J. Symbolic Computation (1999) **11**, 1–000

Theorem Proving Techniques for View Deletion in Databases[†]

CHANDRABOSE ARAVINDAN AND PETER BAUMGARTNER[‡]

Fachbereich Informatik, Universität Koblenz-Landau,
Rheinau 1, D-56075 Koblenz, Germany
peter@informatik.uni-koblenz.de

(Received 7 May 1999)

In this paper, we show how techniques from 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 program and to apply the hyper tableaux calculus [BFN96] to solve the original update problem. The resulting algorithm has the following properties: it works goal-directed (i.e. the search is driven by the update request), it is rational in the sense that it satisfies certain rationality postulates stemming from philosophical works on belief dynamics, and, unlike comparable approaches, it is of polynomial space complexity.

To obtain soundness and completeness results, the hyper tableau calculus is slightly modified for minimal model reasoning. Besides a direct proof we give an alternate proof which gives insights into the relation to previous approaches. As a by-product we thereby derive a soundness and completeness result of hyper tableaux for computing minimal abductive explanations.

Keywords: *database updates, theorem proving, 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] provides 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 philosophical 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 this and other known rational

[†] A preliminary version of this paper was published as [AB97]
[‡] Funded by the DFG within the research programme "Deduction" under grant Fu 263/2-2

0747–7171/90/000000 + 00 $03.00/0

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. 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 related to our diagnosis setup presented in [BFFN97b, BFFN97a], where the hyper tableau 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 extract diagnosis from them (the relation to diagnosis is made precise in Section 4.1 below).

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 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.

In order to be rational, we show that a rationality axiom itself (the so-called "'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, respectively) 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 rest of the paper is organized as follows: We first briefly recall some basic ideas from the field of *belief dynamics*, the so-called AGM-postulates, and instantiate this framework to the special case of deletion of view predicates in deductive databases. Then we turn to the hyper tableau calculus and use it to present our new algorithm. Next, we point out the relation to the diagnosis framework, and prove soundness and completeness. The paper is concluded with some comments on our approach and indications for further work.

2. Background – Rationality of Change

We live in a constantly changing world, and consequently our beliefs have to be revised when there is new information. The central problem of epistemology is to study when we

can be sure that we have revised our belief rationally. This has been studied at a more general and abstract level by researchers from the field of philosophy, leading to a new branch of study: the *belief dynamics*. Defining a *belief set* as a deductively closed set of sentences, Alchourrón, Gärdenfors and Makinson propose certain *rationality postulates*, popularly known as *AGM-postulates*, to be satisfied when the belief set is revised (see [GR95, BJdR97] for very good overviews). The term "revision" is meant in a wide sense here – the general problem of changing belief states. We will refer to this meaning by explicitly speaking about "belief revision". In a narrower sense, revision is one of the following three basic operations on belief sets:

- *Expansion:* A new sentence α consistent with the old belief set K is added to K. The new set is denoted by $K+\alpha$.
- *Revision:* A new sentence α *in*consistent with the old belief set K is added to K. In order to preserve consistency, some old sentences from K are deleted.
- *Contraction:* Some sentence α is retracted from the old belief set K without new information being added. To ensure deductive closure further sentences from K may have to be given up. We write $K—\alpha$ to denote the result of contracting K with α.

Now, for each of these operations there is a set of corresponding AGM-postulates that describes desired, "rational" properties of the operations. They relate the belief sets before and after the operation, and, intentionally, do *not* uniquely specify the result of the operation. Clearly, this would depend at least from the underlying logic and what is meant by "deductive closure". It has been stated in the literature that doing belief revision in this framework has the advantage that it can be based on *classical first-order* logic (as opposed to e.g. non-monotonic or paraconsistent logics). Thus, we also take the term "deductive closure" to mean "closed under first-order derivability".

The AGM-postulates are guided by a few basic integrity constraints: (i) belief sets should be kept consistent, (ii) belief sets are deductively closed, (iii) the changes to K should be kept minimal. In accordance with these guidelines, among the three operations, expansion is the least problematic one. One can define the expansion of a belief set K by a sentence α, written as $K+\alpha$, as $K+\alpha = Cn(K \cup \{\alpha\})$, where $\alpha \in Cn(K)$ iff $K \vdash \alpha$ (in first-order logic).

Unfortunately, revision and contraction are not so straightforward. For instance, when revising K with α one would have to remove elements from K such that α no longer follows, and this might be not unique. Since revision and contraction can be expressed in terms of each other[†], we will concentrate in the sequel on contraction. The AGM-postulates for contraction are as follows:

(K—1) (Closure)	$K—\alpha$ is a belief set
(K—2) (Inclusion)	$K—\alpha \subseteq K$
(K—3) (Vacuity)	If $\alpha \notin K$, then $K—\alpha = K$
(K—4) (Success)	If $\not\vdash \alpha$, then $\alpha \notin K—\alpha$

[†] For instance, in the one direction, revision can be defined to first prepare K by contracting $\neg\alpha$ and then expand with α. The reader is referred to [GR95] for details and the provisos when this is possible.

(K—5) (Recovery) If $\alpha \in K$, then $K \subseteq (K—\alpha)+\alpha$
(K—6) (Extensionality) If $\vdash \alpha \leftrightarrow \beta$, then $K—\alpha = K—\beta$
(K—7) (Conjunction 1) $K—\alpha \cap K—\beta \subseteq K—(\alpha \wedge \beta)$
(K—8) (Conjunction 2) If $\alpha \notin K—(\alpha \wedge \beta)$ then $K—(\alpha \wedge \beta) \subseteq K—\alpha$

The postulate (K—1) just expresses deductive closure, (K—2) expresses the contraction should be a reduction (contraction does not result in new consequences), (K—3) expresses that nothing changes when the formula to be contracted is not in the current belief set, (K—4) expresses that contraction is successful, except for tautologies, which are contained in all belief sets. The postulate (K—3) realises to some degree the *minimality* idea of (cf. the invariant (iii) mentioned above), but the idea is only fully expressed by postulate (K—5): enough must be left of the original theory K so as to enable us to restore it after a contraction. The postulate (K—6) says that contractions with equivalent formulas yield the same result (syntax is irrelevant). The postulates (K—7) and (K—8) relate contractions between a formula α and the stronger formula $\alpha \wedge \beta$.

This concludes our brief and shallow presentation of *belief sets*. We refer the reader to the mentioned surveys [GR95, BJdR97] (which our presentation is based upon) and the referrences therein for further information.

2.1. Knowledge Base Revisions

It has been argued that the requirement of belief sets of being deductively closed is disputable: first, no distinction is made between "basic" knowledge and "derived", and, second, revisions are more naturally thought of as being carried out to some (typically finite) set *representing* a belief set. It is not clear how the AGM-postulates can be applied in real world problems such as database updates.

As one alternative, it is suggested to consider an arbitrary, (typically finite) set KB of sentences and call it a *belief base (or knowledge base) for a belief set* K iff $Cn(KB) = K$. In a next step, the AGM-postulates can be modified to cope with not deductively closed belief bases instead of belief sets. This introduces a more fine-grained structure, as it might well be that we have two belief bases KB and KB' with $KB \neq KB'$ but $Cn(KB) = Cn(KB')$. We will skip these modified AGM-postulates for such *base revisions* here. Instead we directly jump to the particular approach of [AD95, Ara95]. There, a knowledge base KB is defined as a finite set of sentences from a language L that is divided into two parts: an immutable theory KB_I, which is the fixed part of the knowledge; and an updatable theory KB_U. In a deductive database setup KB_U typically consists of the facts and KB_I typically consists of the rules and integrity constraints.

In [AD95, Ara95] further modifed AGM-postulates for this scenario are defined. Because of the inter-expressibility of revision and contraction, it is enough to consider one. The rationality postulates for contracting a sentence α from a knowledge base KB, written as KB—α are reproduced below.

Definition 2.1 (Knowledge Base, KB-equivalence)
Assume as given some first-order language L, and let $K \vdash \alpha$ denote first-order derivability of a sentence α from a set of sentences K. For convenience, define $\alpha \in Cn(K)$ iff $K \vdash \alpha$.

Let $KB = KB_I \cup KB_U$ be a *knowledge base*, consisting of two disjoint sets of sentences. KB_I is called the *immutable part*, and KB_U is called the *updatable part* of KB.

Let α and β be any two sentences. Then, α and β are said to be *KB-equivalent* iff the following condition is satisfied: For all sets of sentences E: $KB_I \cup E \vdash \alpha$ iff $KB_I \cup E \vdash \beta$, where $\vdash$ denotes a first-order derivability relation coinciding with the usual semantic consequence relation $\models$. ■

DEFINITION 2.2 (RATIONALITY POSTULATES FOR KNOWLEDGE BASE CONTRACTIONS)

(KB—1) (Inclusion)	$KB—\alpha \subseteq KB$
(KB—2) (Immutable-inclusion)	$KB_I \subseteq KB—\alpha$
(KB—3) (Vacuity)	If $\alpha \notin Cn(KB)$, then $KB—\alpha = KB$
(KB—4) (Immutable-success)	If $KB_I \not\vdash \alpha$, then $\alpha \notin Cn(KB—\alpha)$.
(KB—5) (Extensionality)	If α and β are KB-equivalent, then $KB—\alpha = KB—\beta$
(KB—6.1) (Strong relevance)	If† $\beta \in KB \setminus KB—\alpha$, then $\alpha \in Cn(KB—\alpha \cup \{\beta\})$
(KB—6.2) (Relevance)	If $\beta \in KB \setminus KB—\alpha$, then there exists KB' with $KB—\alpha \subseteq KB' \subseteq KB$ s.t. $\alpha \notin Cn(KB')$ and $\alpha \in Cn(KB' \cup \{\beta\})$
(KB—6.3) (Weak relevance)	If $\beta \in KB \setminus KB—\alpha$, then there exists KB' with $KB' \subseteq KB$ s.t. $\alpha \notin Cn(KB')$ and $\alpha \in Cn(KB' \cup \{\beta\})$

■

The AGM-postulate (K—1) from above is missing in the new definition. This must obviously be the case because it was a design decision not to insist on deductive closure; (K—2) is contained as (KB—1) in the new definition; the additional immutability restriction (KB—2) expresses that no sentence from KB_I can be thrown away; (KB—3) is the adaption of (K—3) from the AGM-postulates above. As in the case of belief sets, the contraction should be successful, i.e. if $\vdash \alpha$ then $\alpha \notin Cn(KB—\alpha)$. But, since the immutable theory never changes, any knowledge implied by it can also never change. Hence, the above success postulate is rewritten as (KB—4). Similarly, the irrelevance of syntax in (K—6) is adapted to yield (KB—5).

