INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig-
Maximilians-
Universität
München

Generalized Query Answering in Disjunctive Databases Using Minimal Model Generation

Adnan H. Yahya

http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1996-13, August 1996

Generalized Query Answering in Disjunctive Databases Using Minimal Model Generation

Adnan Yahya
Institut für Informatik, Ludwig-Maximilians-Universität München, Germany
Electrical Engineering Department, Birzeit University, Birzeit, Palestine
yahya@informatik.uni-muenchen.de

Abstract

Minimal models underly one of the major semantics for disjunctive theories and a substantial research effort was directed at minimal model reasoning. In this paper we investigate the process of generalized query answering under the minimal model semantics for the class of Disjunctive Deductive Databases. We cover several classes of queries that are of practical importance for database maintenance. Answers that are true in *all* and those that are true in *some* minimal models of the theory are considered and their monotonicity properties are discussed. Our approach is based on having the generalized query induce an order on the models returned by a sound and complete minimal model generating procedure. This makes it possible to introduce refinements to the query answering process such as allowing the specification of conditions under which a query becomes derivable from the database and checking for answer minimality.

1 Introduction

Minimal model semantics was one of the first to be defined for disjunctive theories[15, 12]. It is a natural extension of the semantics usually adopted for the definite case. Several of the model classes defined under other semantics, such as the *perfect* and *stable* models for disjunctive theories with body negation, are subsets of the minimal models of the theory and reduce to the minimal models in the absence of body negation.

Minimal models proved to be important for defining database completion: the mechanism to avoid the explicit storage of negative data. The Closed World Assumption which makes it possible to assume a negative ground atom of a Horn theory if it is not derivable was extended to the case of disjunctive theories [18, 15]. The extension was defined in terms of minimal models. Limiting our attention to the class of minimal models of the theory reconciles the concepts of derivability in all models and in all minimal models of the completed theory for positive and negative formulas [21, 25].

Model generating procedures have been used both for conventional theorem proving tasks [13] and to construct representations of a theory in terms of its minimal model structure and its set of minimally derivable ground clauses (minimal model state) [6, 24, 27]. One problem is that the static model representation of the theory is sensitive to minor updates: an update may require a radical modification of the model structure. Such an alternative representation is not really adequate since one may need to consult the original clausal theory to correctly reflect the effect of updates on the model structure. The close connection between the minimal model structure of a database and the set of minimally derivable ground clauses is well established [20, 27]. This connection makes it possible to switch between representations based on these concepts and to utilize them both to achieve efficient query processing.

Adopting minimal model semantics makes it natural to use minimal models in defining query answers. There can be more than one type of answer depending on the number of models in which the query is satisfied. It is of interest to determine the properties of various answer types including the monotonicity of their behavior. This is important for predicting the effects of database updates on repeated answering of a generalized query. Another issue that arises when answering queries against disjunctive databases is the minimality of answers since indefinite answers are possible. A good query answering procedure must try to return only minimal answers or to identify the minimal components of a nonminimal answer.

In this paper we present an approach to generalized query answering based on minimal model generation. The queries covered include those of importance for database maintenance and exploitation. The 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, to check for answer minimality and to refine the query answering process by specifying updates that will make the query derivable. We also address the issue of answer monotonicity for the different classes of queries and answers considered.

The rest of the paper is organized as follows. In the next section we give some relevant definitions and describe a class of model generating procedures that will later be used for query answering. We define the concept of a generalized query and two classes of query answers: those *true* in all minimal models and those that are *true* in some minimal models. In Section 3 we address the issue of using a minimal model generating procedure for generalized query answering. We also point to the possible use of an alternative representation based on the minimally derivable ground clauses to answer queries. In Section 4 we refine the process so that it is possible to return conditions under which the query becomes derivable/nonderivable and to designate a minimal component of a nonminimal answer. In Section 5 we discuss the monotonicity properties of the generalized query answering process for the classes of queries and answers considered. In section 6 we comment on the merits of our approach and compare it with others discussed in the literature and point to the possible extensions and further research.

2 Preliminaries and Background Material

In this section we review some of the basic concepts related to query answering in disjunctive deductive databases. We assume familiarity with the basic concepts as outlined in [12] and therefore limit ourselves to the basic material needed for the results presented in this paper.

Definition 2.1 *A* disjunctive deductive database (DDDB)*, DB, is a set of clauses of the form:*

$$C = A_1 \vee \cdots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n,$$

where $m, n \geq 0$ and the As and Bs are atoms in a First Order Language (FOL) $\mathcal{L}$ with no function symbols. C is positive if $n = 0$ and denial (negative) if $m = 0$.

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*[1] 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 2.2 *A clause C is range restricted if every variable occurring in the head of C also appears in a the body of C. A database is range restricted iff all its clauses are range restricted.*

In this paper we assume the theory to be range restricted.

2.1 Minimal Model Semantics

Under the minimal model semantics the meaning of the database is defined by its set of minimal models. A formula is a consequence of a theory if and only if that atom is *true* in every minimal model of the theory.

Definition 2.3 *Two DDDBs DB_1 and DB_2 are minimal-model equivalent if and only if they have exactly the same set of minimal models. $DB_1 =_{mm} DB_2$ iff $\mathcal{MM}(DB_1) = \mathcal{MM}(DB_2)$.*

Usually, negative information is not explicitly expressed in the database. Default rules are used to derive negative information. For definite databases the Closed World Assumption (CWA) is usually used [18]. Under CWA an atom A is assumed to be *false* iff A is not in the *unique* minimal model of the database. CWA is not applicable to DDDBs since it may produce inconsistent results. For DDDBs the rule used to define negated atoms is the *Generalized Closed World Assumption ($GCWA$)* which is an extension of the CWA rule to the disjunctive case [15]. $GCWA$ is able to consistently define those *atoms* whose negation can be assumed to be *true* in the database. To assume negative clauses the *Extended Generalized Closed World Assumption ($EGCWA$)* is used[25]. The default rules for the disjunctive case are formally defined as follows:

Definition 2.4 *[15, 7, 25] 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 $\{A_1 \wedge \cdots \wedge A_n\} \subseteq M\}$.*

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 from DB when the derivation process is augmented by the appropriate default rule for negation. We adopt the EGCWA because of the following result:

Lemma 1 *[25] 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 2.5 *Given an inclusion free finite set of finite interpretations $\mathcal{I}$ it is always possible to construct a positive ground theory $DB_{\mathcal{I}}$ such that $\mathcal{MM}(DB_{\mathcal{I}}) = \mathcal{I}$. The subsumption free version of $DB_{\mathcal{I}}$ is the minimal model state of $DB_{\mathcal{I}}$, $\mathcal{MS}(DB_{\mathcal{I}})$ [12].*

[1] As is common in the field, an interpretation is identified by the set of ground atoms assigned *true* in that interpretation. All other atoms of the Herbrand base are assigned *false*.

A procedure to construct the theory DB for $\mathcal{I}$ is to produce all clauses containing at least one atom from each element of $\mathcal{I}$. Removing subsumed clauses will produce the minimal model state.

In [27] it was shown that it is also possible to construct the minimal model state by the application of a complete minimal model generation procedure to the set of models $\mathcal{I}$ provided we treat the elements of $\mathcal{I}$ as clauses (the clause corresponding to an element of $I \in \mathcal{I}$ is the disjunction of the atoms of I).

Definition 2.6 *If $C = A_1 \vee ... \vee A_n$ is a disjunction of atoms, then by $Neg(C)$ we denote the set of clauses in implication form $Neg(C) := \{A_1 \rightarrow \perp, ..., A_n \rightarrow \perp\}$. If $M = \{A_1, ..., A_n\}$ is a finite interpretation then $Neg(M)$ denotes the clause in implication form $Neg(M) = A_1 \wedge ... \wedge A_n \rightarrow \perp$.*

Definition 2.7 *If DB_1 and DB_2 are sets of ground clauses then by $DB = DB_1 \vee DB_2$ we denote the set of all clauses we get by expanding a clause in DB_1 with a clause in DB_2 and removing duplicates. That is, $DB = \{A_1 \vee ... \vee A_n | A_{1,1} \vee ... \vee A_{1,k} \in DB_1,\ A_{2,1} \vee ... \vee A_{2,m} \in DB_2$, and $\{A_1, ...A_n\} = \{A_{1,1}, ..., A_{1,k}\} \cup \{A_{2,1}, ..., A_{2,m}\}$ and A's are ground atoms$\}$*

Lemma 2 *[26][2] Let DB be a theory with the set of minimal models $\mathcal{MM}(DB)$. Let $\mathcal{MM}_1$ and $\mathcal{MM}_2$ be a partition of $\mathcal{MM}(DB)$ such that $\mathcal{MM}_1 \cup \mathcal{MM}_2 = \mathcal{MM}(DB)$ and $\mathcal{MM}_1 \cap \mathcal{MM}_2 = \emptyset$. If DB_i denotes $\mathcal{MS}(\mathcal{MM}_i)$ then $\mathcal{MM}(DB) = \mathcal{MM}(DB_1 \vee DB_2)$.*

2.2 Model Generation

The main results of this paper are based on using model generating procedures with certain properties [3, 24]. We extensively utilize denial clauses (rules with empty heads) to impose an order on the generated models by restricting the search space in certain branches.

