Duality for Goal-Driven Query Processing in Disjunctive Deductive Databases

Article *in* Journal of Automated Reasoning · January 2002

DOI: 10.1023/A:1020109502432 · Source: DBLP

1 author:

Adnan Yahya
Birzeit University
41 PUBLICATIONS **439** CITATIONS

SEE PROFILE

Duality for Goal-Driven Query Processing in Disjunctive Deductive Databases

Adnan Yahya
Electrical Engineering Department, Birzeit University, Birzeit, Palestine
Email: yahya@ee.birzeit.edu WWW:http://www.birzeit.edu/eng/yahya

Abstract

Bottom-up query answering procedures tend to explore a much larger search space than is strictly needed. Top-down processing has a more focused search space which can result in more efficient query answering. We establish a strong connection between model generation and clause derivability that allows us to use a bottom-up procedure for evaluating queries in a top-down fashion. The approach requires no extensive rewriting of the input theory and introduces no new predicates. Rather, it is based on a certain *duality* principle for interpreting logical connectives. It is achieved by reversing the direction of implication arrows in the clauses representing both the theory and the negation of the query. The application of a generic bottom-up procedure to the transformed clause set results in top-down query answering. We give meaning to this transformation and show how it can be utilized for refined query answering by specifying the *minimal* conditions (weakest updates) under which the query becomes derivable from the theory.

1 Introduction

Bottom-up query processing tends to explore a search space much larger than that required for the answer. In contrast, top-down methods perform a more focused search for refutations by utilizing the information contained in the query. Several approaches based on transforming the set of clauses representing the theory or using certain data structures and special algorithms to achieve top-down processing for a given query are available. Most of the transformations involve the introduction of new predicates and/or extensive rule rewriting to enable the more focused search [1, 5, 8, 14, 16, 19].

In this paper we offer an alternative method for top-down processing of positive queries based on a certain concept of *duality*. It exploits the implicit modification of the interpretation of logical connectives induced by the reversal of the implication sign of clauses. No additional modification of the theory is needed. Effectively the same procedure can be used both for bottom-up and top-down query processing, depending on whether the procedure is applied to the theory or to its dualization.

We utilize the information returned by the procedure to allow for refined query answering. The refinement consists of specifying the minimal insertion updates needed for the query to become derivable from the updated theory. We show that our approach makes it possible to split the query answering process into two stages: the first is the generation of the minimal checks needed to ensure the derivability of the query. This is based on the interaction of the query with the *Intensional* part of the database (IDB). The second is the actual look-up of these conditions in the *Extensional* part of the database (EDB). Alternatively, one could integrate the two stages to get shallower computations with the context and associated costs determining the exact choice.

The remainder of the paper is organized as follows. In the next section we give some relevant definitions and background material. In Section 3 we define our *duality-based* procedure for a restricted class of disjunctive theories: that of ground databases with no denial rules. We establish a strong relationship between the set of clauses that need to be derivable in order for a positive query to be *true* in the theory and the set of minimal models of the dualization of the query and the theory. In Section 4 we show how to relax the restrictions on our procedure and the problems involved in allowing for more general theories and queries. We also offer some interpretation and implementation notes on the advanced approach to explain the sources of its improved performance, potential and limitations. In Section 5 we compare our approach with others advanced in the literature and point to possible directions of further research.

2 Preliminaries and Background Material

We assume familiarity with the basic concepts of deductive databases as, e.g., in [11] and limit ourselves to the basic material needed for presenting the results of this paper.

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

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

where $m, n \geq 0$ and the As and Bs are atoms in a First Order Language (FOL) $\mathcal{L}$ with no function symbols. By $Head(C)$, ($Body(C)$) we denote the set of atoms in the head (body) of a clause C of DB. We write C as $Body(C) \rightarrow Head(C)$. Atoms of $Body(C)$ ($Head(C)$) are interpreted conjunctively (disjunctively). Atom $\perp$ (false) refers to the empty head and atom $\top$ (true) refers to the empty body.