Note that we have three variants of the relevance postulate of varying strength. Definition 2.2 is meant to include exactly one of the three cases. To illustrate relevance, consider a knowledge base KB with $KB_I = \{p \leftarrow q \wedge r\}$ and $KB_U = \{q \leftarrow , r \leftarrow , s \leftarrow \}$. Let $\alpha = p \leftarrow$ be the sentence to be contracted. Now, in order to satisfy (KB—4), at least one of $\{q \leftarrow , r \leftarrow \}$ has to be removed from KB_U. Removing $q \leftarrow$ alone (and likewise, removing $r \leftarrow$ alone) would be perfectly in accordance with all three versions of the relevance postulate. Removing both $q \leftarrow$ and $r \leftarrow$ would violate strong relevance, but still conform to relevance and weak relevance.

Now, adding to KB_I the clause $p \leftarrow r$‡ and removing in this new knowledge base both

† The "\" operator means set difference, as usual.

‡ Of course, having both $p \leftarrow r$ and $p \leftarrow q \wedge r$ in a clause set does not make much sense in "usual" first-order theorem proving context. However, the example is for illustration purposes only, and one might well conceive first-order deductive databases where these clauses stand for ground instances of two first-order clauses where subsumption is not applicable.

$q \leftarrow$ and $r \leftarrow$ in order to contract $p \leftarrow$ would still be correct wrt. weak relevance, but no longer correct wrt. relevance.

The weaker forms of relevance are motivated by various works of Hansson [Han91b, Han91a]. In our concrete context of database updates, plausible examples can be conceived that motivate each of them. For example, consider a knowledge base KB with[†] $KB_I = \{student\_dean(X, Y) \leftarrow student\_school(X, Z) \wedge school\_dean(Z, Y)\}$ and $KB_U = \{student\_school(a, b) \leftarrow , school\_dean(b, c) \leftarrow \}$. Then, clearly, $student\_dean(a, c)$ is a logical consequence of KB. Now, in order to delete $student\_dean(a, c)$ it would be rational to delete from KB_U either $student\_school(a, b) \leftarrow$ (because a might have graduated) or $school\_dean(b, c) \leftarrow$ (because c is no longer dean). Both operations conform to strong relevance, but deleting both elements from KB_U in one operation would violate strong relevance (and does not seem rational). On the other hand, consider now $KB_I = \{schol(X) \leftarrow student(X) \wedge thesis(X)\}$ and $KB_U = \{student(a) \leftarrow , thesis(a) \leftarrow \}$. KB_I may say that X is entitled for scholarship if he is a student and doing his thesis. Clearly, $schol(a) \leftarrow$ follows in this case. In order to delete $schol(a) \leftarrow$ (because a graduated) it seems rational to delete both $student(a) \leftarrow$ and $thesis(a) \leftarrow$ from KB_U. Note that this operation does not conform to strong relevance, but both weaker forms are still satisfied.

In the present paper, we concentrate on strong relevance, because it is computationally the most delicate one: strong relevance is a property depending on the whole knowledge base, whereas, on the other extreme, weak relevance may be computed more "locally". It will be interesting future work to weaken our calculus below to cope with the other forms as well.

2.2. An Algorithm Based on Abduction

Now we briefly recall an algorithm for contraction based on abduction [AD95, Ara95]. Some basic definitions, similar to those in [Poo89] required for the algorithm are presented first.

An *abductive framework* is a pair (T, Ab), where both T (the background theory) and Ab (the abducibles) are sets of sentences. An abductive explanation for a sentence α is defined as a subset $\Delta \subseteq Ab$ such that $\Delta \cup T$ is consistent and $\Delta \cup T \models \alpha$.

The following definition contains the "usual" definition of abductive explanations, but slightly specialized to our case. We assume as fixed some first-order language L.

Definition 2.3 ((Minimal) Abductive Explanation)
Let KB be a knowledge base α be a sentence. An *abductive explanation* Δ for α wrt. KB_I is a set Δ of ground literals s.t. $\Delta \cup KB_I \models \alpha$ and $\Delta \cup KB_I$ is consistent. An abductive explanation is *minimal* iff 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. Δ is a minimal abductive explanation of α wrt KB'_I. Further, Δ is said to be *KB-Closed* iff $\Delta \subseteq KB_U$. ■

Our interest is in particular in KB-Closed explanations, because such explanations are used below as a base to determine contractions, and only members of KB_U shall be considered for this (because the other part, KB_I, shall be immutable).

[†] Variables start with capital letters and are considered as universally quantified.

EXAMPLE 2.4
Consider again the knowledge base KB from above whose immutable part KB_I is $\{p \leftarrow q \wedge r, p \leftarrow r\}$. 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, Δ_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 below. The basic idea behind this algorithm is to generate first all (locally minimal) explanations for the sentence to be contracted and then extract the necessary deletions from them. The underlying technical device is called a *hitting set:*

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. ■

Thus, a hitting set of S is obtained by picking one element from every non-empty element from S. Hitting sets are used in the following algorithm for contraction:

Algorithm 1 General contraction algorithm

Input: A knowledge base $KB = KB_I \cup KB_U$ and a sentence α to be contracted.
Output: A new knowledge base $KB' = KB_I \cup KB'_U$

begin

1. Construct a set S = {X | X is a KB-closed minimal abductive explanation for α wrt KB_I}.
2. Determine a minimal hitting set σ(S).
3. Produce $KB' = KB_I \cup (KB_U \setminus \sigma(S))$ as a result.

end.

The following theorem was proven in [AD95, Ara95].

THEOREM 2.1 (CORRECTNESS AND COMPLETENESS OF ALGORITHM 1)
Let KB be a knowledge base and α a sentence.

(1) If Algorithm 1 produces KB' as a result of contracting α from KB, then KB' satisfies all the rationality postulates (KB—1), (KB—2), (KB—3), (KB—4), (KB—5), (KB—6.3). Further, if the hitting set computed at step 2 is minimal, then (KB—6.1) is also satisfied.
(2) Suppose KB'' statisfies all these rationality postulates for contracting α from KB, then KB'' can be produced by Algorithm 1.

The disadvantage of this algorithm is that all (locally minimal) explanations have to be generated first. Since there might be exponentially many abductive explanations (wrt. the number of hitting sets) this algorithm is of exponential space and time complexity. Our new algorithm below avoids the exponential space requirement.

2.3. Database Updates and Previous Approaches

We now instantiate the framework presented so far to the special case of deductive databases. A *definite deductive database DDB* consists of two parts: an *intensional database IDB*, which is a set of definite program clauses, and an *extensional database EDB*, which is 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*. We assume that the language underlying a *DDB* is fixed, and that there are no function symbols, implying that the Herbrand Base is finite. Therefore, the *IDB* is practically a shorthand of its ground instantiation†, written as IDB_G. In the sequel, technically we mean IDB_G when we refer simply to *IDB*. 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* (i.e. occur in the head of some clause in *IDB*) are referred to as *view predicates*; 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*, or *IDB atom*, and similarly an atom with base predicate is a *base atom*, or *EDB 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*. The 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, Dec90, Dec96, GL90, GL91, KM90, Tom88, for example]. 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 *EDB* facts, so that the modified *EDB* together with *IDB* will satisfy the deletion of A from *DDB*.

But what does "satisfy the deletion" mean? Clearly, A should not follow any more from the updated *DDB*. But is this the only criterion? We think that the general results from belief revision provide reasonable guidelines for database updates as well. Hence, we are now embedding the view deletion problem in deductive database in the above framework of knowledge base contractions: a *DDB* represents a knowledge base where the immutable part KB_I is given by IDB_G and the updatable part KB_U is given by the *EDB*. The "insert" operation in deductive database translates into the belief revision term "expansion", and "deletion" translates into "contraction". Hence, the rationality postulates (KB—1), (KB—2), (KB—3), (KB—4), (KB—5), and (KB—6.i) (where $i \in \{1, 2, 3\}$) provide an axiomatic characterization for deleting a view atom A from a deductive database *DDB*. These axioms, or at least some of them, are also present in a more or less explicit form

† 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.

in related work on database updates [DB82, Kel85, Lan90, for example]. Agreement seems to be in particular about the success postulate (KB—4) – that α should not be seen after deletion. In fact, [DB82] defines this to be the only correctness criteria. Concerning relevance, the strong form (KB—6.1) seems to be predominant (for example in [Kel85, Lan90]).

An algorithm for view deletion in deductive databases, based on the general contraction algorithm (cf. Algorithm 1) was presented in [AD95, Ara95]. In fact, it only requires to instantiate Algorithm 1. More specifically, the computation of abductive explanations is accomplished then by an SLD-resolution calculus, where the goal of the derivation is the view atom to be deleted. The set of *all* explanations (of the kind mentioned in Algorithm 1) for that is generated through a *complete* SLD-tree, and a hitting set of these explanations is then deleted from the *EDB*. It was shown in [AD95, Ara95] that this algorithm is rational (Theorem 2.1 above). 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 his algorithm, *all* hitting sets from a complete SLD-tree are computed in first step in a more direct way. Then, an inclusion-minimal set is chosen among them, and any such set determines the *EDB*-atoms actually to be deleted.

An analysis of the algorithm in [DB82] shows that it is a special case of Algorithm 1, and that it conforms to all our postulates, where weak relevance (KB—6.3) is used. See [AD95] for a more detailed comparison.

In contrast to these algorithms, the algorithm we are going to present directly computes a hitting set without explicitly generating all the abductive explanations or hitting sets as a preliminary step. Moreover, the generation of the hitting set is carried out through a hyper tableaux calculus that is focused on the goal.

