of projects on Automated Deduction and Dis-
junctive Logic Programming funded by DFG (grants Fu 263/2-2 and Fu 263/3-1
respectively).

References

[AB97] Chandrabose Aravindan and Peter Baumgartner. A Rational and Effi-
cient Algorithm for View Deletion in Databases. In Jan Maluszynski,
editor, Logic Programming - Proceedings of the 1997 International Sym-

and D. Van Gucht, editors, Proceedings of the second international con-
dference on database theory, volume Lecture Notes in Computer Science

[AD95] Chandrabose Aravindan and Phan Minh Dung. Knowledge base dynam-
ics, abduction, and database updates. Journal of Applied Non-Classic-
<http://www.uni-koblenz.de/~arvind/papers/>.

change: Partial meet contraction and revision functions. The Journal of

[Ara95] Chandrabose Aravindan. Dynamics of Belief: Epistemology, Abduction,
and Database Updates. PhD thesis, Computer Science Program, Asian
Institute of Technology, Bangkok, Thailand, 1995. Available on the web from
<http://www.uni-koblenz.de/~arvind/papers/dissert/>.
References


References


