Generalized Query Answering in Disjunctive Deductive Databases: Procedural and Nonmonotonic Aspects

Adnan H. Yahya
yahya@ee.birzeit.edu

Electrical Engineering Department, Birzeit University, Birzeit, Palestine

Abstract. Generalized queries are defined as sets of clauses in implication form. They cover several tasks of practical importance for database maintenance such as answering positive queries, computing database completions and integrity constraints checking. We address the issue of answering generalized queries under the minimal model semantics for the class of Disjunctive Deductive Databases (DDDBs). Our approach is based on having the query induce an order on the models returned by a sound and complete minimal model generating procedure. We consider answers that are true in *all* and those that are true in *some* minimal models of the theory and investigate the monotonicity properties of the different classes of queries and answers.

1 Introduction

Minimal model semantics was one of the first to be defined for disjunctive theories[12, 10]. Several model classes defined under other semantics, such as the *perfect* and *stable* models for theories with body negation, are subsets of the minimal models and coincide with the minimal models in the absence of body negation.

Minimal models proved important for defining database completion: the mechanism to avoid the explicit storage of negative data. The Closed World Assumption and its extensions to disjunctive theories were defined in terms of minimal models [14, 12, 20]. Limiting our attention to the class of minimal models reconciles the concepts of derivability in all models and in all minimal models of the completed theory for positive and negative formulas [16, 20].

In this paper we consider several aspects of generalized query answering based on minimal model generation. The classes of queries considered are of importance for database maintenance and exploitation. Our approach is based on having the query induce an order on the models returned by a sound and complete minimal model generation procedure. This order is used to answer the query and to decide the monotonicity of the answers returned for the query under consideration.

The rest of the paper is organized as follows. In the next section we give some relevant definitions and describe a sound and complete minimal model generating procedure that will be used for query answering. We define the concept of a

generalized query and two classes of answers: those *true* in *all* minimal models and those that are *true* in *some* minimal models. In Section 3 we show how to use a minimal model generating procedure for generalized query answering. In Section 4 we discuss the monotonicity properties of the generalized query answering process for the classes of queries and answers considered. In Section 5 we give our conclusions and mention some possible directions for further research.

2 Preliminaries and Background Material

We assume familiarity with the basic concepts as in [10] and limit ourselves to briefly recalling the basic material needed for the results presented here.

Definition 1. (DDDB) A *disjunctive deductive database (DDDB)*, DB, is a set of clauses in implication form: $C = A_1 \vee \cdots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$, where $m, n \geq 0$ and the A_i and B_j are atoms in a First Order Language (FOL) $\mathcal{L}$ with no function symbols. C is positive if $n = 0$ (head is $\top$, *true*, empty) and negative or denial if $m = 0$ (body is $\bot$, *false*, empty). By $Head(C)$ we denote the disjunction of atoms $A_1 \vee \cdots \vee A_m$ and by $Body(C)$ we denote the conjunction of atoms $B_1 \wedge \ldots \wedge B_n$. So $C = Head(C) \leftarrow Body(C)$.

The Herbrand base of DB, HB_{DB}, is the set of all ground atoms that can be formed using the predicate symbols and constants in $\mathcal{L}$. A *Herbrand interpretation* is any subset of HB_{DB}. A Herbrand model of DB, M, is a Herbrand interpretation such that $M \models DB$ (all clauses of DB are *true* in M). M is *minimal* if no proper subset of M is a model of DB. The set of all minimal models of DB is denoted by $\mathcal{MM}(DB)$.

Definition 2. (range-restriction) A clause C is range-restricted if every variable occurring in the head of C also appears in the body of C. A database is range-restricted if and only if all its clauses are range-restricted.

In this paper we assume the theory to be range-restricted (RR).

Definition 3. (closed world assumption)[12, 20] Let DB be a DDDB. Then $CWA(DB) = \{\neg A_1 \vee \cdots \vee \neg A_n | A_i \in HB_{DB}$ and $n > 0$ and $\nexists$ a minimal model of DB, M such that $\{A_1, \cdots, A_n\} \subseteq M\}$. n always equal to 1 gives the GCWA and allowing arbitrary values for n results in EGCWA.

The completed database refers to the set of positive and negative ground clauses derivable directly from DB or by the appropriate default rule for negation. We adopt the EGCWA because of the following result:

Lemma 4. *[20] Let DB be a DDDB. Then $DB^c = DB \cup EGCWA(DB)$ has as its models the set of minimal models of DB. That is, $M \models DB^c$ iff $M \in \mathcal{MM}(DB)$.*

Definition 5. If $C = A_1 \vee \ldots \vee A_n$ is a disjunction of atoms, then by $Neg(C)$ we denote the set of clauses in implication form $Neg(C) := \{A_1 \rightarrow \bot, \ldots, A_n \rightarrow \bot\}$. If $M = \{A_1, \ldots, A_n\}$ is a finite interpretation then $Neg(M)$ denotes the clause in implication form $Neg(M) = A_1 \wedge \ldots \wedge A_n \rightarrow \bot$.

2.1 Model Generation

The main results of this paper are based on using a model generating procedure [3, 19]. First we give a brief description of a minimal model generating procedure that is sound and complete [3]: it returns all and only minimal models of its input theory. Given a DDDB, DB, the procedure constructs a (model) tree with the atomic clauses in each root-to-leaf branch representing a minimal model of DB. Starting from $\top$ (*true*) at the root, the procedure expands a tree for DB, by applying the following expansion rules [3]:

Definition 6. (expansion rules) Let DB be a DDDB. If the elements above the horizontal line are in a branch $\mathcal{B}$ then $\mathcal{B}$ can be expanded by the elements below the line in each of the following rules.

Positive unit hyper-resolution (PUHR):

$$\frac{\begin{array}{c} B_1 \\ \vdots \\ B_n \end{array}}{E\sigma}$$

Complement-Splitting:

$$\frac{E_1 \vee E_2}{\begin{array}{c|c} E_1 & E_2 \\ [Neg(E_2)] & \end{array}}$$

where σ is a most general unifier of the body of a clause $(A_1 \wedge ... \wedge A_m \rightarrow E) \in DB$ with $\{B_1, ..., B_n\}$. That is, $\{A_1, ..., A_m\}\sigma = \{B_1, ..., B_n\}$.