Other algorithms, based on deduction trees (SLD or SLD with negation as failure), to delete a view atom from a deductive database have been forwarded in [Dec90, GL90, GL91, KM90]. These algorithms do not necessarily construct complete trees. Such approaches are problematic, as for instance, vacuity (KB—3) might no longer be satisfied. Consider for example $IDB = \{p \leftarrow q \wedge r\}$ and $EDB = \{q \leftarrow\}$. Deleting the view atom p might result then in deleting the base atom q. Since p does not follow from the database, deleting q violates the vacuity postulate. So, in order to avoid constructing a complete SLD tree, the derivability of p should be checked first.

In [Bry90] the computation of view updates has been formalized as a model generation process. This technique of "intentional updates" is rather general, as, among other things, intentional updates may refer to the current as well as the updated database. To make this possible, the (meta-level) updates are formulated at the *object* (formula) level and express the desired effect of the update. For instance, that a view atom p is to be deleted would be formulated as $new(\neg p)$, meaning that the "new" database must no longer entail p. To the given definite program one associates a (disjunctive) program which is used to compute the update. For instance, the *IDB* clause $p \leftarrow q \wedge r$ would be essentially translated into a program that is equivalent to $remove(q) \vee remove(r) \leftarrow new(\neg p)$. The *remove* predicate means the obvious operation on the *EDB*, namely that the argument has to be

deleted. To actually compute the update to the *EDB*, the bottom-up model generation prover SATCHMO is used. It was observed (for the first time, to our knowledge) in [Bry90] that update computation is an abductive problem then. The method of [Bry90] behaves operationally much like Algorithm 1 above, except that the minimization is not employed in the algorithm in [Bry90]. However, at the same time it was suggested in [Bry90] that minimization is an important issue. A result was proven, stating that any minimal update (in the sense of our strong relevance postulate KB—6.1) *can* be computed. We are faithful that our results below can be used to continue the programme proposed in [Bry90] so that *only* minimal updates are computed. This would also achieve that the vacuity postulate (KB-—3) is satisfied: in the example, computing $remove(q)$ and thus deleting q from *EDB* in order to delete p violates vacuity if the *EDB* does not contain r (cf. the discussion in the preceding paragraph).

3. Hyper Tableau Calculus for Minimal Model Reasoning

In [BFN96] a variant of clausal tableau calculus called "hyper tableaux" has been introduced. In the ground case, which we consider here as the basis for our view deletion algorithm, hyper tableaux coincide with the calculi underlying the SATCHMO prover [MB88] and the MGTP system [FH91]. More recent developments in the SATCHMO tradition are described in [BY96] (concerning minimal model reasoning) and in [BT98] (concerning finite models). Improvements for first-order hyper tableaux have been suggested in [Bau98].

We apply the usual notions of first-order logic, in a way consistent to [CL73]. For notions related to tableau calculi in general see [Fit90]. *Clauses*, i.e. multisets of literals, are written as the disjunction $A_1 \vee \cdots \vee A_m \vee \neg B_1 \vee \cdots \vee \neg B_n$ or as an implication $A_1 \vee \cdots \vee A_m \leftarrow B_1 \wedge \cdots \wedge B_n$ (for some $m \geq 0$, $n \geq 0$). The literals $A_1, \ldots, A_m$ (resp. $B_1, \ldots, B_n$) are called the *head* (resp. *body*) of the clause. Clause sets are also called *disjunctive logic programs*, in particular in the context of bottom-up evaluation.

With $\overline{L}$ we denote the complement of a literal L. Two literals L and K are *complementary* if $\overline{L} = K$. Since confusion is less likely, we may use the same operator $\overline{\cdot}$ to denote set complements.

From now on D always denotes a finite ground clause set, also called *database*, and Σ 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 instance (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 children 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 $i \neq j$. 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 $lit(b) = \{L_1, \ldots, L_n\}$. For brevity, we will write expressions like $A \in b$ instead of $A \in lit(b)$. In

order to memorize the fact that a branch contains a contradiction, we allow to label a branch as *closed*. A branch which is not marked as closed is marked as *open*. For brevity, we leave away the term "marked as" and simply say that a "branch is closed/open". A tableau is *closed* if each of its branches is closed, otherwise it is *open*.

DEFINITION 3.1 (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 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) $C = A_1 \vee \cdots \vee A_m \leftarrow B_1 \wedge \cdots \wedge B_n$ is a clause from D (for some $m \geq 0$, $n \geq 0$), 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: T' is the same as T, except that b is replaced by the *new* branches

$$(b, A_1) \ldots, (b, A_m), (b, \neg B_1) \ldots, (b, \neg B_n)$$

and every inconsistent new branch in T' is marked as "closed", and every other new branch is marked 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. ■

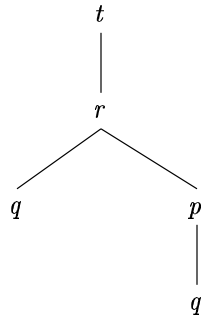
The applicability condition of an extension expresses that *all* body literals have to be satisfied by the branch to be extended (like in hyper *resolution* [Rob65]). This similarity to hyper *resolution* [Rob65] gave the name "hyper tableaux". Notice as an immediate consequence of the definition that open branches never contain negative literals.

EXAMPLE 3.2 (HYPER TABLEAUX)
Consider the following database D:

$$D: \quad p \vee q \leftarrow t \wedge r \qquad t \leftarrow$$
$$q \leftarrow p \wedge t \qquad r \leftarrow$$

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: after an initialization step, 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 $p \vee q \leftarrow t \wedge r$. The left branch is now finished, because $q \leftarrow p \wedge t$ is not applicable, and extension with any other clause would introduce an irregularity. Extension with $q \leftarrow p \wedge t$ at the right branch finishes this branch as well.

Hyper Tableau:



■

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.

3.1. Minimal Models

Next, we equip the calculus defined so far with semantics. Of particular interest is a certain form of minimal model reasoning.

Definition 3.3 ((Minimal) Model, Branch Semantics)
As usual, we represent an interpretation $\mathcal{I}$ for given signature Σ as the set $\{A \in \Sigma \mid \mathcal{I}(A) = \mathit{true}\}$. The notation $\mathcal{I} \models \mathit{object}$ means that *object* is *true* in $\mathcal{I}$ where *object* is a clause, an atom, a literal or a set of these (interpreted conjunctively). In particular, $\mathcal{I} \models A_1 \vee \cdots \vee A_m \leftarrow B_1 \wedge \cdots \wedge B_n$ iff $\{B_1, \ldots, B_n\} \subseteq \mathcal{I}$ implies $\{A_1, \ldots, A_m\} \cap \mathcal{I} \neq \emptyset$. *Minimality* of models (for given clause set D) is defined via set-inclusion.

Let b be a consistent branch in a given tableau. The branch b is mapped to an interpretation by defining $[\![b]\!]_\Sigma := \mathit{lit}(b)$. Usually, we write $[\![b]\!]$ instead of $[\![b]\!]_\Sigma$ and let Σ be given by the context. ■

For instance, the semantics of the left (right) branch b_1 (b_2) in the tableau in Example 3.2 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 Γ of the whole signature Σ[†]. In the sequel, Γ always denotes some subset of the signature Σ. i.e. a subset of the atoms occurring in a clause set under consideration.

Definition 3.4 (Γ-Minimal Models)
For any atom set $\mathcal{J}$ define the *restriction of* $\mathcal{J}$ *to* Γ as $\mathcal{J}|\Gamma := \mathcal{J} \cap \Gamma$. In order to relate atom sets $\mathcal{J}_1$ and $\mathcal{J}_2$ define $\mathcal{J}_1 <_\Gamma \mathcal{J}_2$ iff $\mathcal{J}_1|\Gamma \subset \mathcal{J}_2|\Gamma$, and $\mathcal{J}_1 =_\Gamma \mathcal{J}_2$ iff $\mathcal{J}_1|\Gamma = \mathcal{J}_2|\Gamma$. As usual, the relation $\mathcal{J}_1 \leq_\Gamma \mathcal{J}_2$ is defined as $\mathcal{J}_1 <_\Gamma \mathcal{J}_2$ or $\mathcal{J}_1 =_\Gamma \mathcal{J}_2$. We say that a model $\mathcal{I}$ for a clause set D is Γ-*minimal (for* D*)* iff there is no model $\mathcal{I}'$ for D such that $\mathcal{I}' <_\Gamma \mathcal{I}$ ■

It is easy to see $\leq_\Gamma$ that is a partial order and $=_\Gamma$ 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 Σ-minimal one.

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

Theorem 3.1 (Model Soundness and Γ-Minimal Model Completeness)
Let T be a hyper tableau for a given ground clause set D. Then, for every finished open branch b in T it holds that $[\![b]\!]$ is a (not necessarily Γ-minimal) model for D (Model soundness).

[†] From a circumscriptive point of view [McC85], Γ is the set of atoms to be minimized, and $\Sigma \setminus \Gamma$ varies.

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

Notice that Theorem 3.2 does *not* give us that *only* Γ-minimal models are computed. We turn to this problem in Section 3.2.