Definition 2.8 *Given a DDDB, DB, and a model generating procedure $\mathcal{P}$, by $\mathcal{P}(DB)$ we denote the result returned by $\mathcal{P}$ run with DB as input. We say that $\mathcal{P}$ is:*

1. *Sound: if it returns only models of DB: $\forall M \in \mathcal{P}(DB), M \models DB$.*
2. *Minimal-Model sound if it returns only minimal models of its input: $\mathcal{P}(DB) \subseteq \mathcal{MM}(DB)$.*
3. *Complete: if it returns all the minimal models of DB: $\mathcal{MM}(DB) \subseteq \mathcal{P}(DB)$.*
4. *Strict: if it returns no duplicates.*

Next we give a brief description of successively refined model generating procedures that are sound and complete [3]. Given a DDDB, DB, each of these procedures constructs a tree (model tree) with the ground unit clauses in each root-to-leaf branch representing a model of DB. The completeness implies that the tree has at least one branch representing each minimal model of DB.

Starting from $\top$ as the root, the procedure expands a tree for a range restricted DDDB, DB, by applying the following expansion rules:

Definition 2.9 (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.*

[2] A stronger version of this Definition holds where the subsets of the minimal models need not be disjoint.

Positive unit hyper-resolution (PUHR) rule:

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

Splitting rule:

$$\frac{E_1 \vee E_2}{E_1 \quad | \quad E_2}$$

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

Note that the splitting rule is always applied to *ground* disjunctions. This is possible since our theory is range restricted. The head is always ground when the body is ground (or empty).

Definition 2.10 (model tree) *A Model Tree for a DDDB, DB, is a tree 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 splitting rule) is possible to yield a formula E (respectively, 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 $\{E\}$ (respectively the two child nodes $\{E_1\}$ and $\{E_2\}$) as successor(s) to T.*

While the above definition imposes no order on the node expansions, we elect to maintain an order that will later be exploited for defining the properties of the generated tree.

Definition 2.11 *(***conventions for model generation***) When expanding a model tree we assume that the procedure adheres to the following rules:*

1. *Always select E_1 for splitting a disjunction $(E_1 \vee E_2)$ to be atomic.*
2. *Expand the leftmost atom of a disjunction first.*
3. *As a result of items 1 and 2 atoms of the clause are expanded from left to right (by adding the remainder of the clause, if any, to the top of the theory to be processed in the sibling branch).*

We always expand left branches of the model tree first. Our interest is only in branches with no occurrences of *false* (open branches). The branch expansion is stopped when *false* ($\perp$) is added (the branch closes). Only (ground) disjunctions that are not subsumed in the branch are expanded to avoid unnecessary expansions. The expansion continues until no new expansions are applicable (all branches are saturated). A branch represents the interpretation in which all (ground) unit clauses on that branch are assigned the truth value *true*. For the class of of range restricted DDDBs the procedure is *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. However, not all branches represent minimal models and some branches may be duplicates [3].

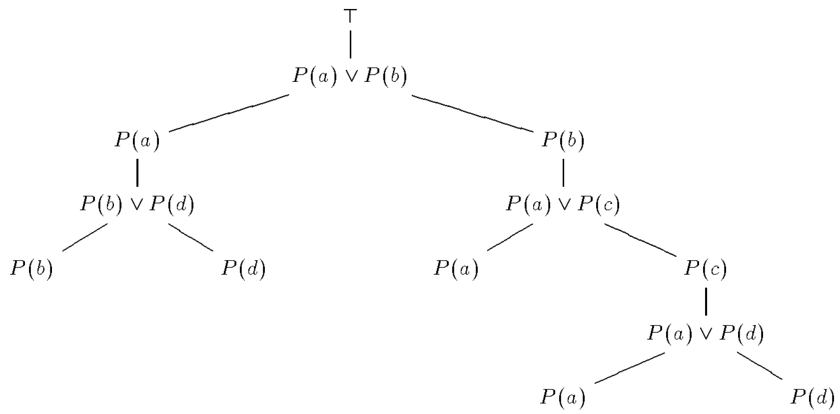

Figure 1: A Model Tree for Example 1 (with nonminimal and duplicate models).

Example 1 *Let DB be the following set of clauses:*

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

Figure 1 is a model tree for DB. The minimal model $\{P(a), P(b)\}$ of DB is generated twice. The tree also has a branch with the nonminimal model $\{P(a), P(b), P(c)\}$. Among others, all minimal models of DB, i.e. $\{P(a), P(b)\}$, $\{P(a), P(d)\}$, and $\{P(b), P(c), P(d)\}$ are generated.

Further, it was shown that replacing the splitting rule by the following one called *Complement Splitting Rule* preserves the completeness and soundness of the model generating procedure.

Definition 2.12 (complement splitting rule)

$$\begin{array}{lcr} \multicolumn{3}{c}{E_1 \vee E_2} \\ \hline E_1 & | & E_2 \\ [Neg(E_2)] & | & \end{array}$$

The adoption of this rule tends to reduce the search space by closing (adding *false* to) branches before they grow into complete nonminimal or duplicate models. Besides, the first (leftmost) model generated by a procedure using this rule is minimal.

Example 2 *Let DB be the set of clauses of Example 1:*

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

Figure 2 gives the model tree for DB. The models of this tree are $\{P(a), P(d)\}$, $\{P(b), P(c), P(a)\}$, $\{P(b), P(a)\}$, and $\{P(b), P(c), P(d)\}$. Note that although some are not minimal, no duplicates are returned and the first model is minimal.

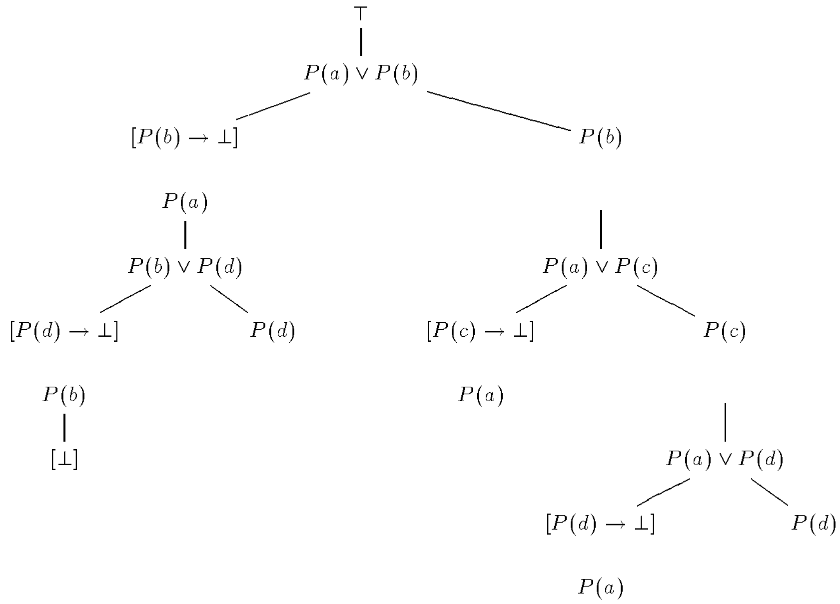

Figure 2: The Model Tree with Complement Splitting for Example 2.

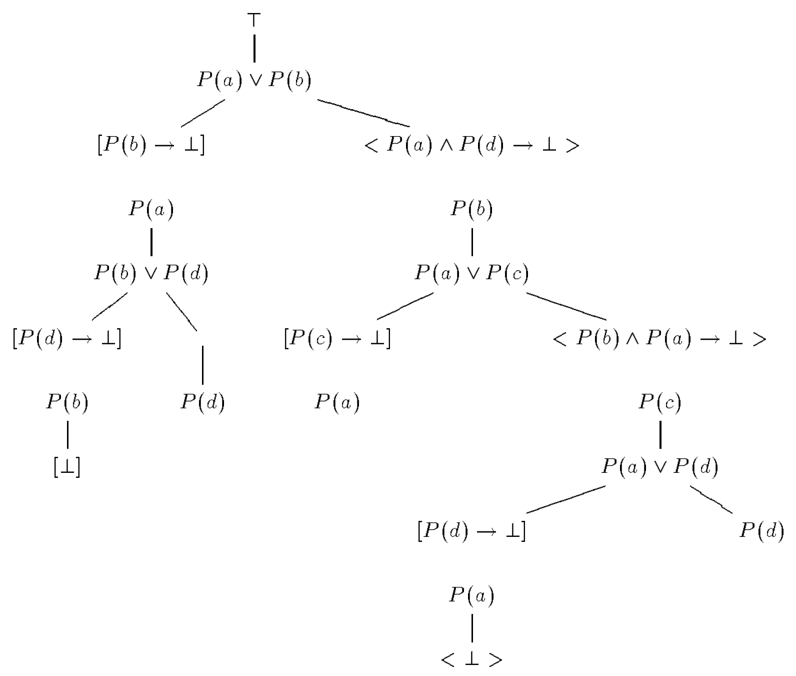

Figure 3: A Run of the Minimal Model Generation Procedure MM-Satchmo for Example 3.

If additionally, for each minimal model, M, generated so far we augment the theory by the negation of M, ($< Neg(M) >$), 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.

Example 3 *Figure 3 gives the search spaces of the minimal model generation procedure for the set of clauses of Examples 1 and 2:*

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

Note that all models returned by the procedure are minimal.

Given a DDDB, DB, with the top clause $C = A_1 \vee ... \vee A_n$ the procedure will generate the set of minimal models of DB rooted at $\top$ with the first (leftmost) branch containing the minimal models with A_1 and none of $A_2, ..., A_n$, the second branch containing the minimal models with A_2, possibly A_1 and none of $\{A_3, ..., A_n\}$, the i^{th} branch with minimal models with A_i, possibly some $A_j, j < i$ and none of $\{A_{i+1}, ..., A_n\}$. The last branch contains A_n possibly with other atoms of C.

[3] contains a Prolog implementation of a series of procedures, for the class of range restricted theories with finite minimal models, called: Satchmo for the program with splitting [13], CS-Satchmo for the implementation with complement splitting and MM-Satchmo for the implementation with model minimization (by including negation of generated minimal models).