Range-restriction ensures that splitting is applied only to *ground* disjunctions.

Definition 7. (model tree) A Model Tree for a DDDB, DB, is a tree structure the nodes of which are (sets of) ground atoms, disjunctions and denials constructed as follows:

1. $\{\top\}$ is the top (root) node of the tree.
2. If T is a leaf node in the tree for DB, such that an application of the PUHR rule (respectively complement splitting rule) is possible to yield a formula E (resp. two formulas E_1 and E_2) not subsumed by an atom already in the branch, then the branch is extended by adding the child node E (resp. the two children nodes $\{E_1, Neg(E_2)\}$ and E_2) as successor(s) to T.

We always select E_1 for splitting a disjunction $(E_1 \vee E_2)$ to be atomic and expand the leftmost atom of a disjunction first. As a result atoms of the clause are expanded from left to right. Our interest is only in branches with no occurrences of *false* ($\bot$), that is, open branches. The branch expansion is stopped when ($\bot$) is added (the branch closes). The expansion continues until no new expansions are applicable (all open branches are saturated). A branch represents the interpretation in which all (ground) unit clauses are assigned the truth value *true*. For the class of RR DDDBs the procedure is model *sound* in the sense that all tree branches represent models of the theory and *complete* in the sense that the tree has at least one branch representing each minimal model of DB. The

first (leftmost) model generated by the procedure is minimal and no duplicates are produced. However, not all branches represent minimal models [3].

If additionally, for each minimal model generated so far, M, we augment the theory by the negation of M, $(< Neg(M) >)$ for subsequent processing steps, then we achieve a model generating procedure that is *minimal model sound* and *complete*. It returns all and only minimal models of its input theory. [3] contains a Prolog implementation of the procedure, called MM-Satchmo.

Example 1. Figure 1 shows the search spaces of MM-Satchmo for $DB = \{$

$$\begin{array}{ll} \top \rightarrow P(a) \vee P(b) & \qquad P(a) \rightarrow P(b) \vee P(d) \\ \top \rightarrow P(a) \vee P(c) & \qquad P(b) \rightarrow P(a) \vee P(d)\} \end{array}$$

Some nonminimal models were deleted by complement splitting ($\bot$ enclosed in square brackets [ ]) and others by model minimization ($\bot$ enclosed in <>). The minimal model tree construction is depicted in Figure 1. All and only minimal models are returned and are represented by the open branches of the tree. $\mathcal{MM}(DB) = \{\{P(a), P(d)\}, \{P(a), P(b)\}, \{P(b), P(c), P(d)\}\}$.

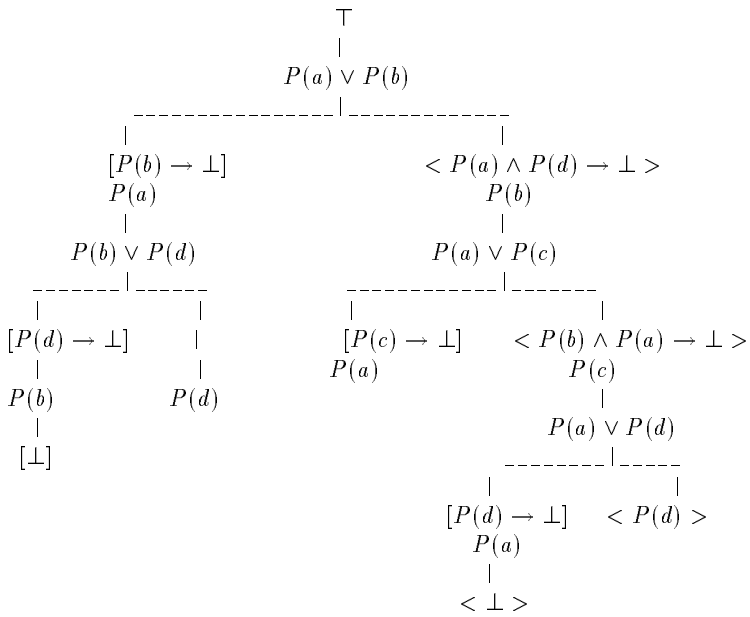

Fig. 1. A Run of the Model Generator MM-Satchmo for Example 1.

2.2 Queries, Answers and the Minimal Model Semantics

We are interested in *yes/no* answers to generalized ground queries which are defined as follows:

Definition 8. (elementary generalized query) An elementary generalized query is a ground clause in implication form: it is positive if the body is empty, negative if the head is empty and mixed otherwise [17].

Definition 9. (positive/negative queries) A query Q is positive (negative) if it can be translated into a set of positive (negative, denial) clauses. Q is mixed if it is neither positive nor negative.

Atomic, conjunctive and disjunctive queries are all positive queries. For a query Q, by $\{Q\}$ and $Neg(Q)$ we denote the set of clauses that represent Q and the negation of Q, respectively.

Definition 10. (answers) Let DB be a DDDB and let Q be a ground query.

- Q is a SURE answer in DB iff Q is true in all minimal models of DB.
  Q is minimal if, additionally, no proper subset of Q is a SURE answer.
- Q is a $MAYBE$ answer in DB iff Q is true in some minimal models of DB.

Clearly, every component of a minimal SURE answer is also a $MAYBE$ answer and every MAYBE answer is a component of a (minimal) SURE answer to a query.

Lemma 11. *Let DB be a DDDB, I be an interpretation and $\mathcal{C}$ be a set of ground denial rules (constraints): $\mathcal{C} = \{C : A_1 \wedge .... \wedge A_n \rightarrow \bot$, where A_i are ground atoms, $i = 1...n$ for some $n\}$; then:*

1. *If $\mathcal{C}$ is violated in I then it is also violated in all supersets of I. That is, if $I \not\models \mathcal{C}$ then $I' \not\models \mathcal{C}$, for all I' such that $I \subset I'$.*
2. *Assume $I \models \mathcal{C}$. Then, $I \models DB \cup \mathcal{C}$ iff $I \models DB$ and $I \not\models DB \cup \mathcal{C}$ iff $I \not\models DB$.*

Proof. Immediate.