PROOF. *Soundness direction:* let b in T be an open finished branch. Suppose, by way of contradiction, that $[\![b]\!]$ is not a model for D. Hence $[\![b]\!]$ is not a model for some clause $C = A_1 \vee \cdots \vee A_m \leftarrow B_1 \wedge \cdots \wedge B_n$. This means that $\{B_1, \ldots B_n\} \subseteq lit(b)$ and $\{A_1, \ldots, A_m\} \cap lit(b) = \emptyset$. Now, $m = 0$ is impossible, because otherwise with b being finished, b would have to be extended by C and b would have been closed 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 Γ-minimal model for D exists, the theorem holds vacuously. Otherwise let $\mathcal{I}$ be a Γ-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 $\Delta \subseteq [\![b]\!]$. It trivially holds that

$$D \cup \Delta \cup \neg\overline{\Delta} \text{ is satisfiable,} \tag{3.1}$$

where $\neg\overline{\Delta} := \{\neg A \mid A \in \overline{\Delta}\}$. Since $\mathcal{I}$ is a Γ-minimal model, Lemma 3.2 is applicable (in the "if" direction) and we conclude $D \cup \neg\overline{\Delta} \models \Delta$. This holds if and only if

$$D \cup \neg\overline{\Delta} \cup \{\bigvee_{A \in \Delta} \neg A\} \text{ is unsatisfiable.} \tag{3.2}$$

Hence, by refutational completeness of Hyper tableaux there is a refutation of this clause set. Further, by 3.1, the subset $D \cup \neg\overline{\Delta}$ is satisfiable. Hence, in any hyper tableau refutation the clause $\bigvee_{A \in \Delta} \neg A$ must be used at least 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. $\Delta \subseteq lit(b')$. 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 $\neg\overline{\Delta}$. 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 $lit(b') \subseteq lit(b)$ is formed. The reason is the following: otherwise every such extension b of b' would be closed, meaning that we could find a refutation of $D \cup \neg\overline{\Delta}$ alone, which by soundness of hyper tableau contradicts the satisfiability of $D \cup \neg\overline{\Delta}$. Thus, b is a branch with $\Delta \subseteq [\![b]\!]$. This concludes the proof of the first step.

Next, we show that for some branch b with $\Delta \subseteq [\![b]\!]$ we have $\overline{\Delta} \cap [\![b]\!] = \emptyset$. Suppose, by way of contradiction, that for every branch b with $\Delta \subseteq [\![b]\!]$ we have $\overline{\Delta} \cap [\![b]\!] \neq \emptyset$. Hence each such branch b can be closed with some literal $\neg A \in \neg\overline{\Delta}$. Thus we can find a refutation of $D \cup \neg\overline{\Delta}$ alone, which, by soundness of hyper tableaux contradicts the satisfiability of $D \cup \neg\overline{\Delta}$. Hence, $\overline{\Delta} \cap [\![b]\!] = \emptyset$ for some branch b. Since we presuppose $\Delta \subseteq [\![b]\!]$ we have together $[\![b]\!]|\Gamma = \Delta$. Since trivially $\mathcal{I}|\Gamma = \Delta$ we conclude $\mathcal{I} =_\Gamma [\![b]\!]$ as claimed. □

For example, since in the tableau in Example 3.2 every open branch is finished, one of its branches contains a Σ-minimal model (the literals $\{t, r, q\}$ in the left branch constitute a Σ-minimal model). The $\{t, r, q\}$-minimal models are $\{t, r, q\}$ and $\{t, r, p, q\}$, which both are computed.

3.2. Minimal Model Soundness

Our plan is to use the hyper tableau calculus for computing Γ-minimal models to ensure the rationality of the view deletion algorithm presented in Section 5.1. For this, we need a strengthening of Theorem 3.1 towards Γ-*minimal model soundness*, i.e. that *only* Γ-minimal models are returned as the result of the computation. This problem is known to be harder than just computing some model [EG93]. Consequently, it is not surprising that some non-trivial extra device is needed to guarantee minimal model soundness. Our proposed technique is based on the following lemma (similar results were given in [Rei87, Nie96c]):

Lemma 3.2 (Minimization Lemma)
Suppose Γ *is partitioned as* $\Gamma = \Delta \cup \overline{\Delta}$, *i.e.* $\Delta \cap \overline{\Delta} = \emptyset$. *Define* $\neg\Delta := \{\neg A \mid A \in \Delta\}$. *Let* D *be a set of clauses. Then*†

(1a) $D \cup \neg\overline{\Delta} \models \Delta$ *and (1b)* $D \cup \neg\overline{\Delta}$ *is satisfiable*

iff

(2a) $D \cup \neg\overline{\Delta} \cup \Delta$ *is satisfiable and (2b)* Δ *is* $\subseteq$*-minimal for this property.*

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

Proof. (1) $\Rightarrow$ (2): Suppose that (1a) and (1b) hold. Let $\mathcal{I}$ be a model for $D \cup \neg\overline{\Delta}$, which exists by (1b). By (1a), $\mathcal{I}$ is a model for Δ as well. Hence (2a) holds. It remains to show (2b). Suppose, by way of contradiction, that (2b) does not hold. Hence there is a partition $\Gamma = \Delta' \cup \overline{\Delta'}$ with $\Delta' \subset \Delta$ such that $D \cup \neg\overline{\Delta'} \cup \Delta'$ is satisfiable. Notice that $D \cup \neg\overline{\Delta'}$ is trivially satisfiable as well. The property $\Delta' \subset \Delta$ implies that $\neg\overline{\Delta} \subset \neg\overline{\Delta'}$. Let $\neg A \in \neg\overline{\Delta'} \setminus \neg\overline{\Delta}$ arbitrary, which exists because the inclusion is strict. We get immediately that $\neg\overline{\Delta} \cup \{\neg A\} \subseteq \neg\overline{\Delta'}$. From this and the just derived fact that $D \cup \neg\overline{\Delta'}$ is satisfiable, it follows that $D \cup \neg\overline{\Delta} \cup \{\neg A\}$ is satisfiable as well. In other words, $D \cup \neg\overline{\Delta} \not\models A$.

Now, since $\neg A \in \neg\overline{\Delta'} \setminus \neg\overline{\Delta}$ this means that $\neg A \notin \neg\overline{\Delta}$. By property of partition, $A \in \Delta$ follows. But with $D \cup \neg\overline{\Delta} \not\models A$ we get immediately that $D \cup \neg\overline{\Delta} \not\models \Delta$. This, however, is a contradiction to (1a). Hence, the assumption that (2b) does not hold must be wrong, and this direction of the lemma is proven.

(2) $\Rightarrow$ (1) Assume that properties (2a) and (2b) hold. Hence, (1b) follows trivially. Suppose, by way of contradiction, that (1a) does not hold. That is, $D \cup \neg\overline{\Delta} \not\models \Delta$. Then there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models D \cup \neg\overline{\Delta}$ and $\mathcal{I} \not\models \Delta$. Let $\overline{\Delta''} = \{A \in \Delta \mid \mathcal{I} \not\models A\}$, which must be non-empty, and let $\Delta'' = \Delta \setminus \overline{\Delta''}$. Notice that $(\overline{\Delta} \cup \overline{\Delta''}) \cup \Delta''$ partitions

† The notation $D \models \Delta$ means $D \models A$, for every $A \in \Delta$.

Γ in mutually disjoint parts. By construction, $\mathcal{I} \models \neg\overline{\Delta''}$ and $\mathcal{I} \models \Delta''$. Together then it holds that $\mathcal{I} \models D \cup \neg(\overline{\Delta} \cup \overline{\Delta''}) \cup \Delta''$. But now, since $\overline{\Delta''}$ is non-empty, it follows that $\Delta'' \subset \Delta$ and this inclusion is strict. So we arrive at a contradiction to (2b), by setting there $\Delta' = \Delta''$ and $\overline{\Delta'} = \overline{\Delta} \cup \overline{\Delta''}$. Hence, the assumption that (1a) does not hold must be wrong, and this direction of the lemma is proven as well. □

Now, Lemma 3.2 provides an important "tool" to achieve minimal model soundness: suppose we are given a hyper tableau with an open finished branch b. The literals on b represent a model $[\![b]\!]$ for the input clause set (cf. Def. 3.3). To determine whether $[\![b]\!]$ is a Γ-minimal model, build $\Delta = [\![b]\!]|\Gamma$ and let $\overline{\Delta} = \Gamma \setminus \Delta$. Now, by Lemma 3.2 and using the definition of Γ-minimal models (Def. 3.4), $[\![b]\!]$ is a Γ-minimal model for M iff $D \cup \neg\overline{\Delta} \models \Delta$. Since Δ represents in this context the conjunction of its members, equivalently $D \cup \neg\overline{\Delta} \cup \{\bigvee_{A \in \Delta} \neg A\}$ is unsatisfiable. Notice that this condition can be tested by a single theorem prover call, which is referred to as the "groundedness test". Interested readers are referred to [Nie96b, Nie96c] for more information on this technique of generating minimal models.

In [BY96] a calculus for minimal-model computation (not: Γ-minimal models) in the SATCHMO tradition [MB88] was proposed. The technique described there to ensure minimal-model soundness is different: minimal model soundness is ensured by the "complement splitting" technique (a kind of factorization) and incremental strategy to compare new minimal model candidates against previously computed minimal models. Thus, all computed minimal models have to be kept in memory (there can be exponentially many), whereas in our approach the test for minimal modelship can be carried out, as just explained, "locally" by a single theorem prover call.

A less general result wrt. minimal model reasoning than the one in Theorem 3.1 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.

4. Renaming and Update Tableaux

The key idea of the new algorithm is to transform the given database along with the view deletion request into a disjunctive logic program. 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. More precisely, one has to seek Γ-minimal models, where Γ will consist of the *negations* of the *EDB*-predicates; Γ-minimal models therefore characterize minimal deletion operations by reading the truth of a negative *EDB*-literal as a deletion.

Next, we are going to introduce the transformation technique — renaming — in a general way; then, it will be used below in a specific way within *update tableaux* for computation of view deletions.

Definition 4.1 (Renaming)
Let $C = \mathcal{A} \leftarrow \mathcal{B}$ be a clause, where $\mathcal{A}$ (resp. $\mathcal{B}$) is a disjunction (resp. conjunction) of

atoms, and let $S \subseteq \Sigma$ be a set of ground atoms. The *renaming of C wrt. S* is

$$C^S = \left(\bigvee_{A \in \mathcal{A}, A \notin S} A\right) \vee \left(\bigvee_{B \in \mathcal{B}, B \in S} \neg B\right) \leftarrow \left(\bigwedge_{B \in \mathcal{B}, B \notin S} B\right) \wedge \left(\bigwedge_{A \in \mathcal{A}, A \in S} \neg A\right)$$

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 use of the trivial fact that $\mathcal{I} \models C$ iff $\mathcal{I} \models C^S$.

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. Notice that $\neg\neg A$ means the negation of atom $\neg A$ then, but not double negation of A.