2.3 Queries, Answers and Minimal Model Semantics

Given a DDDB, DB, the definition of the answer to an atomic query $P(x)$ against DB makes it necessary for a disjunction of atoms with predicate P to be derivable from DB. $a_1 + ... + a_n$ is an answer to $P(x)$ iff $DB \models P(a_1) \vee ... \vee P(a_n)$. If minimality is needed then it must additionally be the case that $DB \not\models P(a_1) \vee ... \vee P(a_m)$ for $m < n$. Minimality may be needed so as not to get too many subsumed answers.

If such a disjunction (pure in predicate P) is not derivable then no answers are assumed (or reported) although mixed clauses (containing atoms in P together with other predicates) may still be minimally derivable. That is $P(x)|\sigma$ for some substitution σ may be in some minimal models of DB when no answers to the query are reported. This asymmetry can be criticized as unfair to mixed clauses that can be used to describe disjunctive information in the database[2]. Clauses with a single predicate are a major source of disjunctive information in the database ($P(a) \vee P(b)$ - we know that the phone number is a or b but we are not sure which), but they need not be the only source. Consider the case when we know the phone number of a person (it is a) but are not sure whether it is a voice phone or a fax $P(a) \vee F(a)$. It may be worthwhile for a query answering procedure to state that a belongs to the person and that he/she can be reached at it (voice or fax). This statement may be more informative than having two possibilities for his phone: one is his own number and the other is not. For that we modify the definition of an answer to account for the admissibility of mixed clauses minimally derivable from DB to generate answers.

To accommodate answers that are *true* in only some minimal models of the theory, we elect to define two concepts of an answer to a query: SURE which is the traditional one and refers to substitutions that make the query *true* in all minimal models of DB and $MAYBE$ answers referring to substitutions that make the query *true* in some minimal models of DB. Clearly, for SURE answers it is worth distinguishing definite and indefinite answers while for the $MAYBE$ case this distinction is, in general, less useful[3] .

Definition 2.13 (syntactic definition of an answer) *Let $P(x)$ be an atomic query against DB.*

- *The set $\{a_1, ..., a_n\}$ (written also as, $a_1 + ... + a_n$) is a SURE (minimal SURE) answer to $P(x)$ iff DB derives $C = P(a_1) \vee ... \vee P(a_n)$. Or equivalently, if C is in the Model State of DB.*

  $\{a_1, ..., a_n\}$ is a minimal SURE iff $DB \vdash C$ and $DB \not\vdash C'$ such that C' subsumes C. Or equivalently, if C is in the Minimal Model State of DB.

- *A constant a is a MAYBE answer to $P(x)$ iff DB minimally derives $P(a) \vee C$ for some positive clause C not derivable from DB ($DB \not\vdash C$). Or equivalently, if $P(a)$ occurs in the Minimal Model State of the theory.*

Definition 2.14 (semantic definition of an answer) *Let $P(x)$ be an atomic query against DB.*

- *The set $\{a_1, ..., a_n\}$ is a SURE answer to $P(x)$ iff $P(a_1) \vee ... \vee P(a_n)$ is true in all minimal models of DB. $\{a_1, ..., a_n\}$ is minimal if, additionally, $\forall a_i\ \exists M \in \mathcal{MM}(DB)$ s.t. $a_i \in M$ and $\forall j \neq i, a_j \notin M$.*

[3] One may take the number of models in which a formula is *true* as a measure of its closeness to being a SURE answer. This is not pursued here.

- *a is a MAYBE answer to $P(x)$ iff $P(a)$ is in some minimal models of DB.*

It is straightforward to extend these definitions to the case of disjunctive and conjunctive queries: queries that are disjunctions of atomic queries and conjunctions of atomic queries, respectively. The syntactic and semantic definitions of SURE and $MAYBE$ answers are equivalent [25].

Example 4 *Consider $DB = \{P(a) \vee P(b), P(c), P(d) \vee R(a)\}$, and the Query $P(x)$. $\{a + b, c\}$ are (SURE) answers to $P(x)$ while d is not. a and b are MAYBE only answers and under our definition so is d. Note that d is not part of any SURE answer of the query.*

We will also be interested in *yes/no* answers to general ground queries.

Definition 2.15 (elementary generalized query) *An elementary generalized query is a ground clause: it is positive if the head is empty, negative if the body is empty and mixed otherwise.*

Definition 2.16 (positive/negative queries) *We say that 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.*

Clearly, 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 2.17 (query answer) *Let DB be a DDDB and let Q be a ground query*[4]*.*

- *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 in DB.

- *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. For positive queries the concepts of derivability and being *true* in all minimal models coincide [17, 21]. Since any superset of a minimal SURE answer is a SURE answer, we are particularly interested in minimal SURE answers.

It is generally desirable that a query answering procedure be able to return the *most strict* answer possible, unless instructed otherwise. A SURE answer cannot be reported as $MAYBE$ and nonminimal answers are not to be returned (at least have to be labeled as *nonminimal* with identification of the minimal answer component).

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

[4] If Q is positive clause then (along the lines of the syntactic characterization of an answer given in Definition 2.13) it is a $SURE$ answer iff it is derivable from DB or equivalently, iff it is in a the model state of DB. Q is a minimal $SURE$ answer if additionally no subset of Q is derivable from DB or equivalently, iff it is in the minimal model state of DB. Q is a $MAYBE$ answer iff one of its atoms occurs in the minimal model state of DB.

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: Recalling that a denial rule (negative clause) is satisfied in I iff an atom of its body in not in I the proof is straight forward [3]. ∎

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

Corollary 1 *Let DB be a DDDB, $\mathcal{M}$ be a set of models of DB such that $\mathcal{MM}(DB) \cap \mathcal{M} = \mathcal{MM}_1$ and $DB_{-\mathcal{M}} = DB \cup \{Neg(M) | M \in \mathcal{M}\}$. Then: $\mathcal{MM}(DB_{-\mathcal{M}}) = \mathcal{MM}(DB) \setminus \mathcal{MM}_1$.*

Proof: Given two distinct minimal models of DB, M_1 and M_2, clearly $M_1 \setminus M_2 \neq \emptyset$ and $M_2 \setminus M_1 \neq \emptyset$. Therefore, $M_1 \not\models Neg(M_1)$ and $M_1 \models Neg(M_2)$.

A model of $DB_{-\mathcal{M}}$ is also a model for $DB \subseteq DB_{-\mathcal{M}}$. If M is not minimal for DB then $M' \subset M$ is a minimal model for DB. As a subset of M, M' satisfies all elements of $DB_{-\mathcal{M}}$. A contradiction. ∎

The essence of Corollary 1 is that the addition of the negative clauses corresponding to non-minimal models of the theory has no effect on the minimal model structure of the theory. Only negative clauses corresponding to minimal models (or subsets of them) can affect the minimal model structure. Special cases of Corollary 1 are the following:

- If $\mathcal{M}$ is complete: $\mathcal{MM}(DB) \subseteq \mathcal{M}$ then $\mathcal{MM}(DB_{-\mathcal{M}}) = \emptyset$.
- If $\mathcal{M}$ contains no minimal models: $\mathcal{MM}(DB) \cap \mathcal{M} = \emptyset$ then $\mathcal{MM}(DB_{-\mathcal{M}}) = \mathcal{MM}(DB)$.

Corollary 2 *Let DB be a disjunctive theory and let $\mathcal{C}$ 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 alone.*

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 $\mathcal{MM}(DB \cup \mathcal{C})$ is inconsistent ($\mathcal{MM}(DB \cup \mathcal{C}) = \emptyset$).*

4. *If $\mathcal{C}_1, ..., \mathcal{C}_n$ are sets of denial rules such that $\mathcal{C}_n \subseteq ... \subseteq \mathcal{C}_1$. Then:*
   $\mathcal{MM}(DB \cup \mathcal{C}_1) \subseteq ... \subseteq \mathcal{MM}(DB \cup \mathcal{C}_n)$.

Corollaries 1 and 2 show that the addition of denial constraints can change the status of models to nonmodels but cannot affect the minimality of models.

3 Query Answering

Two representations of the entire answer set of the theory are possible: one based of the set of (minimal) ground clauses derivable from the theory and another based on the set of (minimal) models of the theory. Answering positive queries can be based on the syntactic or semantic definition/representation.

Using the syntactic definition of an answer one can construct the *Minimal Model State* or the *Clause Tree* for the database: a representation of DB in the form of a tree with branches corresponding to the minimally derivable clauses of the database. These are the clauses that need to be searched. It was shown that the minimal clause tree is the *dual* of the minimal model tree and either of them, if ordered, can serve as a Normal Form for a DDDB [20, 27]. Under such a representation of the theory, finding answers may be reduced to the process of a tree search for clauses with the atoms of interest. We give a brief description of this approach at the end of paragraph 3.3 for the completeness of the presentation.

The other approach is to use the semantic definition of the query answer and to utilize the minimal model representation to search for answers. It is this approach, which is applicable to generalized queries, that we elaborate on in this paper. We try to reduce the process of query answering to the invocation of a minimal model sound and complete model generating procedure (e.g MM-Satchmo [3]).

Using a minimal model generating procedure for query answering in disjunctive theories 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 [6, 24]). The minimal model generating procedure is used to construct such a tree and query answering is converted into searches in the tree. The representation is generally independent of the query and special arrangements such as indexing or tree restructuring are needed to facilitate the search for elements of the query in the tree. If the theory changes state then the minimal model generating procedure can be used to regenerate the minimal model structure. The drawback is that one may need to store two representations of the theory. The original (clausal) one and the minimal model representation since the two representations are only minimal model equivalent but are not equivalent in the more general sense. To demonstrate this point consider the following example:

Example 5 *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 have different results in the two representations. It generates $\{P(c), P(a)\}$ and $\{P(c), P(a), P(b)\}$ for the minimal model and clausal representations of the updated theory, respectively*[5].