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 $\mathcal{L}$. The *disjunctive Herbrand base* is the set of all (finite) positive ground disjunctions formed from the elements of the herbrand base. A *Herbrand interpretation* is any subset of HB_{DB}. A *Herbrand model* of DB, M, is a Herbrand interpretation such that $M \models DB$ (all clauses of DB are *true* in M). A model 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 DDDB, DB, can be partitioned into three sets of clauses:*

1. *The* extensional part *(EDB): a positive disjunctive database corresponding to base relations and containing facts (clauses with empty bodies, positive clauses).*
2. *The* intensional part *(IDB): used to derive new pieces of information.*
3. *The* integrity constraints *(IC_{DB}): used to ensure that the theory satisfies certain properties. We limit integrity constraints to denial constraints (clauses with empty heads).*

Definition 2.3 *A query Q is positive if it can be translated into a set of positive clauses.*

We limit our consideration to positive queries. This includes *disjunctive* queries which are disjunctions of atomic queries and *conjunctive* queries which are conjunctions of atomic queries. Other positive queries can be reduced to a conjunction of disjunctive queries. Queries are assumed ground and have *yes/no* answers. We treat both clauses and models as sets of ground atoms. A set of atoms is interpreted disjunctively when it is referred to as a clause and conjunctively when it is referred to as a model. We can even talk about equality of a clause and a model which is to be interpreted as them having the same underlying set. Unless otherwise stated, we assume that the database under consideration is consistent.

3 The Duality Approach

3.1 Goal Set Expansion

Theorem 1 *Let DB be a ground DDDB (with no denial constraints) and $Q = q_1 \vee ... \vee q_n$, ($\{q_1, ..., q_n\} \subseteq HB_{DB}$), be a ground disjunctive query. If $DB \vdash Q$ then a clause subsuming Q occurs in the head of some clause of DB: $\exists C \in DB$ s. t. $Head(C) \cap Q = Head(C)$.*

Proof: Q is derivable from DB iff it is *true* in every model of DB. Assume that Q is derivable from DB and there exists no clause $C \in DB$ such that $Head(C)$ subsumes Q. Consider $M = HB(DB) \setminus \{q_1, ..., q_n\}$. It satisfies all clause heads since none of them consists entirely of atoms in Q. M is a model of DB and $M \not\models Q$. A contradiction. ∎

The following example shows that Theorem 1 doesn't hold in the presence of denial constraints:

Example 1 *Consider $DB_1 = \{P(d) \rightarrow P(a) \vee P(b) \vee P(c), P(d)\}$ and $DB_2 = \{P(d) \rightarrow P(a) \vee P(b) \vee P(c), P(d), P(c) \rightarrow \bot\}$ and $Q = P(a) \vee P(b)$. Clearly $DB_2 \vdash Q$ but $DB_1 \nvdash Q$.*

In view of Theorem 1, if a subclause of Q is in the EDB then the query is *true*. This can be verified by inspection. If not, we need to look for a clause C with a subclause of Q as (instance of) the head and show that C will derive Q at least in all models of DB in which Q may not be satisfied, ensuring the derivability of Q. One way to do that is by showing that atoms of the body of C are all elements of the EDB. However, this is too strong a condition: it is sufficient but not necessary.

We develop the necessary and sufficient conditions based on results reported in [5, 10, 14, 22].

Definition 3.1 *Let DB be a ground DDDB (with no denial constraints), $Q = q_1 \vee ... \vee q_n$ be a disjunctive query against DB and $C = B_1 \wedge ... \wedge B_k \rightarrow A_1 \vee ... \vee A_l$ be a clause in DB. We define $\mathcal{G}_C^Q$, the goal clause set of Q relative to C, as:*

$$\mathcal{G}_C^Q = \begin{cases} \{q_1 \vee ... \vee q_n \vee B_1, ..., q_1 \vee ... \vee q_n \vee B_k\} & \text{if } \{A_1, ..., A_l\} \subseteq \{q_1, ..., q_n\} \\ \{q_1 \vee ... \vee q_n\} & \text{if } \{A_1, ..., A_l\} \not\subseteq \{q_1, ..., q_n\} \end{cases}$$

Two special cases of the first condition are of interest. The first is when $Head(C)$ is empty ($\bot$). $Head(C)$ is a subset of every set. The condition always holds and expansion is possible on all atoms of $Body(C)$. The second is when $Body(C)$ is empty ($\top$) and the condition ($\{A_1, ..., A_l\} \subseteq \{q_1, ..., q_n\}$) holds. Expanding Q by $\top$ is equivalent to $\top$: resulting in the empty expansion set.