Henceforth, when the hyper tableaux calculus is applied to a renamed clause set, always the modified signature Σ^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 D^S. It can easily be converted into a model for the original clause set D by the following bijection between Σ-interpretations and Σ^S-interpretation:

PROPOSITION 4.1 (RENAMING MODELS)
If $\mathcal{I} \models D$ then $\mathcal{I}^S \models D^S$, where $\mathcal{I}^S(\neg A) = true$ iff $\mathcal{I}(A) = false$ for $A \in S$, and $\mathcal{I}^S(A) = true$ iff $\mathcal{I}(A) = true$ for $A \notin S$.

Conversely, If $\mathcal{I}^S \models D^S$ then $\mathcal{I} \models D$, where $\mathcal{I}(A) = true$ iff $\mathcal{I}^S(\neg A) = false$ for $A \in S$, and $\mathcal{I}^S(A) = true$ iff $\mathcal{I}(A) = true$ for $A \notin S$.

For instance, take $D = \{A \vee C \leftarrow B\}$, $S = \{A, B\}$ and $\mathcal{I} = \{\}$. Hence $D^S = \{C \vee \neg B \leftarrow \neg A\}$ and $\mathcal{I}^S = \{\neg A, \neg B\}$ is a model for D^S. This example also demonstrates that minimality of models is *not* preserved, because $\mathcal{I}^S$ is not a *minimal* model of D^S. Conversely, transforming back $\{\}$, which is the minimal model for D^S, yields $\{A, B\}$ which is not a *minimal* model for D [†]. Instead the following can be achieved:

LEMMA 4.2 (MAXIMAL MODELS BY RENAMING)
Let S be a set of atoms such that $\Gamma \subseteq S$, and let D^S be the renaming of a satisfiable clause set D wrt. S. Let $\Gamma^S = \{\neg A \mid A \in \Gamma\}$ and let for given interpretations $\mathcal{I}$ and $\mathcal{I}^S$

$$\begin{aligned} \Delta &= \{A \in \Gamma \mid \mathcal{I}(A) = true\} \\ \Delta^S &= \{\neg A \in \Gamma^S \mid \mathcal{I}^S(\neg A) = true\} \\ &= \{\neg A \in \Gamma^S \mid \mathcal{I}(A) = false\} \text{ (By Prop. 4.1 and using } \Gamma \subseteq S) \ . \end{aligned}$$

[†] One would have to have that $\Gamma \cap S = \emptyset$ to achieve that Γ-minimality of models is preserved. In the present paper, however, we do not need this result.

Then a Σ^S-interpretation $\mathcal{I}^S$ is a Γ^S-minimal model for $D^S \cup \Delta^S$ iff $\mathcal{I} \models D \cup \Delta$ and there is no set Δ' with $\Delta \subset \Delta' \subseteq \Gamma$ such that $D \cup \Delta'$ is satisfiable.

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

PROOF. We have

$\mathcal{I}^S$ is a Γ^S-minimal model for $D^S \cup \Delta^S$
$\Leftrightarrow$ $\mathcal{I}^S \models D^S \cup \{\neg A \in \Gamma^S \mid \mathcal{I}^S(\neg A) = \mathit{true}\} \cup \{\neg\neg A \mid \neg A \in \Gamma^S \text{ and } \mathcal{I}^S(\neg A) = \mathit{false}\}$ and the second set is minimal, and hence the third set is maximal
$\Leftrightarrow$ $\mathcal{I} \models D \cup \{\neg A \in \Gamma \mid \mathcal{I}(A) = \mathit{false}\} \cup \{A \mid A \in \Gamma \text{ and } \mathcal{I}(A) = \mathit{true}\}$ and the second set is minimal, and hence the third set is maximal (this follows immediately from the definition of $\mathcal{I}^S$)
$\Leftrightarrow$ $\mathcal{I} \models D \cup \Delta$ as claimed

□

We wish to apply the just presented renaming technique for our database update purposes. This is expressed in the next definition (cf. the beginning of Section 2.3 for our terminology).

DEFINITION 4.2 (*IDB** TRANSFORMATION)
Let $IDB \cup EDB$ be a given database. Let $S_0 = EDB \cup \{A \mid A \text{ is a ground } IDB \text{ atom}\}$. Then, IDB^* is defined as the renaming of IDB wrt S_0. ■

REMARK 4.3
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. Consequently, the request "delete A" will be expressed as $\neg A$.

Note that there are no facts in IDB^*, because we assume that IDB contains only definite program clauses. So when we add a delete request such as $\neg A$ to IDB^*, 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 4.4
Consider the following database:

$$IDB: \quad p \leftarrow t \qquad p \leftarrow q \wedge u \qquad q \leftarrow s \qquad u \leftarrow r$$

$$EDB: \quad t \leftarrow \qquad r \leftarrow$$

† Thus, even if we would allow facts $p \leftarrow$ in the IDB, this property would not be affected, because $p \leftarrow$ would be renamed to $\leftarrow \neg p$ in IDB^*.

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

$$\begin{array}{rrcl} IDB^*: & \neg t & \leftarrow & \neg p \\ & \neg q \vee \neg u & \leftarrow & \neg p \\ & & \leftarrow & s \wedge \neg q \\ & \neg r & \leftarrow & \neg u \end{array}$$

■

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 4.5 (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 $HS(b) = \{A \in EDB \mid \neg A \in b\}$. ■

The name "hitting set" is a misnomer here, but we use it in order to compare this approach with previous approaches that generate abductive explanations and a hitting set of them (cf. Section 5.2). This new approach directly generates a "hitting set" without enumerating all the explanations.

4.1. Relation to Diagnosis

Roughly, *diagnosis* is the task to determine plausible explanations for abnormal behavior. One of the main approaches is consistency-based diagnosis according to Reiter [Rei87] (see [CT91] for an overview of alternative diagnostic definitions). In this scenario, a model of a device under consideration is constructed and is used to predict its normal behavior. By comparing this prediction with the actual behavior it is possible to derive a diagnosis.

More precisely, this approach uses a logical description of the device, called the system description (*SD*), formalized by a set of first–order formulas (e.g. an electrical circuit). The diagnostic problem is described by a system description *SD*, a set *COMP* of components (e.g. the gates of the circuit) and a set *OBS* of observations (e.g. the states of input and output lines). With each component we associate a behavioral mode: $Mode(c, Ok)$ (abbreviated by $\neg Ab(c)$) means that component c is behaving correctly, while $Mode(c, Ab)$ (abbreviated by $Ab(c)$) denotes that c is faulty. Of course, the formulas in *SD* take these behavioral modes into account in the description of the components behavior.

DEFINITION 4.6 ([REI87])
A *diagnosis* of $(SD, COMP, OBS)$ is a set $D \subseteq COMP$, such that $SD \cup OBS \cup \{Ab(c) | c \in D\} \cup \{\neg Ab(c) | c \in COMP \setminus D\}$ is consistent. D is called a *minimal diagnosis*, iff it is a minimal set (wrt. $\subseteq$) with this property. ■

That is, a diagnosis D consists of components such that the observed behavior is consistent with the assumption that exactly the components in D are behaving abnormally.

In Reiter's seminal paper [Rei87] a technique is suggested that resembles very much Algorithm 1 above. The counterpart to our abductive explanations is called *failure set* in [Rei87], and the term *hitting set* is used by both. Diagnosis then are just hitting sets.

As with Algorithm 1, the drawback of this approach is its exponential space complexity. Therefore, in [BFFN97b, BFFN97a] it was suggested (among other improvements) to *directly* compute diagnosis (i.e. hitting sets) without the detour through failure sets (or abductive explanations). We used a special hyper tableau calculus for this, and to guarantee correctness, a specialized version of the above Minimization Lemma (Lemma 3.2) was used.

A direct translation of a view deletion problem into a diagnosis problems and is as follows: the *IDB* is considered as the system description *SD*, and each *EDB* atom is "normally" *true*. The idea is thus to translate the minimization of contractions of *EDB* atoms into minimal diagnosis. 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, they can be dispensed with by the following line of reasoning: in any *minimal* diagnosis these axioms of the form $\neg p \leftarrow Ab(p)$ are a consequence of the other clauses. To see this, 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 any approach to compute minimal diagnosis (such as the one in [BFFN97a]) can be used to compute minimal database contractions. However, to gain efficiency in the computation of diagnosis, and make the search for the diagnosis (i.e. database update) goal-oriented, an *initial interpretation* for renaming has to be used; this initial interpretation represents a model of the correctly functioning device under consideration. For the database update task, the interpretation S_0 used in the IDB^*-transformation above thus corresponds to the initial interpretation in the diagnosis domain.

4.2. COMPLETENESS OF UPDATE TABLEAUX

The completeness result for update tableaux developed now will be the basis for the correctness of the final algorithm in Section 5 (the intuition of which was given above in this section).

First, we have to eliminate a trivial form of update tableau:

Definition 4.7 (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 4.8
A special case comes up if an update tableau is trivial. For example, if $IDB = \{p \leftarrow q\}$ and $EDB = \{r \leftarrow \ \}$ then $IDB^* = \{\ \leftarrow \neg p \wedge q\}$. The update tableau for $IDB \cup EDB$ and delete request $\neg p$ consists of one open branch b, which is labeled with $\neg p$ (because $\leftarrow \neg p \wedge q$ is not applicable to $\neg p$). Thus, $HS(b) = \{\}$. Intentionally, this means that the delete request can be fulfilled 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 is compatible with the EDB. This is formalized below.

Lemma 4.3 (Trivial update tableaux)
Let T be a trivial update tableaux 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 tableaux 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 3.1), T must contain open branches. Further, every open branch must close with some clause from EDB^* (because otherwise, by completeness of hyper tableaux (Theorem 3.1), 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, by way of contradiction, 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 inconsistent as well. Contradiction. □

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

LEMMA 4.4
$IDB \cup EDB \cup \{\neg A\}$ is consistent iff there is no EDB-closed abductive explanation for A wrt. IDB.