If updates are frequent then reconstructing the minimal model tree may become costly.

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. We concentrate on this approach here and show that it can be useful for the incremental construction of a static minimal model representation.

[5]This could be looked upon as the satisfiability of clause $P(a) \rightarrow P(b)$ being *nonmonotonic* for clause addition updates of DB. In Section 5 we address this issue and discuss the conditions the satisfiability property is monotonic.

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 the negation is in the form of negative clauses or denial constraints (headless clauses in implication form). 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 DDDBs) [13, 3]. However, minimal model generation produces information that can be used to enrich the query answering process.

Theorem 2 *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 $Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB) \cup \{Q\})$.

($\leftarrow$)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 2 since $Neg(Q)$ consists entirely of denial rules. ∎

The proof of Theorem 2 shows that for a positive query model subsumption (if any) is unidirectional: minimal models of the theory augmented by negative clauses ($DB \cup Neg(Q)$) can subsume 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. This is demonstrated by the following example:

Example 6 *Consider $DB = \{P(a) \rightarrow P(b)\}$ and the query $Q = P(a)$. 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 $\{\}$.*
Note, however, that $\mathcal{MM}(DB) = Min(\mathcal{MM}(DB \cup Neg(Q)) \cup \mathcal{MM}(DB \cup \{Q\}) = \{\}$.

Another point proved by Theorem 2 is that the models abandoned during the model generation process for $DB \cup Neg(Q)$ have no influence on the minimality of the models in the other branch ($DB \cup \{Q\}$). Such models are abandoned for not satisfying $Neg(Q)$. They must satisfy Q and are bound to appear in the other branch (by completeness of the procedure) resulting in a correct minimization process.

When only SURE answers are needed one can run MM-Satchmo on $DB \cup Neg(Q)$. Substitutions for the variables in Q that generate a closed (empty) tree are SURE answers. No more cases need to be considered. When we are interested in $MAYBE$ answers as well as more refined query answering, more cases need to be considered. Theorem 2 suggests a simple procedure for answering positive queries posed to DB. Partition the set of minimal models of DB into two sets: one in which Q is *true* and the other in which Q is *false*. The way is to run two MM-Satchmo processes (Figure 4):

- The first process of MM-Satchmo will operate on the set union of the theory DB and the negation of the query under consideration: $DB \cup Neg(Q)$. We denote the (possibly empty) set of minimal models returned by this application of the procedure by $\mathcal{MM}(DB)_{Neg(Q)}$.
- The second will operate on the set union of the theory DB, Q and the constrains corresponding to the minimal models returned by the first process: $DB \cup \{Q\} \cup \{Neg(M) | M \in \mathcal{MM}(DB)_{Neg(Q)}\}$. We call the (possibly empty) set of minimal models returned by this application of the procedure $\mathcal{MM}(DB)_{\{Q\}}$.

The constraints in the second process are used to remove the models that satisfy Q but are not minimal for DB alone. If the second process is run without the constraints it is easy to note that nonminimal models can be returned since models of the second process can be subsumed by the models of the first. This was demonstrated by Example 6 above [3, 24]. So the two processes are **not independent**.

It is easy to see that, in the second branch, one can avoid adding the positive query to the theory since the minimal models that are produced by its presence and that are not models of the original theory will anyway be deleted by the integrity constraints corresponding to the models generated in the first branch. However, later we will exploit the presence of Q to impose a certain order on set of minimal models generated.

The entire process may be viewed as equivalent to augmenting DB with the clause $\neg Q \vee Q$, a tautology, and therefore a minimal model preserving modification to DB. From the implementation point of view the invocation may consist of prepending the clauses $\{\epsilon \vee Q, \epsilon\}$, where ϵ is an atom not occuring in DB that will be discarded in all resulting models. The left branch (first process) will have ϵ and $Neg(Q)$ (by complement splitting of $\epsilon \vee Q$) while the right branch (second process) will have only $\{Q, \epsilon\}$. The effect of atom ϵ in the left branch is offset by the clause ϵ in the right. The first (left) process will generate the minimal models of the theory in which the query is not satisfied. The second process will return the minimal models of the theory satisfying the query. This will give us better flexibility in answering such queries. The structure of the resulting tree is displayed in Figure 4. If DB is consistent ($\mathcal{MM}(DB) \neq \emptyset$), we can have 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. In this case the query is *false* in all minimal models of DB and its negation for the generated substitutions 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 no minimal models. In this case the query is *true* in all minimal models of DB (a logical consequence of DB). The substitutions in the query will serve as, (not necessarily minimal) SURE answers.
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. In this case the query is *true* in some minimal models ($\mathcal{MM}(DB)_{\{Q\}}$) and *false* in others ($\mathcal{MM}(DB)_{Neg(Q)}$). The resulting substitution is $MAYBE$ answer.

Running two processes makes it possible to compare the models in each branch to the entire set of minimal models. Without that one can only detect complete refutations. The absence of a refutation

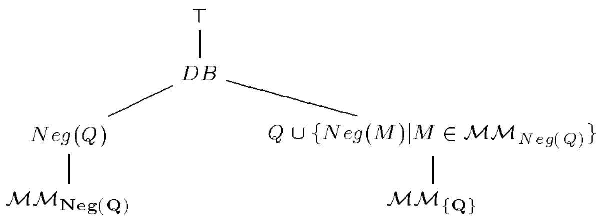

Figure 4: The Minimal Model Tree Structure for Positive Queries.

gives no indication about the number of models in which the query is satisfied. In our approach one always have the entire set of minimal models. 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 to designate the minimal components of Q that are derivable from DB as will be elaborated on in paragraph 4.1.

When Q is a negative query its negation, $Neg(Q)$, is positive. The two copies of MM-Satchmo will operate on Q and $Neg(Q)$, in that order so that 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. For negative queries, the need for the two branches is easy to see.

Consider the following example:

Example 7 $DB = \{\top \rightarrow a \vee b, a \rightarrow c, b \rightarrow c, d \rightarrow e\}$, $Q_1 = c$, $Q_2 = \neg d$ *and* $Q_3 = b$.

- $DB \cup \{\neg c\} \vdash \Box$. *(*$DB \cup \{\neg c\}$ *has no models).*
- $DB \cup \{d\} \not\vdash \Box$. *(*$DB \cup \{d\}$ *has the minimal models* $\{a, c, d, e\}$ *and* $\{b, c, d, e\}$*). None of these models is minimal for* $DB.DB \cup \{\neg d\}$ *has the set of minimal models* $\{\{a, c\}\{b, c\}\}$. $\neg d \in GCWA(DB)$.
- $DB \cup \{\neg b\} \not\vdash \Box$. *(*$DB \cup \{\neg b\}$ *has the only minimal model* $\{a, c\}$*, a minimal model for* DB*).*

The corresponding trees are given in Figures 5, 6 and 7. Example 13 offers more complex cases.

3.2 Mixed Queries

Mixed ground queries that contain both negative and positive atoms 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. So 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

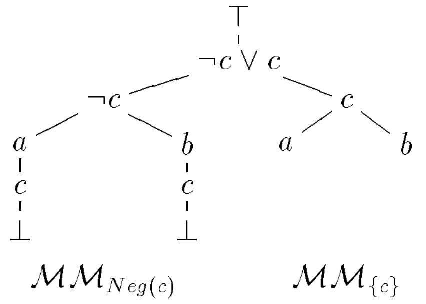

Figure 5: A Minimal Model Tree Structure for $Q_1 = c$ for Example 7.

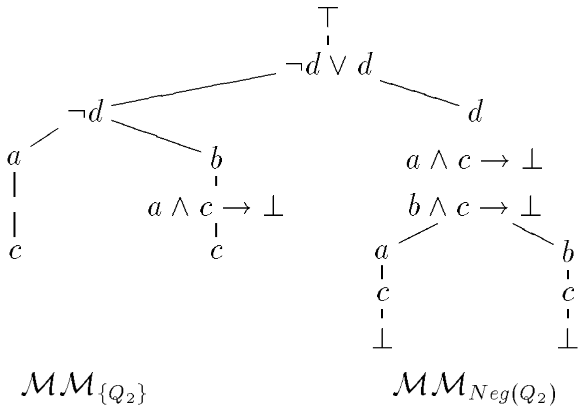

Figure 6: A Minimal Model Tree Structure for $Q_2 = \neg d$ for Example 7.

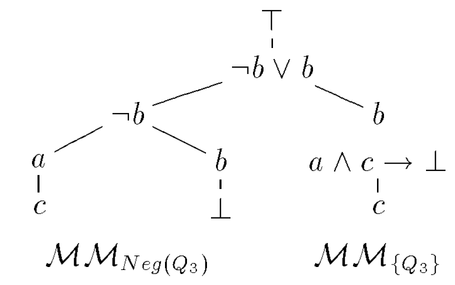

Figure 7: A Minimal Model Tree Structure for $Q_3 = b$ for Example 7.

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* $M \not\models Head(Q)$.

We use the order induced by the query 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 (Corollary 2). We start by searching for minimal models in which $Head(Q)$ is *false* by adding $Head(Q) \rightarrow \perp$ to the theory to be expanded in the current branch. Denote the set of these models 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*$(Q) \rightarrow \perp$ 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 8 displays the model structure for the resulting tree.

Theorem 3 *Under the above partitioning of the set of minimal models of DB induced by components of Q(Figure 8):*

- *Q is true in DB if and only if $\mathcal{MM}_{1,2} = \emptyset$.*

Proof: The correctness of the model computation process is the result of computing most restricted models first as required by Corollary 2.