$\mathcal{G}_C^Q$ is meant to define a set of clauses, that need to be derivable from (E)DB to prove that Q is derivable from DB through clause C. While we are replacing the provability of one clause Q by the provability of several, the latter are longer disjunctions and, therefore, each of them has a better chance of being proved. Clearly it is possible to delete all nonminimal elements from the clause set $\mathcal{G}_C^Q$. In particular, if $\{A_1, ..., A_l\} \subseteq \{q_1, ..., q_n\}$ and $\{B_1, ..., B_k\} \cap \{q_1, ..., q_n\} \neq \emptyset$ then $\mathcal{G}_C^Q$ is subsumed by the single clause $q_1 \vee ... \vee q_n$ and therefore $\mathcal{G}_C^Q = \{q_1 \vee ... \vee q_n\}$ in this case too. We make this explicit when defining the consequence operator $\mathcal{T}_{DB}^g$ later in this section (Definition 3.2). By $(Body(C) \vee Q)$ we denote the set $\{q_1 \vee ... \vee q_n \vee B_i | B_i \in Body(C)\}$.

Theorem 2 *Let DB be a ground DDDB (with no denial constraints) and $Q = q_1 \vee ... \vee q_n$ be a disjunctive query. $DB \vdash Q$ if and only if there exists a clause C in DB such that $Head(C)$ subsumes Q and the formula $(Body(C) \vee Q)$ is derivable from DB.*

Proof: Assume that the conditions hold but Q is not *true* in some model of DB, say M. From $M \models (Body(C) \vee Q)$ we get that $Body(C) \subseteq M$. Consequently $Head(C) \subseteq Q$ is *true* in M. Q needs to be *true* in M by an application of clause C. A contradiction.

Now, assume that for all relevant clauses C of DB, $(Body(C) \vee Q)$ is not derivable from DB. Consider a suitable model M of DB: there is a clause $(B \vee Q)$ for some $B \in Body(C)$ such that both B and Q are false in M. C is satisfied (but does not fire) in M. Q is not derivable from DB (it is false in M). ∎

Corollary 1 *[10] Under the conditions of Theorem 2 and let C be as defined there:*

1. *Q is derivable from DB iff $(Body(C) \vee Q)$ is derivable from $DB \setminus \{C\}$.*
2. *If $Body(C) \cap Q \neq \emptyset$ then $DB \vdash Q$ iff $DB \setminus \{C\} \vdash Q$.*
3. *$DB \nvdash Q$ if and only if for all clauses C in DB such that $Head(C)$ subsumes Q, $(Q \vee Body(C))$ is not derivable from DB.*

Proof: 1. If $Body(C) \vee Q$ is derivable from $DB \setminus \{C\}$ then Q is derivable from DB by the application of clause C.

Let $Body(C) \vee Q$ be derivable from DB. Assume that $DB \setminus \{C\} \nvdash (Body(C) \vee Q)$. There exists an interpretation M s. t. $M \models (DB \setminus \{C\})$ but $M \not\models Body(C)$. $Body(C) \cap M \neq Body(C)$. $M \models C$ and consequently $M \models DB$ and therefore $M \models (Body(C) \vee Q)$. A contradiction.

2. Immediate since $Body(C) \vee Q$ is subsumed by Q. That is: $Q \in \{(B \vee Q) \mid B \in Body(C)\}$.
3. Straightforward.

∎

Repeated application of Theorem 2 may be needed to finish the task. Consider the example:

Example 2 *Let $DB_3 = \{C_1 = a \vee b, C_2 = c \vee d, C_3 = a \wedge d \rightarrow e \vee b, C_4 = b \wedge d \rightarrow f \vee c, C_5 = c \rightarrow g\}$. Let $Q_1 = b \vee e \vee g$, $Q_2 = b \vee c \vee f$ and $Q_3 = g \vee f$.*

1. *$Q_1 = b \vee e \vee g$: Applying the clause C_3 to Q_1 yields the set $\{C_6 = b \vee e \vee g \vee a, C_7 = b \vee e \vee g \vee d\}$. $EDB \vdash C_6$ (subsumed by C_1) while C_7 is not so we apply clause C_5 to C_7 to yield $\{C_8 = b \vee e \vee g \vee d \vee c\}$ which is derivable from EDB (subsumed by C_2) and consequently so is Q_1.*
2. *$Q_2 = b \vee c \vee f$: Applying the clause C_4 to Q_2 yields the set $\{C_9 = c \vee f \vee b, C_{10} = b \vee c \vee f \vee d\}$. C_{10} is derivable from EDB while C_9 is not. All attempts to solve C_9 (through C_4) fail and therefore $DB \nvdash Q_2$. Note that we could have generated C_9 alone since $Q_2 \cap Body(C_4)$ is not empty without changing the final result (actually, no expansion of Q_2 is possible).*
3. *$Q_3 = g \vee f$: Applying the clause C_5 to Q_3 yields the set $\{C_{11} = c \vee g \vee f\}$. C_{11} is not derivable from EDB. Applying the clause C_4 to C_{11} yields the set $\{C_{12} = c \vee g \vee f \vee b, C_{13} = c \vee g \vee f \vee d\}$. C_{13} is derivable from EDB while C_{12} is not. No clauses are available to extend C_{12} so $DB \nvdash Q_2$.*