PROOF. "Only-if"-direction: assume, by way of contradiction, 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, by way of contradiction, 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. □

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

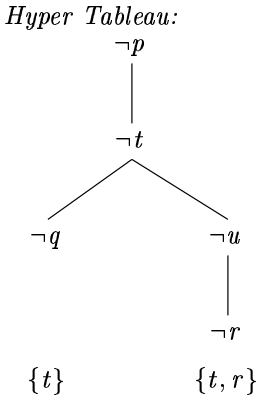
THEOREM 4.5 (COMPLETENESS OF UPDATE TABLEAUX)
Let T be a non-trivial update tableau for $IDB \cup EDB$ and delete request $\neg A$. Then for every minimal set $\alpha \subseteq EDB$ such that $\{\neg A\} \cup IDB \cup (EDB \setminus \alpha)$ is satisfiable there is an open finished branch b in T such that $HS(b) = \alpha$.

PROOF. We have that $\{\neg A\} \cup IDB \cup (EDB \setminus \alpha)$ is satisfiable, with α minimal as given, iff for some interpretation $\mathcal{I}$: $\mathcal{I} \models \{\neg A\} \cup IDB \cup (EDB \setminus \alpha)$ and $\mathcal{I}(A') = \mathit{false}$ for every $A' \in \alpha$. Recall from Definition 4.2 that $EDB \subseteq S_0$; hence Lemma 4.2 is applicable (in the if direction), setting there $S = S_0$, $\Gamma = EDB^{S_0}$, $\Delta = (EDB \setminus \alpha)$, $\Delta^{S_0} = \neg\alpha$, $D = \{\neg A\} \cup IDB$ and $\mathcal{I} = \mathcal{I}^{S_0}$. Hence, by Lemma 4.2, $\mathcal{I}^{S_0}$ is an EDB^{S_0}-minimal model for $\{\neg A\} \cup IDB^* \cup \neg\alpha$. $\mathcal{I}^{S_0}$ is an EDB^{S_0}-minimal model for $\{\neg A\} \cup IDB^*$ as well (*). For, suppose to the contrary, that there is a model $\mathcal{I}' <_{EDB^{S_0}} \mathcal{I}^{S_0}$ for $\{\neg A\} \cup IDB^*$. This means that there is a strict subset $\neg\alpha' \subset \neg\alpha$ such that $\mathcal{I}'(\neg A') = \mathit{true}$ for every $\neg A \in \neg\alpha'$ and $\mathcal{I}'(A') = \mathit{false}$ for every $A' \in EDB^{S_0} \setminus \neg\alpha'$. But then, we can apply Lemma 4.2 in the only-if direction by setting there $S = S^0$, $\Gamma = EDB^{S_0}$, $\Delta = (EDB \setminus \alpha')$, $\Delta^{S_0} = \neg\alpha'$, $D = \{\neg A\} \cup IDB$ and $\mathcal{I} = \mathcal{I}'$. However, this yields with $\neg\alpha' \subset \neg\alpha$ and hence $\alpha' \subset \alpha$ a contradiction to the maximality of Δ in Lemma 4.2. Hence, (*) holds.

Now that (*) is proven, by Theorem 3.1 there is an open finished branch b in T such that $[\![b]\!] =_{EDB^{S_0}} \mathcal{I}^{S_0}$. Further, $\mathcal{I}^{S_0} =_{EDB^{S_0}} \neg\alpha$, because $\neg\alpha = \Delta^S = \{\neg A \in EDB^{S_0} \mid \mathcal{I}^{S_0}(\neg A) = \mathit{true}$ (cf. Lemma 4.2). Hence, by Definition of $HS(b)$ (Def. 4.5) there is an open finished branch b in T such that $HS(b) = \alpha$. □

In summary, 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. Indeed, extracting updates from models of the transformed program does not necessarily result in a rational deletion, as the strong relevance property may be violated. The following example demonstrates this.

EXAMPLE 4.9
Let us continue with example 4.4. 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.



■

That means, we will have to modify the approach such that *soundness* will be achieved.

5. The Algorithm

We are turning to the just mentioned problem of achieving sound contractions. In brief, to filter out only the rational hitting sets, the postulate (KB—6.1) itself 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 5.1 (MINIMALITY TEST)
Let T be an update tableau for $IDB \cup EDB$ and delete request $\neg A$. We say that an open finished branch b in T *satisfies the minimality test* iff $\forall s \in HS(b) : IDB \cup EDB \setminus HS(b) \cup \{s\} \vdash A$. ■

DEFINITION 5.2 (UPDATE TABLEAU SATISFYING MINIMALITY)
An update tableau for given $IDB \cup EDB$ and delete request $\neg A$ is transformed into an update tableau *satisfying minimality* by marking every open finished branch as closed which does not satisfy minimality. ■

EXAMPLE 5.3
Continuing with the same example, after constructing the branch corresponding to the hitting set $\{t, r\}$, the minimality test is carried out as follows: It is checked if the resulting database with each member of the 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. $IDB \cup EDB \setminus \{t, r\} \cup \{r\} \nvdash p$, and hence this branch fails the minimality test. ■

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

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 minimality (cf. Definition 5.2) 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 4.5)

end.

Algorithm 2 generates a minimal hitting set in a generate-and-test manner: put a little bit more abstractly, branches b are enumerated, information is extracted from b and this information is tested to meet the desired minimality criterion. The important point is that only one branch at a time has to be kept in memory – the minimality test does not rely on previously computed information (this technique was first suggested in [Nie96b] to compute circumscription). Since the length of a branch is bounded by regularity to $|\Sigma|$, the space complexity is polynomial wrt. the size of the signature Σ. The worst case space complexity of Algorithm 1 and related approaches (such as the one in [Tom88]) that do not have this locality is exponential.

Concerning time complexity: minimal model computation is Π_2^p-complete [EG93]. We cannot expect a better behavior for Algorithm 2.

THEOREM 5.1 (CORRECTNESS AND COMPLETENESS OF ALGORITHM 2)
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), i.e. Algorithm 2 is sound. Further, any deletion that satisfies these postulates can be computed by this algorithm, i.e. completeness holds.

PROOF. Suppose as given a database $IDB \cup EDB$ and a ground view atom A to be deleted. Clearly, Algorithm 2 terminates due to the regularity restriction (cf. Section 3). Let $IDB \cup EDB \setminus HS(b)$ be the result of the computation, where b is a branch of the constructed update tableau T that satisfies minimality.

That the mentioned postulates (cf. Def. 2.2) are satisfied is obtained easily in most cases. Recall that the link between belief revision and database updates was given by identifying KB_I with IDB, KB_U with EDB and the contraction request α with the view atom A to be deleted. The whole knowledge base KB consists of $KB_I \cup KB_U$, i.e. $IDB \cup EDB$.

The postulate (KB—1) is satisfied trivially by Algorithm 2, because we do insert neither in EDB nor in IDB. If at all, *only* EDB shrinks, and therefore (KB—2) is satisfied.

Concerning (KB—3) suppose that $\alpha \notin Cn(KB)$. To show that (KB—3) is satisfied, we have to show that $HS(b) = \{\}$.

Equivalently to the given condition $\alpha \notin Cn(KB)$, the set $\{\neg A\} \cup IDB \cup EDB$ is satisfiable. Hence, by Lemma 4.3 the tableau T is trivial. So, T contains a branch b' with $HS(b') = \{\}$. This proves that a suitable branch with empty hitting set *can* be computed by Algorithm 2. It remains to be shown that indeed such a branch *must* be computed: suppose, by way of contradiction, that $HS(b) \neq \{\}$. We are given that b satisfies minimality. So there is an $s \in HS(b)$ such that $IDB \cup EDB \setminus HS(b) \cup \{s\} \models A$. Equivalently, $IDB \cup EDB \setminus HS(b) \cup \{s\} \cup \{\neg A\}$ is unsatisfiable. Since, $s \in HS(b)$ and trivially $EDB \setminus HS(b) \subseteq EDB$, the set $IDB \cup EDB \cup \{\neg A\}$ is unsatisfiable as well. This, however, is a plain contradiction to the given condition that $\{\neg A\} \cup IDB \cup EDB$ is satisfiable. Therefore, $HS(b) = \{\}$ as claimed, and so (KB—3) is satisfied.

Next we turn to (KB—4). The condition that $KB_I \not\vdash \alpha$ holds for all our delete requests, because α is an atom, and an atom never follows from a set of definite program clauses, which is just our assumption about IDB[†]. Thus we have to show that $\alpha \notin Cn(KB—\alpha)$, i.e. $IDB \cup EDB \setminus HS(b) \not\models A$.

We are given that b satisfies minimality. This implies that b is a finished open branch. By the soundness of hyper tableaux (Theorem 3.1) $[\![b]\!]$ is a model for the underlying clause set. For the case of update tableaux, this means that $[\![b]\!] \models IDB^* \cup \{\neg A \leftarrow\}$. Now consider the set $b \cap EDB^*$. By definition of branch semantics, all atoms in $b \cap EDB^*$ are *true* in $[\![b]\!]$, and all atoms in $EDB^* \setminus (b \cap EDB^*)$ are *false* in $[\![b]\!]$. Next we translate back the semantics of the renamed clause set to the original clause set. Recall that for the renaming transformation we use the set S^0 (cf. Def. 4.2) which includes EDB and also A. Hence, we can apply the second statement in Proposition 4.1 by setting there $S = S^0$, $\mathcal{I}^S = [\![b]\!]$ and $D^S = IDB^* \cup \{\neg A \leftarrow\}$, and conclude that there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models IDB \cup \{\leftarrow A\}$. Since, as obtained above, all atoms in $b \cap EDB^*$ are *true* in $[\![b]\!]$, and $HS(b)$ is just the renamed version of $b \cap EDB^*$, i.e. $\neg HS(b) = b \cap EDB^*$, we also have that every atom $A \in HS(b)$ is *false* in $\mathcal{I}$ (because the corresponding atom $\neg A \in b \cap EDB^*$ is *true* in $[\![b]\!]$). Similarly, since all atoms in $EDB^* \setminus (b \cap EDB^*)$ are *false* in $[\![b]\!]$, all atoms in $EDB \setminus HS(b)$ are *true* in $\mathcal{I}$. Together with $\mathcal{I} \models IDB \cup \{\leftarrow A\}$ this implies trivially $\mathcal{I} \models IDB \cup \{\leftarrow A\} \cup EDB \setminus HS(b)$. In other words, this set is satisfiable. Equivalently $IDB \cup EDB \setminus HS(b) \not\models A$, which was to be shown. Hence, (KB—4) is satisfied.