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 8 *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 9.*

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 10.*

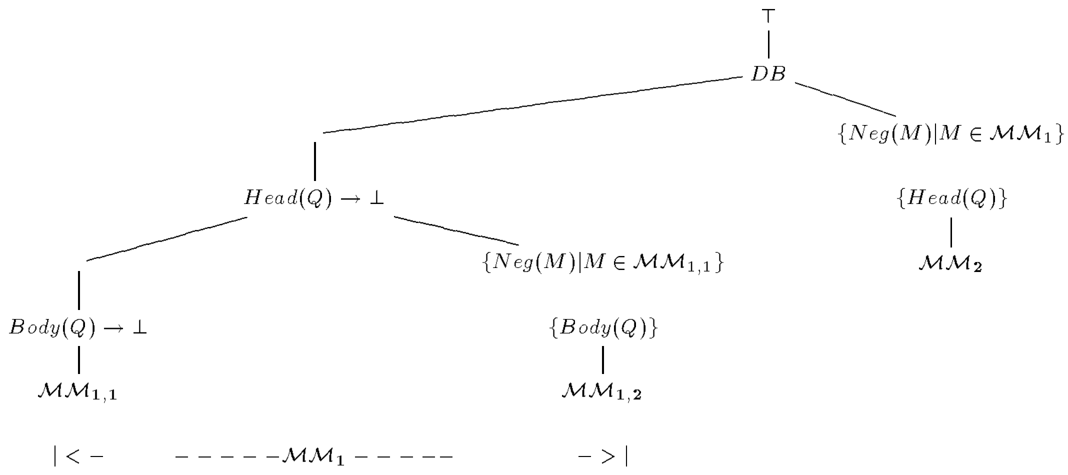

Figure 8: The Model Tree Structure for Nonpositive Queries.

It is easy to verify the answers by noting that $\mathcal{MM}(DB) = \{\{a, b, d\}, \{a, e\}, \{c, d\}, \{c, e\}\}$.

Note that a mixed query can be interpreted as an integrity constraint. Answering it is checking for the satisfiability of the integrity constraint 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) [9]. One can elect to weaken this concept so as to give an affirmative answer under the $MAYBE$ semantics when Q is satisfied in at least one minimal model of DB. Clearly this will be the case when $\mathcal{MM}(DB) \setminus \mathcal{MM}_{1,2} = \mathcal{MM}_{1,1} \cup \mathcal{MM}_2$ is nonempty[6]. 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) [9].

Answering mixed queries can be treated as a generalization of other cases. Other queries are special cases of the mixed query as follows:

- Positive queries have empty bodies: Q can be rewritten as $\top \to Q$ and:
    1. $Head(Q) \to \perp$ is $Neg(Q)$,
    2. $Body(Q) \to \perp$ is $\top \to \perp$: a contradiction.
    3. $Body(Q) = \top$ and adding it has no effect,
    4. $\mathcal{MM}_{1,1} = \emptyset$ (in view of item 2),
    5. $\mathcal{MM}_{1,2} = \mathcal{MM}_{Neg(Q)}$,

[6] The set $\mathcal{MM}(DB) \setminus \mathcal{MM}_{1,2}$ is the set of minimal models in which the constraint (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.

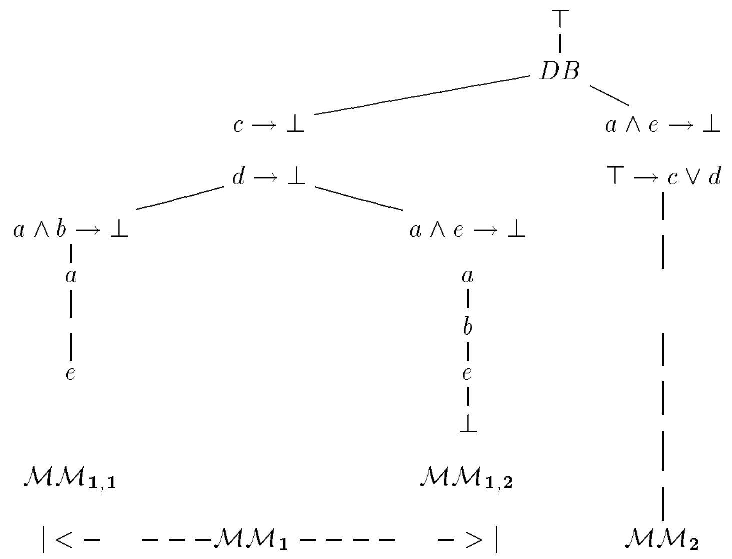

Figure 9: The Model Tree Structure for Q_1, Example 8.

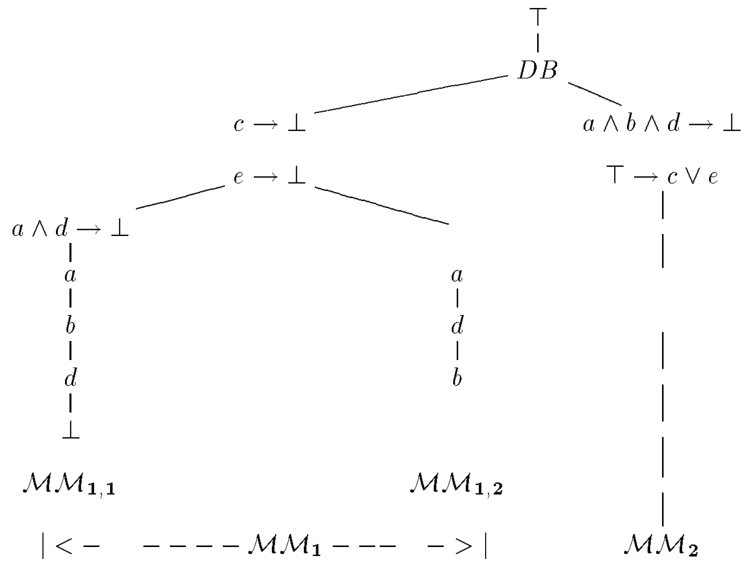

Figure 10: The Model Tree Structure for Q_2, Example 8.

#	Item	Positive Query	Negative Query
0	Query Q	empty body: $\top \rightarrow Q$	empty head: $Q \rightarrow \bot$
1	$Head(Q) \rightarrow \bot$	$Neg(Q)$	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: Correspondence Between Mixed and Positive/Negative Queries.

6. $\mathcal{MM}_2 = \mathcal{MM}_{\{Q\}}$.

- Negative queries have empty heads Q is $Q \rightarrow \bot$,

  1. $Head(Q) \rightarrow \bot$ adds nothing,
  2. $Body(Q) \rightarrow \bot$ is Q,
  3. $Body(Q)$ is $Neg(Q)$,
  4. $\mathcal{MM}_{1,1} = \mathcal{MM}_{\{Q\}}$ (in view of item 2),
  5. $\mathcal{MM}_{1,2} = \mathcal{MM}_{Neg(Q)}$ (in view of item 3),
  6. $\mathcal{MM}_2 = \emptyset$, $(Head(Q) = \bot)$.

These results are summarized in Table 1.

The common feature of the seemingly different classes: the class of 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. No positive atom is added to the model tree with the sole purpose of satisfying a generalized query. 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 [17, 29].

3.3 Query Answering with Minimal Model States

Given a DDDB, DB, one possible approach to use a model generating procedure for answering a positive query is to use the duality principle to construct the minimal model state for the theory. The minimal model state $\mathcal{MS}(DB)$ is the set of minimal ground disjunctions derivable from DB.

Given such a representation, the process of query answering can be reduced to a search process in the (tree) representation of the minimal model state.

In [27] it was shown that the minimal clause tree (the tree representation of the minimal model state) of a DDDB, DB, is the dual of the minimal model representation of DB. It can be constructed by applying a minimal model generating procedure to the set of minimal models of DB treated as a set of clauses (or alternatively, to the set of minimal models when both clauses and models are treated as sets of atoms).

So, the approach is to apply the minimal model generating procedure (say MM-Satchmo) a first time to DB producing the set of minimal models $\mathcal{MM}(DB)$, then a second time to $\mathcal{MM}(DB)$ to generate the set of minimal clauses of DB, $\mathcal{MS}(DB)$. The process of query answering is then performed over $\mathcal{MS}(DB)$ and is basically a search process for the relevant clauses in $\mathcal{MS}(DB)$ corresponding to the given query. Details on how to perform this operation are described in [27].

A point to note is that the first run of the procedure is applied to the database itself which in principle can be any range restricted theory with finite minimal models (see [3] for how to deal with non-range restricted theories). This is a more general class than the one presented in [27], and encompasses a large subset of real life applications in the database field. The second run of the minimal model generation procedure is applied to the set of minimal models produced at the first stage. On the negative side, the set $\mathcal{MM}(DB)$ can be much larger than the set of elements in DB and therefore the second application may be costly. On the positive side the elements of the set $\mathcal{MM}(DB)$ are all ground. The groundness property can be exploited to make the second pass of the model generating procedure more efficient [27, 17]. Consider the following example:

Example 9 *[20] Let* $DB = \{PATH(x,z), PATH(z,y) \rightarrow PATH(x,y),$ $ARC(x,y) \rightarrow PATH(x,y), ARC(a,b) \vee ARC(a,c), ARC(b,d), ARC(c,d)\}$.

Let $Q_1 = PATH(a,d)$, $Q_2 = PATH(a,x)$ *and* $Q_2 = PATH(b,d) \vee PATH(c,d)$.