As a counterexample for the case of nondenial rules consider $DB = \{P(a)\}$, the single rule $P(a) \rightarrow P(b)$ and the interpretations $\{P(a)\}$ and $\{P(a), P(b)\}$. Only the latter satisfies the constraint.

Theorem 12. *Let DB be a DDDB and $\mathcal{C}$ be a set of denial rules. Then:*

1. *If M is a minimal model for $DB \cup \mathcal{C}$ then M is a minimal model for DB.*
2. *$\mathcal{MM}(DB \cup \mathcal{C}) = \mathcal{MM}(DB) \setminus \{M : M \not\models \mathcal{C}\}$.*
3. *If $\mathcal{C} = \{Neg(M) | M \in \mathcal{MM}(DB)\}$ then $(DB \cup \mathcal{C})$ is inconsistent: (has no models and $\mathcal{MM}(DB \cup \mathcal{C}) = \emptyset$).*
4. *If $\mathcal{C}_1, ..., \mathcal{C}_n$ are sets of denial rules such that $\mathcal{C}_n \subseteq ... \subseteq \mathcal{C}_1$. Then: $\mathcal{MM}(DB \cup \mathcal{C}_1) \subseteq ... \subseteq \mathcal{MM}(DB \cup \mathcal{C}_n)$.*

Proof. Straightforward.

Theorem 12 shows that adding denial constraints can change the status of models to nonmodels but cannot affect model minimality.

3 Query Answering

Using the semantic characterization of query answers (Definition 10) we try to reduce the process of query answering to an invocation of a sound and complete minimal model generating procedure (e.g. MM-Satchmo [3]). This can be done in two ways:

The first is to use a static representation of the theory in terms of its minimal models, say in the form of a minimal model tree. The query answering is converted into searches in the tree [5, 19]. The minimal model generating procedure is used to construct such a tree and the representation is independent of the query. Special arrangements such as indexing or tree restructuring are needed to facilitate the search for elements of the query in the tree. However, if the theory changes state then the model generating procedure can be used to regenerate the minimal model structure of the updated theory. If updates are frequent then reconstructing the minimal model tree may become costly. Another drawback is that one may need to store two representations of the theory: the original (clausal) and the minimal model representation, since the two representations are only minimal model equivalent in the sense that they have the same set of minimal models but are not equivalent in the more general sense as demonstrated by the following example:

Example 2. Consider the DDDB, $DB = \{P(c), P(a) \rightarrow P(b)\}$ with the only minimal model $\{P(c)\}$. Updating DB by adding $P(a)$ will result in minimal model sets: $\{P(c), P(a)\}$ and $\{P(c), P(a), P(b)\}$ for the minimal model and clausal representations of the (original and updated) theory, respectively.

The second way is to retain only the clausal representation and generate the minimal models, possibly in a query induced order, at query answering time. In this paper we concentrate on the last approach.

3.1 Answering Positive and Negative Queries

The standard approach for query answering is to try to refute the theory augmented by the negation of the query. For positive queries, minimal model reasoning is the same as reasoning under "all models semantics". It was shown that a complete minimal model generating procedure is sound and complete for refutations (for DDDB's) [11, 3]. However, minimal model generation produces information that can be used to enrich the query answering process.

Theorem 13. *Let DB be a DDDB and Q be a positive query. Then:*
$\mathcal{MM}(DB) = Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB \cup \{Q\}))$,
where $Min(S)$ returns the set of minimal elements of the set S.

Proof. ($\rightarrow$) Let $M \in \mathcal{MM}(DB)$. Either $M \models Q$ and $M \not\models Neg(Q)$: $M \in \mathcal{MM}(DB \cup \{Q\})$ and is also in $Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB \cup \{Q\}))$.

Or else $M \models Neg(Q)$ and $M \not\models Q$. $M \in \mathcal{MM}(DB \cup Neg(Q))$ and is also in $Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB) \cup \{Q\}))$ by Theorem 12.

($\leftarrow$)Let $M \in \mathcal{MM}(DB \cup \{Q\})$. Two cases are possible: $M \in \mathcal{MM}(DB)$ and $M \notin \mathcal{MM}(DB \cup Neg(Q))$ and therefore $M \in Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB \cup \{Q\}))$. Or else, M is a nonminimal model of DB. There exists $M_1 \subset M$ such that $M_1 \in \mathcal{MM}(DB)$. $M_1 \not\models Q$. $M_1 \models Neg(Q)$. $M_1 \in \mathcal{MM}(DB \cup Neg(Q))$. $M_1 \in Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB \cup \{Q\}))$.

If $M \in \mathcal{MM}(DB \cup Neg(Q))$ then it is also a minimal model of DB by Theorem 12 since $Neg(Q)$ consists entirely of denial rules.

Example 3. Let $DB = \{P(a) \rightarrow P(b)\}$ and $Q = P(a)$. DB has the only minimal model $\{\}$. The minimal model for $DB \cup \{\neg P(a)\}$ is $\{\}$ while the minimal model for $DB \cup \{P(a)\}$ is $\{P(a), P(b)\}$ which is subsumed by $\{\}$. $\mathcal{MM}(DB) = Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB \cup \{Q\})) = \{\}$.

For positive queries model subsumption, if any, is unidirectional: minimal models of $DB \cup Neg(Q)$ can subsume (be a subset of) minimal models of $DB \cup \{Q\}$ but not the reverse. This is so since a model of $DB \cup Neg(Q)$ has no elements of Q while $DB \cup \{Q\}$ must have some. Theorem 13 suggests a simple procedure for answering positive queries by partitioning the set of minimal models of DB into two sets: one in which Q is *true* and the other in which Q is *false* then check for model minimality. Our way is to run two MM-Satchmo processes:

- The first process of MM-Satchmo will operate on $DB \cup Neg(Q)$. We denote the (possibly empty) set of minimal models returned by $\mathcal{MM}(DB)_{Neg(Q)}$.
- The second will operate on the set union of the theory DB, the query Q and the constraints corresponding to the minimal models returned by the first process. That is, it operates on $DB \cup \{Q\} \cup \{Neg(M) | M \in \mathcal{MM}(DB)_{Neg(Q)}\}$. We call the (possibly empty) set of minimal models returned $\mathcal{MM}(DB)_{\{Q\}}$.