The next postulate is (KB—5). Since we deal with update request that are view atoms only, KB-equivalence is just syntactical equality among atoms. Thus, (KB—5) is trivially satisfied.

The final postulate shown to be satisfied is (KB—6.1). Hence let $\beta \in KB \setminus KB—\alpha$. We have to show that $\alpha \in Cn(KB—\alpha) \cup \{\beta\}$. Since $KB \setminus KB—\alpha$ is just the set of elements removed from KB_U, this means in terms of our algorithm, that for every $s \in HS(b)$ it holds that $IDB \cup EDB \setminus HS(b) \cup \{s\} \models A$. This property, however, is nothing but the minimality test (cf. Def. 5.1). Since b satisfies the minimality test, postulate (KB—6.1) is satisfied.

Since all postulates are now proven to be satisfied, the whole proof is completed. □

[†] Observe that every clause in $IDB \cup \{\leftarrow A\}$ contains a negative literal and hence is satisfiable.

5.1. Minimality Test Again

We are now going to relate the minimality test to a similar test from the literature. Interestingly, the minimality test of the previous section 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 its equivalence with the *minimality test* technique.

Definition 5.4 (Groundedness Test)
Let T be an update tableau for $IDB \cup EDB$ and delete request $\neg A$. We say that an open finished branch b in T *satisfies the groundedness test* iff $\forall s \in HS(b) : IDB^* \cup \{\leftarrow \neg B \mid B \in EDB \setminus HS(b)\} \cup \{\neg A \leftarrow\} \vdash \neg s \leftarrow$. ■

It is common to the groundedness test and the minimality test that they both ask for implications to be proven. Hence, any refutationally complete theorem prover (e.g. a hyper tableau prover) can be used as an implementation in both cases. 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^*. Since both tests are equivalent (cf. Proposition 5.2), this difference is irrelevant from a theoretical point of view. However, from a practical point of view there can well be a difference: typically, EDB will be very large, and $HS(b)$ will be comparably small. Hence $EDB \setminus HS(b)$ still is large. Now, in the groundedness test, $EDB \setminus HS(b)$ is translated into a set of *negative* unit clauses, whereas in the minimality test $EDB \setminus HS(b)$ is translated into a set of *positive* unit clauses. For a bottom-up calculus like hyper-tableaux it is advantageous to minimize the number of positive clauses, because they can instantly be applied and give rise for further inferences. On the other side, in the groundedness test the definite program IDB is translated into a *disjunctive* logic program. Since this causes branching in a bottom-up prover, the positive effect of having few positive unit clauses is compensated to some degree.

In the mentioned diagnosis application of hyper tableaux (Section 4.1), bringing in "semantics" by renaming turned out to improve efficiency dramatically [BFFN97a]. It is conceivable that the same holds for the comparable situation here, i.e. that the groundedness test is superior to the minimality test.

Next we will establish the "semantical" equivalence of both techniques. A further and natural characterization in terms of minimal models will be given as well.

Proposition 5.2
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 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 the groundedness test, as well as the minimality test achieve a minimization of the *deletion* of EDB-atoms such that consistency with a given delete request is recovered.

PROOF. (a) iff (b): We have

Branch b satisfies the groundedness test
$\Leftrightarrow$ $\forall s \in HS(b) : IDB^* \cup \{ \leftarrow \neg B \mid B \in EDB \setminus HS(b)\} \cup \{\neg A \leftarrow \} \vdash \neg s \leftarrow$
$\Leftrightarrow$ $\forall s \in HS(b) : \forall \mathcal{I}^{S_0} : \mathcal{I}^{S_0} \not\models IDB^* \cup \{ \leftarrow \neg B \mid B \in EDB \setminus HS(b)\} \cup \{\neg A \leftarrow \} \cup \{ \leftarrow \neg s\}$ (*)
$\Leftrightarrow$ $\forall s \in HS(b) : \forall \mathcal{I} : \mathcal{I} \not\models IDB \cup \{B \leftarrow \mid B \in EDB \setminus HS(b)\} \cup \{ \leftarrow A\} \cup \{s \leftarrow \}$
This equivalence holds by Proposition 4.1: for any pair of interpretations $\mathcal{I}^{S_0}$ for the previous line and $\mathcal{I}$ for this line: a renamed clause occurring in the previous line is *true* in $\mathcal{I}^{S_0}$ iff the corresponding clause in this line is *true* in $\mathcal{I}$.
$\Leftrightarrow$ $\forall s \in HS(b) : IDB \cup \{B \leftarrow \mid B \in EDB \setminus HS(b)\} \cup \{ \leftarrow A\} \vdash \leftarrow s$
$\Leftrightarrow$ $\forall s \in HS(b) : IDB \cup EDB \setminus HS(b) \cup \{s\} \vdash A$
$\Leftrightarrow$ b satisfies the minimality test.

(a) iff (c): We have

Branch b satisfies the groundedness test
$\Leftrightarrow$ $\forall s \in HS(b) : IDB^* \cup \{ \leftarrow \neg B \mid B \in EDB \setminus HS(b)\} \cup \{\neg A \leftarrow \} \vdash \neg s \leftarrow$
$\Leftrightarrow$ $\underbrace{IDB^* \cup \{\neg A \leftarrow \}}_{D} \cup \underbrace{\{ \leftarrow \neg B \mid B \in EDB \setminus HS(b)\}}_{\neg\overline{\Delta}} \vdash \underbrace{\{\neg s \leftarrow \mid s \in HS(b)\}}_{\Delta}$ (*)

The indicated definitions of D, $\neg\overline{\Delta}$ and Δ prepare for the application of Lemma 3.2. For the (a) $\Rightarrow$ (c) direction, Lemma 3.2 is to be applied in the only-if direction. To do so, we have to show first that $D \cup \neg\overline{\Delta}$ 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 $\leftarrow \neg B$ such that $B \in EDB \setminus HS(b)$ 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 $EDB \setminus HS(b)$ and in $HS(b)$. Thus, b remains an open branch when $\{\leftarrow \neg B \mid B \in EDB \setminus HS(b)\}$ would be added to the clause set $IDB^* \cup \{\neg A \leftarrow \}$. Further, b remains finished, because adding negative clauses does not enable other extension steps than those closing branches. Thus, together, $D \cup \neg\overline{\Delta}$ is satisfiable, because b constitutes a model for it.

Hence, Lemma 3.2 is applicable as suggested, and from line (*) we conclude that

$$\underbrace{IDB^* \cup \{\neg A \leftarrow \}}_{D} \cup \underbrace{\{ \leftarrow \neg B \mid B \in EDB \setminus HS(b)\}}_{\neg\overline{\Delta}} \cup \underbrace{\{\neg s \leftarrow \mid s \in HS(b)\}}_{\Delta} \quad (**)$$

is satisfiable and $HS(b)$ is minimal (in the sense stated in item (2b) in Lemma 3.2). From this minimality property it follows easily that $[\![b]\!]$ is an EDB^*-minimal model for $IDB^* \cup \{\neg A\}$ as claimed. Hence, the (a) $\Rightarrow$ (c) direction is proven.

The proof of the (c) $\Rightarrow$ (a) direction is quite similar: suppose that $[\![b]\!]$ is an EDB^*-minimal model for $IDB^* \cup \{\neg A\}$. Then, line (**) is satisfiable, and $HS(b)$ is minimal (in the sense stated in item (2b) in Lemma 3.2) due to the EDB^*-minimality of $[\![b]\!]$. But then, Lemma 3.2 is applicable in the if-direction, and we conclude line (*). By the equivalences preceding line (*), b satisfies the groundedness test. □

5.2. An Alternative Proof

Now we follow an alternative way to show the rationality of Algorithm 2. We do this in order to gain more insight on how Algorithm 2 is related to the previous approach presented in Section 2.2 that generates (abductive) explanations and computes 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 5.5 (*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 at least negated *EDB*-atom then.

Example 5.6 (*EDB*-cut)
Consider this database:

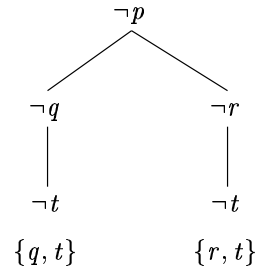
$$IDB: \quad p \leftarrow q \wedge r \qquad\qquad EDB: \quad q \leftarrow$$
$$p \leftarrow t \qquad\qquad r \leftarrow$$
$$t \leftarrow$$

So, $S^0 = \{p, q, r, t\}$ and IDB^* is as follows:

$$IDB^*: \quad \neg q \vee \neg r \leftarrow \neg p$$
$$\neg t \leftarrow \neg p$$

An update tableau for delete request $\neg p$ is shown on the right. Below the leaves of the branches are the hitting sets (both satisfy minimality). All the *EDB*-cuts are displayed on the right.

Hyper Tableau:



EDB-cuts:
$\{q, r\}$
$\{t, r\}$
$\{q, t\}$
$\{t\}$

■

Readers familiar with abduction might have already realized that an *EDB*-cut across the tableau constitutes an explanation for the view atom being deleted. More precisely:

Lemma 5.3
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 Δ. 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 $\Delta^* := \{ \ \leftarrow \neg A \mid A \in \Delta\}$. 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 tableau, T is a hyper tableau for just this set.

By soundness of hyper tableaux (Theorem 3.1), 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 $\ \leftarrow \neg A_i \in \Delta^*$. This holds, because otherwise, by completeness of hyper tableaux (Theorem 3.1), we would have a contradiction to the inconsistency of $IDB^* \cup \Delta^* \cup \{\neg A \leftarrow \ \}$. Furthermore, each of the negative unit clauses in Δ^* 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 Δ, as stated in (*). Thus, in other words, collecting the respective EDB-atoms from the b_i's gives the EDB-cut Δ of T.