The first run of MM-SATCHMO generates the set of minimal models:

$\mathcal{MM}(DB) = \{M_1 = \{PATH(a,d), PATH(a,b), PATH(b,d), PATH(c,d)\},$ $M_2 = \{PATH(a,d), PATH(a,c), PATH(b,d), PATH(c,d)\}\}$. *Clearly* $PATH(a,d)$ *is in both minimal models and is therefore true in DB. Searching one can see that* $b+c$ *is an answer to* Q_2. Q_3 *has a yes answer. Running MM-SATCHMO on the set of clauses* $\{C_1 = M_1^d = PATH(a,d) \vee PATH(a,b) \vee PATH(b,d) \vee PATH(c,d), C_2 = M_2^d = PATH(a,d) \vee PATH(a,c) \vee PATH(b,d) \vee PATH(c,d)\}$ *generates the set of minimal clauses:*

$\mathcal{MS}(DB) = \{PATH(a,d), PATH(a,c) \vee PATH(a,b), PATH(b,d), PATH(c,d)\}$.

Searching for the answers to both Q_1 *and* Q_2 *as well as verifying that* Q_3 *is a nonminimal answer is straightforward.*

Updates to the database will be reflected on the structure of its minimal state representation. Therefore, these updates need to be propagated to the set of clauses derivable from the database either through updating the clause tree to reflect the current state or by reconstructing the new clause tree by twice applying the model generation procedure to the new state of the database. The choice may depend on such factors as the frequency and nature of updates.

The fact that our definitions of $SURE$ and $MAYBE$ answers were defined both syntactically and semantically makes it possible to express them in terms of minimal models or minimally derivable clauses. Detecting a $MAYBE$ answer is a straightforward operation in view of Definition 2.13. It just needs to occur in the minimal clause representation of the database. Testing for minimality under the minimal model state representation of the database is easy. A clause Q is a minimal answer if and only if it is in the minimal model state. If Q is not minimal then all subclauses of Q that are members of the minimal model state are minimal answers. That was the case for Q_3 above.

4 Refined Query Answering

The fact that our approach to query answering is based on a query-induced ordering of the minimal models of the theory can be further utilized to refine the process. We consider two possibilities for this refinement: specifying the conditions under which the query becomes derivable from the theory and isolating a minimal component of a not necessarily minimal answer.

4.1 Update Specifications

The cases when all the minimal models of DB satisfy the query or none of them does correspond to the usual (SURE) interpretation to affirming the query or its negation under CWA, respectively. The remaining case, when the query is satisfied in some but not all minimal models, leaves room for more refined answers. This is possible since the procedure partitions the set of minimal models of the theory into two subsets: the first is the one in which the query is *true* and the other is the set in which the query is *false*. By Lemma 2 the original theory $DB =_{mm} DB_{\{Q\}} \vee DB_{Neg(Q)}$, where $DB_{\{Q\}}$ is the ground theory with the minimal models $\mathcal{MM}(DB)_{\{Q\}}$ and $DB_{Neg(Q)}$ is the ground theory with the minimal models $\mathcal{MM}(DB)_{Neg(Q)}$ (recall Definitions 2.5 and Theorem 2).

The procedure in a sense returns the components of the minimal model structure of the theory that need to be updated to guarantee that the query becomes a SURE answer or to assume its negation under CWA. The issue of how to modify the theory to achieve the required properties has all the elements and complications of the database update problem and the full treatment of this issue has been a topic of extensive research [8]. For example, one can think of two ways under which a positive clause C can become a logical consequence of DB. The first is roughly to remove all models of the theory in which no atom of C appears. The second is to add every such model an atom (or more) of C so that the clause C becomes *true* in all minimal models of DB. The *reverse* needs to be done to make C *false* in all minimal models of DB. Similar reasoning can be applied to other query types. While the full treatment of this topic is beyond the scope of this paper, we give some glimpses about the possibility of achieving this goal by showing how to remove the unwanted minimal models of a theory. The method we outline is compatible with the way the procedure used for query answering operates. For brevity, we limit our treatment to the case of a positive query. The extension to the general case is straight forward.

Theorem 4 *Let DB be a DDDB and Q be a positive query. Let $DB =_{mm} DB_{Neg(Q)} \vee DB_{\{Q\}}$, where $DB_{Neg(Q)}$ and $DB_{\{Q\}}$ are the ground theories (model states) corresponding to the first and second branches of the model tree of DB, respectively. Then:*

- $DB' = DB \cup \{Neg(M) | M \in \mathcal{MM}(DB_{Neg(Q)})\} \vdash Q$. *Additionally,* $\mathcal{MM}(DB') = \mathcal{MM}(DB) \cap \mathcal{MM}(DB)_{\{Q\}} = \mathcal{MM}(DB)_{\{Q\}} = \mathcal{MM}(DB_{\{Q\}})$ *and* $DB' =_{mm} DB_{\{Q\}}$.

- $DB'' = DB \cup \{Neg(M) | M \in \mathcal{MM}(DB)_{\{Q\}}\}$ *and* $Neg(Q) \in EGCWA(DB'')$. *Additionally,* $\mathcal{MM}(DB'') = \mathcal{MM}(DB) \cap \mathcal{MM}(DB)_{Neg(Q)} = \mathcal{MM}(DB)_{Neg(Q)} = \mathcal{MM}(DB_{Neg(Q)})$. $DB'' =_{mm} DB_{Neg(Q)} =_{mm} DB \cup Neg(Q)$

Proof:
- The added constraints remove all undesirable models of DB: the minimal models of DB falsifying Q and the models of DB in which Q is *true* but are nonminimal for being subsumed by one of the abandoned minimal models[7].
- The added constraints remove all the models satisfying Q.

■

Consider the following example:

Example 10 *Let* $DB = \{P(a) \vee P(b), P(a) \vee P(c) \vee P(d), P(a) \vee P(d), P(b) \vee P(c) \vee P(e), P(c) \vee P(d) \vee P(e), P(a) \vee P(c) \vee P(d) \vee P(e)\}$, *and the query* $Q = P(b)$.
$\mathcal{MM}(DB) = \{\{P(a), P(c)\}, \{P(a), P(e)\}, \{P(b), P(d)\}\}$.
$DB_{\{P(b)\}} = DB \cup \{P(b)\} = \{P(b), P(a) \vee P(d), P(c) \vee P(d) \vee P(e)\}$.
$\mathcal{MM}(DB_{\{P(b)\}}) = \{\{P(b), P(a), P(c)\}, \{P(a), P(b), P(e)\}, \{P(b), P(d)\}\}$.
$DB_{Neg(P(b))} = DB \cup \{\neg P(b)\} = \{P(a), P(c) \vee P(e)\}$.
$\mathcal{MM}(DB_{Neg(P(b))}) = \{\{P(a), P(c)\}, \{P(a), P(e)\}\}$.
$DB' = DB \cup \{P(a) \wedge P(c) \rightarrow \bot, P(a) \wedge P(e) \rightarrow \bot\}$ *and* $DB' \vdash P(b)$.
$\mathcal{MM}(DB') = \{\{P(b), P(d)\}\}$.
$DB'' = DB \cup \{P(b) \wedge P(d) \rightarrow \bot\} = DB \cup \{\neg P(b)\}$.
$\mathcal{MM}(DB'') = \{\{P(a), P(c)\}, \{P(a), P(e)\}\}$ *and* $P(b) \in EGCWA(DB'')$.

The advantage of this approach to update is that it is passive: it involves only adding constraints that prune some models while retaining the original set of clauses. The update can be incorporated into the query answering procedure or left to the user. Other approaches to updates can be adopted to achieve particular results such as better efficiency or minimality of change to the original theory [8]. Note that while the input theory is not necessarily ground, the updates are specified in terms of ground clauses which can be a major simplification factor.

4.2 Answer Minimality

If the procedure returns *no* when looking for *yes/no* SURE answers to a positive query Q then there is no issue of minimality involved. However, it may be the case that both Q and a subset of Q are *yes* answers. In this case we would like to be able to report this fact or even to give the minimal component of Q that is still an answer.

Finding the minimal answer of a query is not a straightforward operation [14]. The brute force approach is to try all possible subsets of the query for derivability from the database, probably starting from sets with the least cardinality. Since the number of subsets of the query can be large, the amount of work needed may be prohibitive. Towards this aim we may utilize the order imposed on the search space of our minimal model generation procedure.

[7] Clearly, one can remove from each constraints atoms shared by all minimal models of the original theory. This may reduce the size of individual constraints, although not their number [24].

To answer a disjunctive query $Q = q_1 \vee ... \vee q_n$, the model tree is split into two branches: the first is the one with $Neg(Q)$ added to DB. If this returns a nonempty set of minimal models then Q is not an answer. If it returns no models then Q is a *yes* answer and further processing may be discontinued if we are not concerned about minimality. If minimality is of interest, the other branch of the model tree starts with Q as the top clause. Since Q is derivable from DB adding it will not change the resulting model structure: only minimal models of DB are returned.

Now we are set to partition the minimal models in which Q is *true*. In the first branch are all minimal models containing only q_1 and none of the other disjuncts of Q, if any. If this branch (the one with $\{q_1, \neg q_2, ..., \neg q_n\}$) is empty no judgment can be made about the membership of q_1 in a minimal answer. There is, however, some minimal answer component of Q not including q_1. We ignore q_1 and pursue that component as if we are testing the minimality of the *yes* answer $q_2 \vee ... \vee q_n$. On the other hand, if the branch with $\{q_1, \neg q_2, ..., \neg q_n\}$ is not empty then by Definition 2.14, q_1 is in a minimal component of Q. We pursue (one of) the minimal component of Q containing q_1.