Theorem 2 and Corollary 1 suggest a simple approach to proving Q. Keep generating clauses until a sufficient group is subsumed by clauses in DB. The following points need to be emphasized:

- The expansion of clauses is continued until the elements of the goal clause set are found to be subsumed by elements in EDB or until no more applications of the rules of IDB are possible.
- The top-down nature of the expansion process since the clauses are selected on their (head) matching (subsuming) the goal clause (current query).
- The clause (instance) used for expansion is applied only once and it can be deleted from the database during the rest of the search process, thus guaranteeing termination.

Definition 3.2 *Let $DB = EDB \cup IDB$ be a ground DDDB (with no denial constraints), Q be a ground positive clause and $\mathcal{S}$ be a set of ground positive clauses. We define the consequence operator $\mathcal{T}^g_{DB}$ that maps sets of positive clauses into sets of positive clauses of the disjunctive Herbrand base of DB as follows [6, 9, 18]:*

$$\mathcal{T}^g_{DB}(\{Q\}) = \begin{cases} \{Q \vee B | & \exists C \in DB \ s.t. Head(C) \subseteq Q, Body(C) \cap Q = \emptyset, B \in Body(C)\}. \\ \{Q\} & Otherwise \end{cases}$$

$\mathcal{T}^g_{DB}(\mathcal{S}) = \bigcup_{C \in \mathcal{S}} \mathcal{T}^g_{DB}(\{C\})$,

$\mathcal{T}^g_{DB} \uparrow 0(\mathcal{S}) = \mathcal{S}$,

$\mathcal{T}^g_{DB} \uparrow \alpha(\mathcal{S}) = \mathcal{T}^g_{DB}(\mathcal{T}^g_{DB} \uparrow (\alpha \perp 1)(\mathcal{S}))$ *for successor ordinal* α,

$\mathcal{T}^g_{DB} \uparrow \alpha(\mathcal{S}) = lub\{\mathcal{T}^g_{DB} \uparrow \beta(\mathcal{S}) : \beta < \alpha\}$ *for limit ordinal* α,

$lfp(\mathcal{T}^g_{DB}) = \mathcal{T}^g_{DB} \uparrow \omega(\mathcal{S})$, *where ω is the first limit ordinal.*

We may assume that subsumed clauses are deleted (minimization is performed) at each stage. This is so since subsumed clauses are automatically derivable from DB when the subsuming clauses are[1]. However, we elect not to adopt this assumption to stay as general as possible.

Example 3 *Consider $DB = EDB \cup IDB = \{C_1 = a \vee b, C_2 = c \vee d\} \cup \{C_3 = a \wedge d \rightarrow e \vee b, C_4 = b \wedge d \rightarrow f \vee c, C_5 = c \rightarrow g\}$. Let $Q_1 = b \vee e \vee g$, $Q_2 = b \vee c \vee f$ and $Q_3 = g \vee f$ as in Example 2.*
$\mathcal{T}^g_{IDB} \uparrow 1(\{Q_1\}) = \{C_6 = b \vee e \vee g \vee a, C_7 = b \vee e \vee g \vee d, C_8 = b \vee e \vee g \vee c\}$.
$\mathcal{T}^g_{IDB} \uparrow 2(\{Q_1\}) = \{C_9 = b \vee e \vee g \vee a \vee c, C_{10} = b \vee e \vee g \vee d \vee c\} = \mathcal{T}^g_{IDB} \uparrow \omega(\{Q_1\})$.
$\mathcal{T}^g_{IDB} \uparrow 1(\{Q_2\}) = \{C_{11} = b \vee c \vee f\} = \mathcal{T}^g_{IDB} \uparrow \omega(\{Q_2\})$.
$\mathcal{T}^g_{IDB} \uparrow 1(\{Q_3\}) = \{C_{12} = g \vee c \vee f\}$.
$\mathcal{T}^g_{IDB} \uparrow 2(\{Q_3\}) = \{C_{13} = g \vee c \vee f \vee b, C_{14} = g \vee c \vee f \vee d\} = \mathcal{T}^g_{IDB} \uparrow \omega(\{Q_3\})$.
Note that the expansion can produce nonminimal elements. For example, if we let $DB' = DB \cup \{C_0 = f \wedge d \rightarrow a \vee g\}$ then applying C_0 to C_9 gives $\mathcal{T}^g_{IDB'} \uparrow \omega(\{Q_1\}) = \{C_{14} = b \vee e \vee g \vee a \vee c \vee d, C_{15} = b \vee e \vee g \vee a \vee c \vee f, C_{10} = b \vee e \vee g \vee d \vee c\}$. C_{14} is nonminimal. It is subsumed by C_{10}.