The constraints in the second process are used to remove the models that satisfy Q but are not minimal for DB alone. The two processes are **not** independent. While we can avoid adding $\{Q\}$ in the second process we can use it to impose an order on the set of minimal models generated in the second branch. The entire process is equivalent to augmenting DB with the clause $\neg Q \vee Q$, a tautology, and therefore a minimal model preserving modification. The first (left) process will generate the minimal models of the theory in which the query is not satisfied. The second process returns the minimal models satisfying the query. The structure of the resulting tree is displayed in Figure 2. If DB is consistent ($\mathcal{MM}(DB) \neq \emptyset$), we can have one of the following possible cases:

1. $\mathcal{MM}(DB)_{Neg(Q)} = \mathcal{MM}(DB)$ and $\mathcal{MM}(DB)_{\{Q\}} = \emptyset$. That is, the first process returns all the minimal models of DB and the second returns no minimal models. The query is *false* in all minimal models of DB and its negation can be assumed to be *true* under the Closed World Assumption.
2. $\mathcal{MM}(DB)_{\{Q\}} = \mathcal{MM}(DB)$ and $\mathcal{MM}(DB)_{Neg(Q)} = \emptyset$. That is, the second process returns all the minimal models of DB and the first returns none. The query is *true* in all minimal models of DB (a logical consequence of DB) and Q is a (not necessarily minimal) SURE answer.

3. $\mathcal{MM}(DB)_{Neg(Q)} \neq \emptyset$ and $\mathcal{MM}(DB)_{\{Q\}} \neq \emptyset$. That is, each of the two processes returns some minimal models of DB. The query is *true* in some minimal models ($\mathcal{MM}(DB)_{\{Q\}}$) and *false* in others ($\mathcal{MM}(DB)_{Neg(Q)}$). Q is a $MAYBE$ answer.

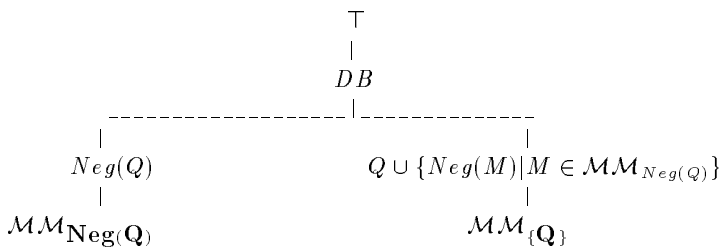

Fig. 2. The Minimal Model Tree Structure for Positive Queries.

One may elect to have the procedure stop when the first process generates no models on the assumption that the query is a logical consequence of the theory. However, running the second process will have the added advantage of showing that there are models for the theory and therefore it is consistent. Additionally we may want to use the second pass for more refined query answering [17].

When Q is negative, $Neg(Q)$ is positive. MM-Satchmo will operate on Q and $Neg(Q)$, in that order to maintain the unidirectional model subsumption property. That is, we still process the negative component first. The results obtained for positive queries can be applied here with the obvious modifications.

Example 4. $DB = \{\top \rightarrow a \vee b, a \rightarrow c, b \rightarrow c, d \rightarrow e\}$. $Q_1 = c$, $Q_2 = b$ and $Q_3 = \neg d$. $\mathcal{MM}(DB) = \{\{a, c\}\{b, c\}\}$.

- $DB \cup \{\neg c\} \vdash \Box$. ($DB \cup \{\neg c\}$ has no models). The tree for Q_1 is given in Figure 3-a.
- $DB \cup \{\neg b\} \nvdash \Box$. $DB \cup \{\neg b\}$ has the only minimal model $\{a, c\}$, a minimal model for DB. Q_2 is a MAYBE answer. The tree for Q_2 is given in Figure 3-2.
- $DB \cup \{d\} \nvdash \Box$. $DB \cup \{d\}$ has the minimal models $\{a, c, d, e\}$ and $\{b, c, d, e\}$. None of these models is minimal for DB. $\mathcal{MM}(DB \cup \{\neg d\}) = \{\{a, c\}\{b, c\}\} = \mathcal{MM}(DB)$. $\neg d \in GCWA(DB)$. The tree for Q_3 is given in Figure 3-c.

Example 7 offers some more complex cases.

3.2 Mixed Queries

A mixed query can be represented as a clause in implication form with the conjunction of negatively occurring atoms as the body and the disjunction of positively occurring atoms as the head.

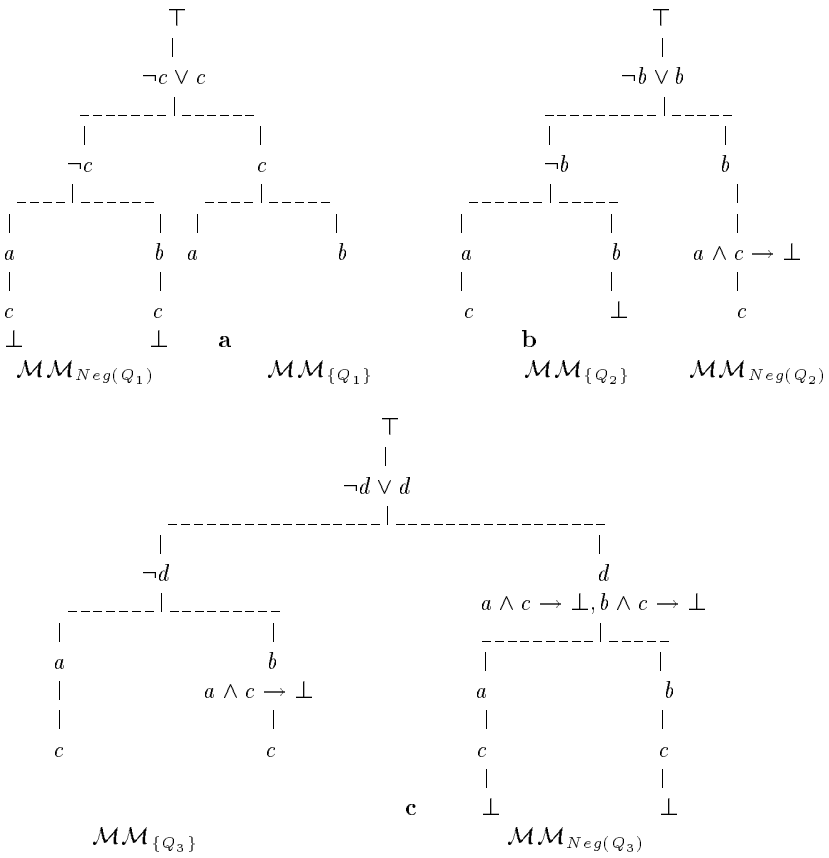

Fig. 3. Minimal Model Tree Structure for Queries of Example 4.