The second branch will have models with q_2, possibly q_1 but none of $q_i, \forall i > 2$. By Definition 2.14, q_2 will be in the same minimal answer as q_1 only if this branch contains models in which q_1 is *false*. We pursue these models first within the second branch to decide if q_2 is in the minimal answer together with q_1. If the set with models of the form $\{q_2, \neg q_1, \neg q_3, ... \neg q_n\}$ is not empty then q_1 and q_2 are in the same minimal answer and we continue to pursue this answer. If the set with models of the form $\{q_2, \neg q_1, \neg q_3, ... \neg q_n\}$ is empty then q_2 need not be in the same minimal answer as q_1. The minimal answer is *true* in the models of the current branch since it has the disjunct q_1. The remaining minimal models will satisfy some elements of the set $\{q_3, ..., q_n\}$. Therefore, we can ignore q_2 and look for disjuncts from the set $\{q_3, ..., q_n\}$.

On processing element q_i we first check for the models containing none of elements q_j $(j < i)$ already shown to be in the minimal answer being developed. We proceed in the same spirit until the n^{th} branch. Note that when Q has more than one minimal component we pursue the rightmost among them. In this sense the result depends on the order of the atoms in Q.

The pseudo code in Algorithm 1 is a rough description of the approach to isolating a minimal component $MinAnswer$ of a *true* disjunctive query $Query$ (also represented as the set of its atoms $\{Query\}$) against a DDDB, *Database*. The minimal answer is to be interpreted as the disjunction of atoms in $MinAnswer$. $NotProcessed$ is the set of atoms of $Query$ not processed so far. $Neg(S)$ is the set of negative literals corresponding to atoms in S.

Algorithm 1 *Minimal Answer of a Disjunctive query Q.*

```
   Procedure FindMinAnswer(Database, Query);
   { Query is a ground disjunctive query represented as the set of its atomic
components {Query}*}

Let NotProcessed = {Query};

Let MinAnswer = {};

While NotProcessed ≠ {} do
     Let Atom = Next(NotProcessed);
     Let NotProcessed = NotProcessed \ {Atom};
```

```
If MM(Database ∪ Atom ∪ Neg(MinAnswer) ∪ Neg(NotProcessed)) ∩ MM(Database) ≠
{}
Then Let MinAnswer = MinAnswer ∪ {Atom};

Done;

Return (MinAnswer);
```

An example of the possible cases for a query $Q = a \vee b \vee c$ is displayed in Table 2. A "y" entry indicates a nonempty set of models with corresponding set of literals. The negative literals in the entry represent additional constraints on the model set to be tested for emptiness. For example, $[\neg a]$ is added to check for models with no a in addition to the constraints in the header. These constraints represent atoms that are already known to be in the minimal component. An X in the term status entry reflects the unknown nature of the inclusion of that atom in a minimal answer set (though it is definitely not in the minimal answer being constructed).

No.	$a, \neg b, \neg c \in M$	$b, \neg c \in M$	$c \in M$	Term Status a, b, c	Minimal Answer
0	$\emptyset$	$\emptyset$	$\emptyset$	N,N, N	$(\square \in DB)$
1	$\emptyset$	$\emptyset$	y	X,X, Y	(c)
2	$\emptyset$	y	$\emptyset$	X,Y, N	(b)
3	$\emptyset$	y	y $[\neg b]$	X,Y, Y	$(b + c)$
4	y	$\emptyset$	$\emptyset$	Y,N, N	(a)
5	y	$\emptyset$	y $[\neg a]$	Y,X, Y	$(a + c)$
6	y	y $[\neg a]$	$\emptyset$	Y,Y, N	$(a + b)$
7	y	y $[\neg a]$	y $[\neg a, \neg b]$	Y,Y, Y	$(a + b + c)$

Table 2: Checking the Minimality of a *yes* Answer $a + b + c$

Additionally, the order in which the atoms of the negative counterpart of Q were introduced into the refutation process usually bears information about the minimality of Q that can be further exploited to narrow down the search for minimally derivable components of Q. Note that the negation of Q is a set of negative unit clauses (denial rules) that can be incorporated into the refutation process one at a time. Clearly, if this approach is adopted then we can always assert that the disjunction of atoms the denials of which were used before the refutation was achieved cannot be nonminimal. Only the last added item can cause nonminimality. It is this information that we can use in the other component of the process to search for minimality.

5 Monotonicity Properties of Generalized Query Answering

The class of queries discussed span many of the applications encountered in database maintenance and exploitation. For each we considered both MAYBE and SURE answers. Of interest is the monotonicity of the query answering process. 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 for the query answering process can be nonmonotonic for DDDBs even for positive queries.

Definition 5.1 *(**monotonicity**) 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*[8]*: property π is monotonic if whenever π holds in DB then π also holds in DB^+.*

The following lemma is an extension of a result in [5] that relates the models of successive states of a DDDB before and after a clause addition update.

Lemma 3 *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 Theorem 1 and Corollaries 1 and 2 and Example 11. ∎

Consider the following example:

Example 11 $DB = \{a \vee b, c\}$. $DB^+ = \{a \vee b, c, a \rightarrow b\}$. $\mathcal{MM}(DB) = \{\{a, c\}, \{b, c\}\}$. $\mathcal{MM}(DB^+) = \{\{b, c\}\}$.

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 other atoms that were not previously derivable. The minimal model remains unchanged otherwise.

When adding clauses to a DDDB, DB, the models may remain the same or some may attempt to expand. The expansion process may, however, render some of the expanded models nonminimal for the updated theory and call for their removal (e.g. for being subsumed by other minimal models of DB^+). There may be minimal models of DB that are not subsets of any model of DB^+. It is important to determine which query answers are monotonic (and the queries themselves are *persistent* [22]) and which answers are nonmonotonic (and queries based on them are not persistent [22]). This can simplify inference revision after database updates. Next we explore this issue:

Theorem 5 *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 $Q = Body(Q) \rightarrow Head(Q)$.*

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

- *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 DB^+ be the result of updating DB by clause addition. If the added clause is negative (denial rule) then by Theorem 1, $\mathcal{MM}(DB^+) \subseteq \mathcal{MM}(DB)$ and the result is clear. Otherwise, by Theorem 3, 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 $Head(Q)$ is true in only 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, property P may not hold for DB^+.

- 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.

■

Note that while the content of individual models grows monotonically with respect to database growth (during the interval on which it is defined), the set of minimal models itself need not. The cardinality of the minimal model set may decrease as a result of such updates. The overall effect will be the nonmonotonic nature of some properties that are affected by this model pruning. It is possible that the number of elements having the property will not decrease if the relevant models are retained (maybe after expanding) or may decrease when some of the relevant models are rendered nonminimal.

Corollary 3 *Given a DDDB, DB, DB^+ the updated version of DB by clause addition and a query Q. Then:*

1. *If Q is positive:*
   - *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: Immediately follows from Theorem 5 and Definition 2.14 of answers[9]. ∎

Example 13 *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 [1] in the context of defining the sub-implication which is also based on minimal model properties[10]. 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 derivation of negative information under closed world reasoning (e.g. CWA, GCWA, EGCWA, ECWA) [15, 7, 25] is based on the absence of certain atoms in the minimal models and its nonmonotonicity is in line with the results of Theorem 5.

We considered only addition updates but didn't limit ourselves to adding positive clauses. Nonpositive clauses are allowed as well. Positive and mixed clause addition updates 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 Corollaries 1 and 2, including making DB^+ inconsistent.

It is possible also 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.

---

[9] A closely related result is the fact that reasoning under stable (and perfect) model semantics for the class of disjunctive databases with body negation (disjunctive normal databases) is nonmonotonic. Note that the set of stable (perfect) models is a subset of the set of minimal models of a database.

Example 12 *Let* $DB = \{notP(a) \rightarrow S(a)\}$*. The only stable (and perfect) model of DB is* $\{S(a)\}$*.* $Q = S(a)$ *is true (derivable) under the stable (and perfect) model semantics. However, for* $DB^+ = \{P(a), notP(a) \rightarrow S(a)\}$ *the only stable (and perfect) model of DB is* $P(a)$*.* $Q = S(a)$ *is not true (not derivable) under the stable (and perfect) model semantics in* DB^+ *and* $DB \subset DB^+$*.*

The detailed study of this issue is beyond the scope of this paper since it is not relevant for the class of theories considered here (perfect, stable and minimal models are the same for DDDBs). This fact may be viewed as an advantage of the minimal model semantics: *SURE* answers under this semantics are monotonic.

[10] Even with the assignment of the intermediate truth value 1/2 (1 being *true* and 0 being *false*) to elements that are MAYBE answer of the query, as in [1], the nonmonotonicity is still observed by noting that the truth value assignment may decrease after a database update.

Once more, the information returned by a generalized query answering procedure based on model generation can be utilized to decide the monotonicity properties of individual queries. As suggested by Theorem 3 and the model tree structure for generalized queries depicted in Figure 8, it is easy to see that a query Q is monotonic if and only if $\mathcal{MM}_2 = \mathcal{MM}(DB)$ for *yes* answers. Recall that $\mathcal{MM}_2$ is the set of minimal models of DB in which the head of the query is *true*.

In a sense, this is suggesting that the outlined procedure not only answers queries but also makes it possible tag them as *monotonic/nonmonotonic* in which case they may not need to be recomputed (no rechecking of constraints is needed) after an update, at no extra cost. Once a query is tagged as monotonic, future database updates will not affect its status and it need not be checked any more. This can be employed to enable incremental integrity constraints checking [16] and/or incremental construction of the minimal model tree for a theory. After an update, only nonmonotonic rules (queries) need to be rechecked. If not satisfied then further additions may be initiated to guarantee their satisfaction[11]. Actually, one may reduce the 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.