Later we prove that if we start with a disjunctive query Q, then $\mathcal{T}^g_{DB} \uparrow \omega(\{Q\})$ is the (goal, hence the superscript g) set of clauses that all need to be subsumed by EDB in order to prove Q.

[1] Note, however, that the partial order on sets of interpretations (sets of clauses in our case) $\mathcal{I}$ and $\mathcal{J}$ is defined through the set inclusion relationship ($\subseteq$) between the *minimal elements* of the sets. That is, $\mathcal{J} \sqsubseteq \mathcal{I}$ if and only if $\forall I \in Min(\mathcal{I}), \exists J \in Min(\mathcal{J}), J \subseteq I$, where $Min(\mathcal{I}) = \{I | I \in \mathcal{I}, \ \nexists I' \in \mathcal{I} \ s.t. \ I' \subset I\}$. So, rather than minimizing at every step, we minimize after the complete computation. Clearly, nonminimal elements will have no effect on the partial order and the results of [6] hold here as well.

3.2 The Duality Transformation

Definition 3.3 *(**dual clause**) Let $C = Body(C) \rightarrow Head(C)$ be a clause of a DDDB, DB. We define the dual clause of C, $C^d = Head(C) \rightarrow Body(C)$. $Head(C) = Body(C^d)$ and $Body(C) = Head(C^d)$. $\bot$ ($\top$) in the head (body) of C is replaced by $\top$ ($\bot$) in the body (head) of C^d. The dual of a set of clauses $\mathcal{S}$ is the set $\mathcal{S}^d$ of the duals of each of the members of $\mathcal{S}$.*

Next we establish a strong connection between $\mathcal{T}^g_{IDB} \uparrow \omega(\{Q\})$ and the minimal models of the dual theory and query.

Lemma 1 *Let DB be a ground DDDB (with no denial constraints), $Q = q_1 \vee ... \vee q_n$ be a disjunctive query against DB, $Q^d = \{q_1, ..., q_n\}$, and $C = B_1 \wedge ... \wedge B_k \rightarrow A_1 \vee ... \vee A_l$ be a clause in DB. Then*[2]

- $\mathcal{MM}(\{C^d\} \cup Q^d) \subseteq \mathcal{G}^Q_C$
- $\mathcal{MM}(\{C^d\} \cup Q^d) = Min(\mathcal{G}^Q_C)$

Proof: By definition $C^d = A_1 \wedge ... \wedge A_l \rightarrow B_1 \vee ... \vee B_k$. Three cases are possible:

1. If $Q^d \cap \{A_1, ..., A_l\} \neq \{A_1, ..., A_l\}$ then $\mathcal{MM}(\{C^d\} \cup Q^d) = \{Q^d\} = \mathcal{G}^Q_C$.
2. If $Q^d \cap \{A_1, ..., A_l\} = \{A_1, ..., A_l\}$ and $Q^d \cap \{B_1, ..., B_k\} \neq \emptyset$ then $\mathcal{MM}(\{C^d\} \cup Q^d) = \{Q^d\} \subset \mathcal{G}^Q_C$ and since $Min(\mathcal{G}^Q_C) = \{Q^d\}$ then $\mathcal{MM}(\{C^d\} \cup Q^d) = Min(\mathcal{G}^Q_C)$.
3. If $Q^d \cap \{A_1, ..., A_l\} = \{A_1, ..., A_l\}$ and $Q^d \cap \{B_1, ..., B_k\} = \emptyset$ then $\mathcal{MM}(\{C^d\} \cup Q^d) = Min(\{\{Q^d \cup \{B_1\}\}, ...\{Q^d \cup \{B_k\}\}\}) = Min(\mathcal{G}^Q_C)$.