Let $Q = Body(Q) \rightarrow Head(Q)$ or $Q = \neg Body(Q) \vee Head(Q)$. Q is *true* in DB if all minimal models of DB satisfy Q and *false* otherwise. That is, Q is *false* if and only if there exists a minimal model of DB in which Q is *false*: $\exists M \in \mathcal{MM}(DB) | M \models Body(Q)\ and\ M \not\models Head(Q)$.

To answer such a query, we use the order it induces on the minimal model set to find the elements in which the query is falsified, if any. To retain the unidirectionality of model subsumption, we work with most constrained theories first (Theorem 12 item 4). We start by searching for minimal models in which $Head(Q)$ is *false* by adding $Head(Q) \rightarrow \bot$ to the theory to be expanded in the current branch. We denote this set by $\mathcal{MM}_1$. The set of remaining minimal models of DB, those in which the head of Q is *true*, is denoted by $\mathcal{MM}_2$. Clearly, $\mathcal{MM}(DB) = \mathcal{MM}_1 \cup \mathcal{MM}_2$. Further, we split $\mathcal{MM}_1$ into two sub-branches: first we find the set of minimal models in which $Body(Q)$ is *false* by adding $Body(Q) \rightarrow \bot$ and denote this set by $\mathcal{MM}_{1,1}$. Then we find the minimal models in which $Body(Q)$ is *true* by adding $Body(Q)$ and the negation

of all elements of $\mathcal{MM}_{1,1}$, $\{Neg(M)|M \in \mathcal{MM}_{1,1}\}$. We call this set $\mathcal{MM}_{1,2}$. Figure 4 displays the model structure for the resulting tree.

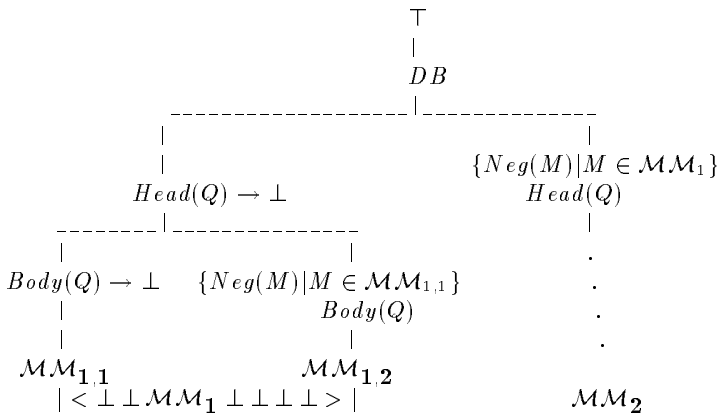

Fig. 4. The Model Tree Structure for Nonpositive Queries.

Theorem 14. *Under the above partitioning of the set of minimal models of DB induced by components of Q(Figure 4): Q is true in DB iff* $\mathcal{MM}_{1,2} = \emptyset$.

Proof. The correctness of the model computation process is the result of computing most constrained models first as required by Theorem 12.

Q is satisfied by elements of $\mathcal{MM}_2$ by having $Head(Q)$ satisfied. Q is satisfied by elements of $\mathcal{MM}_{1,1}$ by having $Body(Q)$ falsified. Q can be falsified only by an element $M \in \mathcal{MM}_{1,2}$ satisfying $Body(Q)$ while $Head(Q)$ is falsified in M. The result follows immediately.

Example 5. Let $DB = \{\top \rightarrow a \vee c, \top \rightarrow b \vee c \vee e, \top \rightarrow c \vee d \vee e, c \rightarrow d \vee e\}$, $Q_1 = a \wedge b \rightarrow c \vee d$ and $Q_2 = a \wedge d \rightarrow c \vee e$. For Q_1: $\mathcal{MM}_{1,2} = \emptyset$ and therefore Q_1 is *true* in DB. The tree is given in Figure 5-a.

For Q_2: $\mathcal{MM}_{1,2} = \{\{a, b, d\}\}$ and therefore Q_2 is *false* in DB. The corresponding tree is given in Figure 5-b. It is easy to verify the answers by noting that $\mathcal{MM}(DB) = \{\{a, b, d\}, \{a, e\}, \{c, d\}, \{c, e\}\}$.

A mixed query can be interpreted as an integrity constraint. Answering it is checking for the satisfiability in the current state of the database. Satisfiability of a constraint under the $SURE$ semantics is interpreted as having it *true* in all minimal models of the theory (theoremhood approach) [8]. This can be weakened to give an affirmative answer under the $MAYBE$ semantics when Q is satisfied in at least one minimal model of DB. This happens when $\mathcal{MM}(DB) \backslash \mathcal{MM}_{1,2} =$