We note here that the source of nonmonotonicity for MAYBE answers to positive queries is confined to the disjunctive component of the database. Atoms that are definite, and consequently occur in every minimal model of the database, are monotonic by the virtue of them being SURE answers to a positive query. They cannot change status as a result of (clause addition) updates. Only indefinite atoms of the Herbrand base exhibit the nonmonotonic behavior. Any growth of the database will keep the set of derivable atoms as is or increase it (assuming the change keeps the database consistent). The Herbrand base is partitioned into two sets of atoms: those that are monotonic (definite) and those that are nonmonotonic (occur only in indefinite minimal clauses). Therefore, to determine the effect of an update on the status of different atoms of the Herbrand base revisions can be confined to atoms with only disjunctive occurrences in the theory (atoms that occur in a non-unit clause of the minimal model state of the database [12])

The nonmonotonicity of $MAYBE$ answers under the model state representation is reflected in the fact that while (clause addition) updates cannot shrink a clause derivable from the current state it can nevertheless generate a clause subsuming it. Therefore, an atom occurring in the minimal clause structure of the original database may have no occurrences in the representation of the updated theory. $SURE$ answers on the other hand are monotonic. A clause corresponding to a $SURE$ answer can only be subsumed by a newly generated clause which doesn't change the status of the $SURE$ answer, although it may affect its minimality.

6 Conclusion and Remarks

We presented an approach for 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 minimal model generating procedure . The concept of a query was extended to cover most classes of practical importance for database maintenance and exploitation. The efficiency of the approach depends on the efficiency of the used model generating procedure. Experiments with a prototype for our

[11]Note that a monotonically satisfied rule (a rule with a *true* head) acts like a query in that it adds nothing to the minimal model structure of the theory after an update.

procedure pointed to its efficiency as compared with similar ones reported in the literature [17, 19]. 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. However, this is a major improvement on approaches that produce models then compare them to detect nonminimality. 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 [17, 19, 29, 24]. 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 and optimization techniques will enable sharing of atoms between models [6, 24]. An added advantage of the advanced approach is that it deals with a single class of theories, namely DDDBs, at all stages of query evaluation. In principle, it is possible to convert queries to denials, and thus prevent their active participation in the model generation process, by moving head atoms to the body resulting in a *normal* disjunctive database. However, care must be observed when processing negative body atoms to ensure the correctness of the query evaluation process. Concepts like stratification and perfect models come into play [5, 29].

Approaches based on model generation are bottom-up in nature and tend to explore a much larger than needed search space for answering a query[12]. In our case the extensive use of negative clause addition to the theory can be viewed as a way to prune the search space and thus obtain shallower refutations. Additionally, we used the wealth of information returned by the outlined approach to refine the query answering process in more than one direction: to decide the minimality of disjunctive answers and to isolate a minimal component of a nonminimal answers. The results returned by the procedure were also used to determine the updates (e.g. in the form of ground denial rules) that need to be incorporated into the theory so that to guarantee certain properties of the returned answers.

We also made a 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 is a byproduct of the query answering process. This was shown to be useful for incremental construction of the minimal model structure of the theory.

The definition of a $MAYBE$ answer adopted here differs from the *maybe* components of the I-tables and M-tables in [10] and [11]. There, A is a *maybe* answer if it occurs in a positive clause derivable from the database even when all such clauses are subsumed by others not containing A. A need not be in some minimal model of the current state of the database (or a component of a minimal indefinite answer) although it may have been in such a model at an earlier stage. In essence, that is treating disjunctions inclusively rather than exclusively as we do in our case which is based on minimal models. The approach in [10] and [11] relies on minimal model constructs as well to derive

[12]Reference [28] offers an approach to using a minimal model generation procedure to perform top-down answering of positive queries in disjunctive deductive databases.

sure answers. However, *maybe* components are not based on a minimal model characterization but rather on the syntactic structure of the (historical) database (and on the inclusive interpretation of disjunctions). We believe that basing *sure* answers on the minimal model structure that naturally assumes an exclusive interpretation of disjunctions while basing *maybe* answers on syntactical history of the database and an inclusive interpretation of disjunctions will complicate closed world inferences and may not be appropriate for all types of real life disjunctions. It may result in an atom classified both as a *possible/maybe* answer and belonging to the database completion, which are, in our view, two not compatible properties. The structures developed for maintaining the ground clauses of the extended relational database in [10] and [11] (I-tables, M-tables) may be utilized as appropriate data structures in our approach. The minimal model generation algorithm described here may also be utilized to generate the information content (minimal model structure) of the these tables.

Our definition also differs from that of a possible (*pos*) answers of [23], which, while relying on the derivability of the answer from certain components of the database, does not limit itself to the case of minimal models of the theory. Certain possible answers may be in no minimal model of the theory but in some disjunctive clause possibly subsumed by another clause of the theory.

Basing the definition of an answer on nonminimal models or on structures that are not models of the current state of the database necessitate redefining the concept of database closure (GCWA, EGCWA, ECWA) [15, 7, 25] which are defined in terms of the minimal model structure of the theory.

In a DDDB context, the *maybe* and *pos* atoms of [10] and [23] (with possible participation of definite data) will still participate in generating new atoms with a similar *truth assignment*, similar to the $MAYBE$ answers in our case. An advantage of basing (*maybe* and *pos*) on syntactic features is better monotonicity behavior that is not enjoyed by the $MAYBE$ answers as defined here.

The issue of minimality of answers addressed here was addressed in other approaches to query answering as well [14]. In addition to the modified notion of an answer, our approach tries to exploit the properties of the model generating process and the order it imposes on the minimal models returned to determine the minimality of answers rather than using multiple applications of the model generation procedure to subsets of the required (ground) answer. Testing for answer minimality when the minimal model state is used is part of the query evaluation process and comes at no additional cost.

The approaches for efficient query evaluation discussed in [26, 4] can be applied as pre-processing step in our approach.

The use of a similar approach to answering queries under different database semantics such as stable and perfect semantics and the development of an integrated system based on using minimal model generators for different aspects of database processing such as integrity enforcement are among the possible topics of future work.

Acknowledgments

I thank F. Bry for useful discussions and suggestions on the topic of this paper. This research was done while the author was visiting with the PMS-Group at the Institut für Informatik, LMU München on an Alexander von Humboldt Research Fellowship. The support of the PMS-Group and the Alexander von Humboldt Stiftung is appreciated.

References

[1] G. Bossu and P. Siegel. Saturation, nonmonotonic reasoning and the closed-world assumption. *Artificial Intelligence*, 25(1):13–63, January 1985.

[2] F. Bry. Personal Communications. 1995.

[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] U.S. Chakravarthy, J. Grant, and J. Minker. Foundations of semantic query optimization for deductive databases. In J. Minker, editor, *Proc. Workshop on Foundations of Deductive Databases and Logic Programming*, pages 67–101, Washington, D.C., August 1986.

[5] 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.

[6] 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.

[7] M. Gelfond, H. Przymusinska, and T.C. Przymusinski. The extended closed world assumption and its relation to parallel circumscription. *Proc. Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 133–139, 1986.

[8] J.Grant, J.Horty, J. Lobo, and J. Minker. View updates in disjunctive deductive databases. *Journal of Automated Reasoning*, 11:249–267, 1993.

[9] 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.

[10] K.C. Liu and R. Sunderraman. Indefinite and maybe information in relational databases. *ACM Transactions on Database Systems*, 15(1):1–39, 1990.

[11] Ken-Chih Liu and Rajshekhar Sunderraman. A generalized relational model for indefinite and maybe information. In *IEEE Transactions on Knowledge and Data Engineering*, pages 65–77, March 1991.

[12] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.

[13] 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.

[14] J. Minker and J. Grant. Answering queries in indefinite databases and the null value problem. In P. Kanellakis, editor, *Advances in Computing Research*, pages 247–267. 1986.

[15] J. Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science 138*, pages 292–308. Springer-Verlag, 1982.

[16] J-M. Nicolas. Logic for improving integrity checking in relational data bases. *Acta Informatica*, 18(3):227–253, 1982.

[17] 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.

[18] 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.

[19] H. Schüz and T. Geisler. Efficient model generation through compilation. In *Proceedings of the 13^{th} International Conference on Automated Deduction (CADE)*, New Jersey, USA, July 1996. Springer-Verlag. LNCS series.

[20] D. Seipel. *Efficient Reasoning in Disjunctive Deductive Databases*. Habilitation thesis, Fakultät für Informatik, Universität Tübingen, June 1995.

[21] M. Suchenek. First-order syntactic characterizations of minimal entailment, domain minimal entailment and herbrand entailment. *Journal of Automated Reasoning*, 10:237–236, 1993.

[22] G. Wagner. Disjunctive, deductive and active knowledge bases. Technical Report 22/94, Free University of Berlin, Free University of Berlin, Germany, 1994.

[23] M. H. Williams and Q. Kong. Incomplete information in a deductive database. *Data and Knowledge Engineering*, 3(3):197–220, March 1988.

[24] 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.

[25] A. Yahya and L.J. Henschen. Deduction in Non-Horn Databases. *J. Automated Reasoning*, 1(2):141–160, 1985.

[26] A. Yahya and J. Minker. Query evaluation in partitioned disjunctive deductive databases. *International Journal of Intelligent and Cooperative Information Systems*, 3(4):385–414, 1994.

[27] A. Yahya and J. Minker. Representations for disjunctive deductive databases. Technical Report CS-TR-3111 UMIACS-TR-93-70, Department of Computer Science and UMIACS, University of Maryland, College Park, July 1993.

[28] A. Yahya. A goal-driven approach to efficient query processing in disjunctive theories. Technical Report PMS-FB-1996-12, Department of Computer Science, University of Munich, Munich, Germany, July 1996. WWW: http://www.informatik.uni-muenchen.de/pms/publikationen/berichte/PMS-FB-1996-12.ps.gz.

[29] 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.