The argumentation for the second item is similar. Let $\Delta' \in S'$ be an arbitrary element in $S\prime$. 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 $\Delta^* := \{ \ \leftarrow \neg A \mid A \in \Delta'\}$ 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), Δ' is an (EDB-closed) abductive explanation for A by Def. 2.3 (notice that KB_I there is just IDB here). Trivially Δ' contains a minimal such explanation Δ. That is, $\Delta \in S$ as desired. □

Remark 5.7
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 minimality. The canonical counterexample is in the preceding Example 5.6. Now consider the atom p. There are two minimal abductive explanations for p, which are $\{t\}$ and $\{q, r\}$. An update tableau (even satisfying minimality) for $\neg p$, however, will also admit the EDB-cuts $\{q, t\}$ and $\{r, t\}$, as displayed in the tableau in Example 5.6. This example demonstrates that computing minimal hitting sets and computing minimal abductive explanations are rather different things. ■

The above lemma precisely characterizes which 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 5.4 ([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 5.5
Let T be a non-trivial update tableau for $IDB \cup EDB$ and a delete request $\neg A$ that satisfies the 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 5.3 and Lemma 5.4. □

From the belief dynamics results recalled in Section 2 (Algorithm 1 and Theorem 2.1), it follows that Algorithm 2 is rational. Thus, another proof of our main result Theorem 5.1 along these lines can be given.

6. Concluding Remarks

We have presented an algorithm for deleting a view atom from a definite database.

Moreover, we have shown in Theorem 5.1 that this algorithm is rational in the sense that it satisfies the rationality postulates that are justified from a philosophical angle.

The key idea of our approach is to transform the given database into a disjunctive logic program in such a way that updates can be extracted from the models of this transformed program. More precisely, every *minimal* model wrt. a certain set S^0 of predicates represents the actions to be taken to satisfy the update. Thus, we have also shown that deleting a view atom can be rephrased as a circumscription problem, where S^0 are the predicates to be minimized, and all other predicates vary. At the heart of our algorithm is a minimality test that cancels non-rational update candidates. We showed that this minimality test is equivalent to the groundedness test known from the literature. Furthermore, we have sketched how a diagnosis machine can be used to compute view updates.

In contrast to previous approaches for view updates, our algorithm is of polynomial space complexity. The reason is, that the new algorithm does *not* compute possibly exponentially many abductive explanations in a first step and then extract updates by means of a minimal hitting set.

Instead, a candidate update is directly extracted from a model construction for the transformed clause, which is done by the hyper tableaux calculus. Hyper tableaux need to consider only one branch at a time, and the length of each branch is bounded by the regularity condition by the number of atoms in the input language. The final minimality test where update *candidates* have to qualify as *minimal* updates does not influence this polynomial space complexity.

As a by-product when investigating the relation to previous approaches, we got a new soundness and completeness result (Lemma 5.3) for abductive explanation computation with hyper tableaux.

Our approach works on the assumption that the *EDB* is available and the complete *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.

Acknowledgments

The authors would like to thank all the members of the Artificial Intelligence Research Group at the University of Koblenz, Germany, for stimulating discussions on this topic. This research is a part of projects on Automated Deduction and Disjunctive Logic Programming funded by DFG (grants `Fu 263/2-2` and `Fu 263/3-1` respectively). Three reviewers gave very helpful suggestions for improvements.

References

[AB97] Chandrabose Aravindan and Peter Baumgartner. A Rational and Efficient Algorithm for View Deletion in Databases. In Jan Maluszynski, editor, *Logic Programming - Proceedings of the 1997 International Symposium*, Port Jefferson, New York, 1997. The MIT Press.

[Abi88] S. Abiteboul. Updates: A new frontier. In M. Gyssens, J. Paredaens, and D. Van Gucht, editors, *Proceedings of the second international conference on database theory*, volume Lecture Notes in Computer Science 326, pages 1–18. Springer-Verlag, 1988.

[AD95] Chandrabose Aravindan and Phan Minh Dung. Knowledge base dynamics, abduction, and database updates. *Journal of Applied Non-Classical Logics*, 5(1):51–76, 1995.

[AGM85] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

[Ara95] Chandrabose Aravindan. *Dynamics of Belief: Epistemology, Abduction, and Database Updates*. PhD thesis, Computer Science Program, Asian Institute of Technology, Bangkok, Thailand, 1995.

[Bau98] Peter Baumgartner. Hyper Tableaux — The Next Generation. In Harry de Swaart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, number 1397 in Lecture Notes in Artificial Intelligence. Springer, 1998.

[BFFN97a] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, and Wolfgang Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. In *15th International Joint Conference on Artificial Intelligence (IJCAI 97)*, pages 460–465, Nagoya, 1997. International Joint Conference on Artificial Intelligence.

[BFFN97b] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, and Wolfgang Nejdl. Tableaux for Diagnosis Applications. In Didier Galmiche, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, number 1227 in Lecture Notes in Artificial Intelligence, pages 76–90. Springer, 1997.

[BFN96] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper Tableaux. In *Proc. JELIA 96*, number 1126 in Lecture Notes in Artificial Intelligence. European Workshop on Logic in AI, Springer, 1996.

[BJdR97] Patrick Blackburn, Jan Jaspers, and Maarten de Rijke. Reasoning about changing information. *South African Computer Journal*, 19:2–26, 1997.

[Bry90] François Bry. Intensional updates: Abduction via deduction. In D. H. D. Warren and P. Szeredi, editors, *Proceedings of International Conference on Logic Programming*, pages 561–575. The MIT Press, 1990.

[BT98] François Bry and Sunna Torge. A Deduction Method Complete for Refutation and Finite Satisfiability. In *Proc. 6th European Workshop on Logics in AI (JELIA)*, LNAI. Springer, 1998.

[BY96] François Bry and Adnan Yahya. Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 1071 in Lecture Notes in Artificial Intelligence, pages 143–159. Springer, 1996.

[CL73] C. Chang and R. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.

[CT91] Luca Console and Pietro Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.

[DB82] U. Dayal and P. A. Bernstein. On the correct translation of update operations on relational views. *ACM Transactions on Database Systems*, 8(3):381–416, 1982.

[Dec90] Hendrik Decker. Drawing updates from derivations. In *Proceedings of the Third International Conference on Database Technology.* Springer-Verlag, 1990.

[Dec96] Hendrik Decker. An extension of SLD by abduction and integrity maintenance for view updating in deductive databases. In Michael Maher, editor, *Proceedings of Joint International Conference and Symposium on Logic Programming*, pages 157–169. The MIT Press, 1996.

[EG93] Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed world reasoning are π_2^p-complete. *Theoretical Computer Science*, 114:231–245, 1993.

[Fit90] M. Fitting. *First Order Logic and Automated Theorem Proving.* Texts and Monographs in Computer Science. Springer, 1990.

[FH91] H. Fujita and R. Hasegawa. A Model Generation Theorem Prover in KL1 using a Ramified-Stack Algorithm. In *Proc. of the Eighth International Conference on Logic Programming*, pages 535–548, Paris, France, 1991.

[Gär92] P. Gärdenfors. Belief Revision: An Introduction. In P. Gärdenfors, editor, *Belief Revision*, pages 1–28. Cambridge University Press, 1992.

[GL90] A. Guessoum and J. W. Lloyd. Updating knowledge bases. *New Generation Computing*, 8, 1990.

[GL91] A. Guessoum and J. W. Lloyd. Updating knowledge bases II. *New Generation Computing*, 10, 1991.

[GM95] Ashish Gupta and Inderpal Singh Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE DE Bulletin*, 18(2):3–19, 1995.

[GR95] P. Gärdenfors and H. Rott. Belief Revision. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in AI and Logic Programming*, volume IV: Epistemic and Temporal Reasoning, pages 35–132. Oxford University Press, 1995.

[Han91a] S. O. Hansson. *Belief base dynamics.* PhD thesis, Uppsala University, Sweden, 1991.

[Han91b] S. O. Hansson. Belief contraction without recovery. *Studia Logica*, 50(2):251–260, 1991.

[Kel85] A. M. Keller. Algorithms for translating view updates to database updates for views involving selections, projections, and joins. In *Proceedings of the Fourth ACM Symposium on Prinsciples of Database Systems*, pages 154–163. ACM, 1985.

[KM90] A. C. Kakas and P. Mancarella. Database updates through abduction. Technical report, Department of Computing, Imperial College, London, U.K., 1990.

[Lan90] R. Langerak. View updates in relational databases with an independent scheme. *ACM Transactions on Database Systems*, 15(1):40–66, 1990.

[Llo87] J. W. Lloyd. *Foundations of Logic Programming.* Springer–Verlag, second extended edition, 1987.

[MB88] Rainer Manthey and François Bry. SATCHMO: a theorem prover implemented in Prolog. In Ewing Lusk and Ross Overbeek, editors, *Proceedings of the 9^{th} Conference on Automated Deduction, Argonne, Illinois, May 1988*, volume 310 of *Lecture Notes in Computer Science*, pages 415–434. Springer, 1988.

[McC85] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1985.

[Nie96b] Ilkka Niemelä. Implementing circumscription using a tableau method. In W. Wahlster, editor, *Proceedings of the 12^{th} European Conference on Artificial Intelligence*, pages 80–84. John Wiley & Sons Ltd, 1996.

[Nie96c] Ilkka Niemelä. A tableau calculus for minimal model reasoning. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Proceedings of the fifth workshop on theorem proving with analytic tableaux and related methods*, number 1071 in Lecture Notes in Artificial Intelligence, pages 278–294. Springer-Verlag, 1996.

[Poo89] David Poole. Explanation and prediction: An architecture for default and abductive reasoning. *Computational Intelligence*, pages 97–110, 1989.

[Rei87] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, April 1987.

[Rob65] J. A. Robinson. Automated deduction with hyper-resolution. *Internat. J. Comput. Math.*, 1:227–234, 1965.

[Tom88] A. Tomasic. View update translation via deduction and annotation. In M. Gyssens, J. Paredaens, and D. van Gucht, editors, *Proceedings of the International Conference on Database Technology*, volume Lecture Notes in Computer Science 326, pages 338–352. Springer-Verlag, 1988.