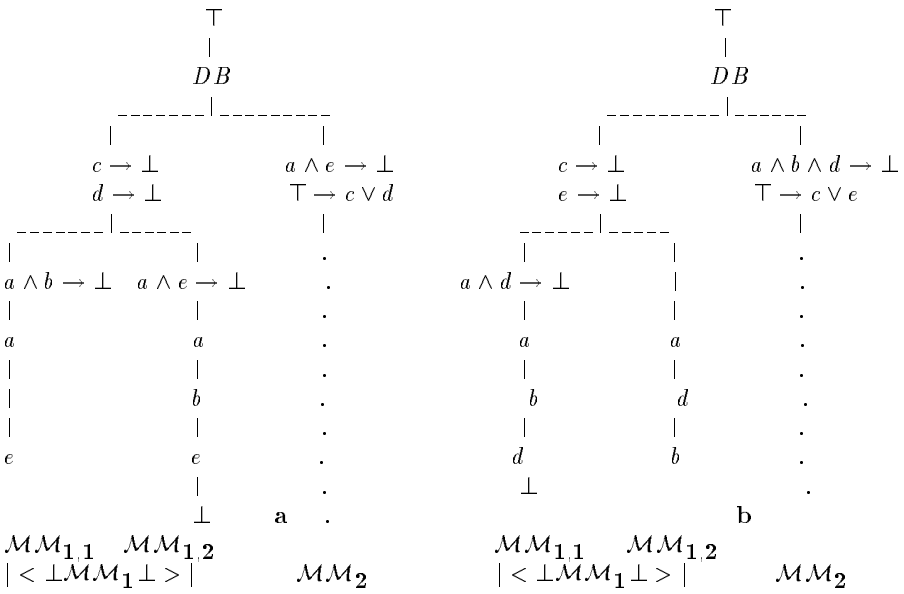

Fig. 5. The Model Trees Structure for Queries of Example 5.

$\mathcal{MM}_{1,1} \cup \mathcal{MM}_2$ is nonempty[1]. Answering Q in this case is integrity checking where the satisfiability of a constraint is interpreted as having it *true* in at least one minimal model of the theory (consistency approach) [8].

Mixed queries can be viewed as a generalization of other cases as reflected in Table 1.

#	Item of Fig. 4	For a Positive Query	For a Negative Query
0	Query Form	$\top \rightarrow Q$ (empty body)	$Q \rightarrow \bot$(empty head)
1	$Head(Q) \rightarrow \bot$	$Neg(Q)$	$\bot \rightarrow \bot$ adds nothing
2	$Body(Q) \rightarrow \bot$	$\top \rightarrow \bot$: a contradiction	$\{Q\}$
3	$Body(Q)$	$\top$, adding it has no effect	$Neg(Q)$
4	$\mathcal{MM}_{1,1}$	$\emptyset$ (In view of item 2)	$\mathcal{MM}_{\{Q\}}$ (In view of item 2)
5	$\mathcal{MM}_{1,2}$	$\mathcal{MM}_{Neg(Q)}$	$\mathcal{MM}_{Neg(Q)}$ (In view of item 3)
6	$\mathcal{MM}_2$	$\mathcal{MM}_{\{Q\}}$	$\emptyset$, $(Head(Q) = \bot)$

Table 1. Positive/Negative Queries as Special Cases of Mixed Queries.

[1] The set $\mathcal{MM}(DB) \setminus \mathcal{MM}_{1,2}$ is the set of minimal models in which the constraint corresponding to Q is satisfied. This may be interpreted as the set of the *legitimate* minimal models of DB given the constraint Q and its consistency interpretation. The detailed treatment of this issue is beyond the scope of this paper.

The common feature of the seemingly different classes of queries: the class of pure (positive and negative) queries and the class of mixed queries under the minimal model semantics is that the queries themselves are not allowed to "actively" participate in the model generation process. In this regard they exhibit the same behavior as integrity constraints. This is in line with the *epistemic* or *meta-level* view of integrity constraints under which the constraints are understood as statements specifying what is true about the DDDB rather than about the world modeled by the DDDB [7, 15, 9]. Answering the types of queries discussed here can therefore be viewed as checking if the corresponding epistemic constraint holds in the given theory. No positive atom is added to the model tree with the sole purpose of satisfying a generalized query[2]. In this regard they look more like integrity constraints and differ from positive facts and derivation rules which are used to add atoms to the model tree. The generalized query answering process consists of checking that the query holds in every minimal model of the theory. In a sense, the query is treated as an element *external* to the theory: it may participate in ordering the tree branches or even closing them but not in their expansion. The approach presented here can be viewed as a way to achieve this behavior.

Another point to stress is that while we used the collection of constraints corresponding to generated minimal models to ensure minimal model soundness, other approaches for minimality checking can be utilized [13, 18].

4 Monotonicity Properties of Query Answering

The classes of queries discussed in this paper span many of the applications encountered in database maintenance and exploitation. For each class we considered both MAYBE and SURE answers. Of interest is the monotonicity of the query answering process for each of the query classes considered. This refers to the validity of an already generated answer to a query after the database undergoes a clause addition update.

In this section we show that different classes of queries/answers exhibit different monotonicity properties and use the results to prove that certain inferences used in the query answering process can be nonmonotonic for DDDBs even for positive queries.

Definition 15. (monotonicity) Let DB and DB^+ be two consecutive states[3] of a DDDB such that DB^+ is the result of adding some clauses to DB: $DB \subseteq DB^+$. Property π is monotonic if whenever π holds in DB then π also holds in DB^+.

[2] The order of model generation and the additional constraints corresponding to each minimal model produced ensure that query items added during the answering process have no effect on the minimal model structure.

[3] We assume that DB and DB^+ are consistent.

The following lemma is an extension of a result in [6] that relates the models of successive states of a disjunctive deductive database, before and after a clause addition update.

Lemma 16. *Let DB and DB^+ be two consecutive states of a DDDB such that DB^+ is the result of adding some clauses to DB: $DB \subseteq DB^+$. Then:*

- *For all $M^+ \models DB^+$ there exists $M \models DB$ such that $M \subseteq M^+$. In particular: for all $M^+ \in \mathcal{MM}(DB^+)$ there exists $M \in \mathcal{MM}(DB)$ such that $M \subseteq M^+$.*
- *There may exist models $M \in \mathcal{MM}(DB)$ but no $M^+ \in \mathcal{MM}(DB^+)$ such that $M \subseteq M^+$.*

Proof. Immediate in view of Lemma 11, Theorem 12 and Example 6.

Example 6. $DB = \{a \vee b, c\}$. $DB^+ = DB \cup \{a \rightarrow b\}$. $\mathcal{MM}(DB) = \{\{a, c\}, \{b, c\}\}$. $\mathcal{MM}(DB^+) = \{\{b, c\}\}$.

Note that for a definite database the only relevant cardinality is that of its only minimal model. Adding a (positive) definite fact will result in extending the minimal model by adding that and maybe some other atoms that were not previously derivable. The minimal model remains unchanged otherwise.

Theorem 17. *Let DB and DB^+ be two states of a DDDB such that DB^+ is the result of adding clauses to DB: $DB \subseteq DB^+$ and Q be a generalized query such that $Q = Body(Q) \rightarrow Head(Q)$.*

- *Assume that Q is true in all minimal models of DB (a SURE answer). If this is because:*
  1. *$Head(Q)$ is true in all minimal models of DB then Q is true in all minimal models of DB^+ (Monotonic).*
  2. *Or else $Body(Q)$ is false in some minimal models of DB then Q need not be true in all minimal models of DB^+ (Nonmonotonic).*
- *If Q is true in some, but not all, minimal models of DB (a MAYBE answer) then Q need not be true in any minimal models of DB^+ (Nonmonotonic).*

Proof. – Let $C \in DB^+ \setminus DB$. If C is negative (denial rule) then by Theorem 11, $\mathcal{MM}(DB^+) \subseteq \mathcal{MM}(DB)$ and the result is clear. Otherwise, by Lemma 16, for any $M^+ \in \mathcal{MM}(DB^+)$ there is an $M \in \mathcal{MM}(DB)$ such that $M \subseteq M^+$.

1. If $Head(Q)$ is *true* in all elements of $\mathcal{MM}(DB)$ then Q necessarily holds for any M^+ since M^+ is a (not necessarily proper) superset of an element in $\mathcal{MM}(DB)$.
2. If $Body(Q)$ is *false* in some elements of $\mathcal{MM}(DB)$ then $Body(Q)$ may become *true* in the expansions of such models and thus make the Q *false* if its head was not earlier satisfied.

– If Q is *true* only in some elements of $\mathcal{MM} \subset \mathcal{MM}(DB)$, then it may hold for no element of $\mathcal{MM}(DB^+)$ if every one of the expansions of the elements of $\mathcal{MM}$, call this set $\mathcal{MM}^+$, is subsumed by elements in the set $(\mathcal{MM}(DB^+) \setminus \mathcal{MM}^+)$. That is, if for all $M^+ \in \mathcal{MM}^+$ $\exists M \in (\mathcal{MM}(DB^+) \setminus \mathcal{MM})$ such that $M \subset M^+$. Therefore, Q may be *false* in DB^+.

Corollary 18. *Given a DDDB, DB, DB^+ the updated version of DB by clause addition and a query Q. Then:*

1. *If Q is positive then:*
   - *The SURE answer property is monotonic. If Q is a SURE answer in DB then it is also a SURE answer in DB^+.*
   - *The MAYBE answer property is nonmonotonic. Q can be a MAYBE answer in DB but not a MAYBE answer in DB^+.*
2. *If Q is nonpositive (negative or mixed) then both SURE and MAYBE answers are nonmonotonic.*

Proof. Immediate.

Example 7. Let $DB = \{P(a) \vee P(b), Q(a), Q(b), P(c) \vee P(d),\ P(d) \rightarrow P(c) \vee P(a)\}$. $Q_1 = P(a)$, $Q_2 = P(a) \wedge Q(a)$, $Q_3 = (P(a) \wedge Q(a)) \vee (P(b) \wedge Q(b))$, $Q_4 = P(c) \vee P(d)$, $Q_5 = Q(a) \rightarrow P(a)$, $Q_6 = P(e) \rightarrow P(a)$, $Q_7 = P(b) \wedge P(e) \rightarrow \perp$, $Q_8 = P(b) \wedge Q(b) \rightarrow \perp$. $\mathcal{MM}(DB) = \{\{P(a), Q(a), Q(b), P(c)\}, \{P(b), Q(a), Q(b), P(c)\}, \{P(a), Q(a), Q(b), P(d)\}\}$. Consider $DB^+ = DB \cup \{P(b), P(c), P(e)\}$. $\mathcal{MM}(DB^+) = \{\{P(b), Q(a), Q(b), P(c), P(e)\}\}$.

Q_1 is a MAYBE answer in DB but not in DB^+. Q_2 is a MAYBE answer in DB but not in DB^+. Q_3 and Q_4 are SURE answers in DB and DB^+. Q_5 is a MAYBE answer in DB but not in DB^+. Q_6 is a SURE answer in DB but not in DB^+. Q_7 is a SURE answer in DB but not in DB^+. Q_8 is a MAYBE answer in DB but not in DB^+.

The monotonicity of the SURE answers for positive queries was established in [2] in the context of defining the sub-implication which is also based on minimal model properties. Our results show that, in general, the monotonicity of answers depends not only on the query itself but also on the minimal model structure of the theory and how it relates to the query under consideration. The nonmonotonicity of closed world reasoning is in line with Theorem 17.

We considered only addition updates but didn't limit ourselves to adding positive clauses. The addition of nonpositive clauses is allowed as well. Positive and mixed clause addition may change the status of individual minimal models in the transition (from DB to DB^+), when some of the minimal models of DB attempt to expand. Negative clauses, however, cannot cause model expansion. They can at most make minimal models of DB nonmodels of DB^+, as suggested by Theorem 12, including making DB^+ inconsistent.

It is possible to use similar reasoning to obtain monotonicity results, parallel to those discussed here, for the case of *no* answers to queries. One may also consider the case when updates are performed through clause deletions. However, we don't elaborate on these issues here.

An important point is that the information returned by the query answering procedure can be utilized to decide the monotonicity properties of individual queries. As suggested by Theorem 17 and Corollary 18 and the tree in Figure 4, a generalized query Q is monotonic if and only if $\mathcal{MM}_2 = \mathcal{MM}(DB)$ for *yes* answers. As a result the outlined procedure makes it possible to tag an

answer as monotonic/ nonmonotonic at no extra cost. Once a query is tagged as monotonic, future database updates will not affect its status and it need not be rechecked. This can be employed to enable an incremental construction of the minimal model tree for a theory. After an update, only nonmonotonic rules (treated as queries) need to be rechecked. If not satisfied then further additions may be initiated to guarantee their satisfaction. Actually, one may reduce the checking granularity by relating the monotonicity of individual clauses to individual models. However, the gain achieved by incremental checking needs to be weighted against the overhead cost of maintaining the necessary tables.

5 Conclusion and Remarks

We presented an approach to generalized query answering under the minimal model semantics for the class of range-restricted disjunctive deductive databases. It is based on the use of a sound and complete minimal model generating procedure . The concept of a query was extended to cover many classes of practical importance for database maintenance and exploitation. The efficiency of the approach depends on the efficiency of the used minimal model generating procedure. Experiments with a prototype of our procedure pointed to its efficiency as compared with similar ones reported in the literature [13]. It was able to handle theories with large numbers of models [3]. Of course, since the procedure retains already generated minimal models for subsequent model generation, one should expect the performance to degrade when the number of minimal models is very large: space requirements to store the corresponding constraints and the time needed to process them will increase. However, this is a major improvement on approaches that produce a complete set of models then compare them to test model minimality. Additionally, any efficiency enhancement tuning of the model generating procedure will reflect on the query answering process outlined in this paper without affecting the reported theoretical results [13, 18, 19]. Of course, the size of individual models can be large and the number of models will generally depend on the degree of indefiniteness of the theory. Adopting the model tree structure, separating the definite and indefinite components of the theory and other optimization techniques will enable sharing of atoms between models [5, 19]. The fact that our approach is limited to range-restricted DDDBs is an important limitation despite the algorithm given in [3] to convert other theories to this format. Therefore, our approach will benefit from approaches to minimal model generations that can handle DDDBs that are not range-restricted [1]. One of the main advantages of our approach is that it returns information that can be used to fine-tune the query answering process so as to decide the answer monotonicity or to specify the updates needed to have particular answers. If the user is interested in a simple *yes/no* answer then the minimal model generating procedure can be guided by the query to construct the most relevant models to the query answering process.

We also made distinction between $SURE$ and $MAYBE$ answers to a query. Both concepts were defined in terms of minimal models. We presented some

results regarding the monotonicity properties of different types of answers to different classes of queries. *SURE* answers to positive queries were shown to be monotonic relative to updating the database by clause addition. MAYBE answers on the other hand were shown to have a nonmonotonic nature and therefore needed re-computation after database updates. While other types of queries exhibited nonmonotonic behavior for all types of answers considered, we defined the conditions under which the answers are monotonic. Determining if these conditions hold can be viewed as a byproduct of the query answering process. This was shown to be useful for incremental construction of the minimal model structure of the theory.

Among the topics for further research are the use of a similar approach to answering queries under other database semantics such as stable and perfect model semantics [18] and treating answer monotonicity under updates other than clause addition. Another topic is using the monotonicity results of this paper to develop incremental methods for query processing in DDDBs [4] and the development of an integrated system based on a minimal model generator for the different aspects of database processing such as integrity enforcement and updates.

Acknowledgement:

Part of this research was done while the author was visiting at Munich University. The author thanks Prof. F.Bry and his group in Munich, the Alexander von Humboldt Stiftung for the support and the anonymous referees for their valuable comments.

References

1. P. Baumgatrner, U. Furbach, and I. Niemmelä. Hyper tabluaux. Technical Report 8-96, Institut für Informatik, Univesität Koblenz, Koblenz, Germany, feb 1996.
2. G. Bossu and P. Siegel. Saturation, nonmonotonic reasoning and the closed-world assumption. *Artificial Intelligence*, 25(1):13–63, January 1985.
3. F. Bry and A. Yahya. Minimal model generation with positive unit hyper-resolution tableaux. 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*, pages 143–159, Palermo, Italy, May 1996. Springer-Verlag. Vol. 1071, Full version: http://www.pms.informatik.uni-muenchen.de/publikationen/.
4. G. Dong, S. Jianwen, and R. Topor. Nonrecursive incremental evaluation of datalog queries. *Annals of Mathematics and Artificial Intelligence*, 14(1):187–223, 1995.
5. J. A. Fernández and J. Minker. Computing perfect models of stratified disjunctive databases. *Annals of Mathematics and Artificial Intelligence*, 1993. Submitted. Preliminary version presented at the ILPS'91 Workshop on Disjunctive Logic Programs, San Diego, California.
6. J. A. Fernández and J. Minker. Bottom-up computation of perfect models for disjunctive stratified theories. *Journal of Logic Programming*, 25(1):33–50, 1995.

7. A.C. Kakas, R.A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.
8. R. Kowalski and F. Sadri. A theorem proving approach to database integrity. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, 1988.
9. R.A. Kowalski. Problems and promises of computational logic. In *Lecture Notes in Computer Science Series*, pages 80–95. Springer-Verlag, 1990.
10. J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.
11. R. Manthey and F. Bry. Satchmo: a theorem prover implemented in prolog. In J.L. Lassez, editor, *Proc. 9^{th} CADE*, pages 456–459, 1988.
12. J. Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science 138*, pages 292–308. Springer-Verlag, 1982.
13. I. 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*, pages 278–294, Palermo, Italy, May 1996. Springer-Verlag. Vol. 1071.
14. R. Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.
15. R. Reiter. On asking what a database knows. In J.Lloyd, editor, *Proc. Symposium on Computational Logic*, 1990. Lecture Notes in Computer Science.
16. M. Suchenek. First-order syntactic characterizations of minimal entailment, domain minimal entailment and herbrand entailment. *Journal of Automated Reasoning*, 10:237–236, 1993.
17. A. Yahya. Generalized query answering in disjunctive databases using minimal model generation. Technical Report PMS-FB-96-13, LMU-München, Munich University, Munich, Germany, aug 1996. WWW: http://www.informatik.uni-muenchen.de/pms/publikationen/berichte/PMS-FB-1996-13.ps.gz.
18. A. Yahya. Model generation in disjunctive normal databases. Technical Report PMS-FB-96-10, LMU-München, Munich University, Munich, Germany, jun 1996. WWW: http://www.informatik.uni-muenchen.de/pms/publikationen/berichte/PMS-FB-1996-10.ps.gz.
19. A. Yahya, J.A. Fernandez, and J. Minker. Ordered model trees: A normal form for disjunctive deductive databases. *J. Automated Reasoning*, 13(1):117–144, 1994.
20. A. Yahya and L.J. Henschen. Deduction in Non-Horn Databases. *J. Automated Reasoning*, 1(2):141–160, 1985.

This article was processed using the LaTeX macro package with LLNCS style