■

Theorem 3 *Let DB be a ground DDDB (with no denial constraints) and $Q = q_1 \vee ... \vee q_n$ be a disjunctive query. Let $IDB^d_Q = Q^d \cup IDB^d = \{q_1, ..., q_n\} \cup \{C^d | C \in IDB\}$. Then:*

- $\mathcal{MM}(IDB^d_Q) \subseteq \mathcal{T}^g_{IDB} \uparrow \omega(\{Q\})$.
- $\mathcal{MM}(IDB^d_Q) = Min(\mathcal{T}^g_{IDB} \uparrow \omega(\{Q\}))$.

Proof: (Sketch). $\forall M \in \mathcal{MM}(IDB^d_Q)$, $Q^d \subseteq M$. $\mathcal{T}^g_{IDB} \uparrow 0(\{Q\}) = \{Q\}$.

$\mathcal{T}^g_{IDB} \uparrow 1(\{Q\}) = \{Q \vee B | \exists C \in IDB \; s.t. Head(C) \subseteq Q, Body(C) \cap Q = \emptyset \; and \; B \; in \; Body(C)\}$. By Lemma 1 the minimal models of $\{C^d\} \cup Q^d$ are in the set $\mathcal{T}^g_{DB} \uparrow 1(\{Q\})$. Using this as the base step i, an induction step can be constructed by applying the operator $\mathcal{T}^g_{IDB}$ to the set $\mathcal{T}^g_{IDB} \uparrow 1(\{Q\})$ and using Lemma 1 to show that the minimal models of the set consisting of the atoms of each clause generated at step i and a matching clause in IDB^d are among the elements returned by the new application of the operator $\mathcal{T}^g_{IDB}$. ■

[2] There is a slight abuse of notation here as far as Q and Q^d are concerned. When $Q = q_1 \vee ... \vee q_n$, the negation of Q, $Neg(Q) = q_1 \vee ... \vee q_n \rightarrow \bot = \{q_1 \rightarrow \bot, ..., q_n \rightarrow \bot\}$. The dual of $Neg(Q) = (Neg(Q))^d = \{\top \rightarrow q_1, ..., \top \rightarrow q_n\} = \top \rightarrow q_1 \wedge ... \wedge q_n$. That is $Neg(Q) = q_1 \vee ... \vee q_n \rightarrow \bot$ and $(Neg(Q))^d = \top \rightarrow q_1 \wedge ... \wedge q_n$. So the correct reading of Q and Q^d is that we are using the negation of Q during the forward chaining mode and the dual of that negation in the top-down computation. It happens that when $Q = q_1 \vee ... \vee q_n$ then the dual of the negation of Q is $\top \rightarrow q_1 \wedge ... \wedge q_n = Q^d = \{q_1, ..., q_n\}$. For a conjunctive query $Q = q_1 \wedge ... \wedge q_n$, $Neg(Q) = q_1 \wedge ... \wedge q_n \rightarrow \bot$. The dual of this negation is $\top \rightarrow q_1 \vee ... \vee q_n$. Here also $Q^d = q_1 \vee ... \vee q_n$, as expected. See also paragraph 4.4 for a related discussion.

Theorem 4 *Under the conditions of Theorem 3: Q is derivable from DB if and only if: EDB derives the clause set $\{C|C \in Min(\mathcal{T}^g_{IDB} \uparrow \omega(\{Q\}))\}$;*
or equivalently $\forall M = \{A_1, ..., A_l\} \in \mathcal{MM}(IDB^d_Q), EDB \vdash C_M = A_1 \vee ... \vee A_l$.

Proof:

- Let M be in $\mathcal{MM}(IDB^d_Q)$ and Q be derivable from DB. We show that $EDB \vdash C_M$. Assume that EDB does not derive C_M. There exists a model M' of EDB such that $M' \cap M = \emptyset$. Clearly $M' \not\models Q$, since $Q^d \subseteq M$. We extend M' into a model of DB, say M'', such that $M'' \not\models Q$ and thus get a contradiction. For any clause $C \in IDB$:
  1. If C^d fired during the generation of M then C is trivially satisfied in M' since at least one of the body atoms of C is not in M'. That is: $\forall C \in IDBs.t.Head(C) \subseteq C_M, \exists A \in (Body(C) \cap M)$ and $A \notin M'$ since $M' \cap M = \emptyset$. $M' \models C$.
  2. For other clauses, if $M' \models Body(C)$ but $M' \not\models Head(C)$ then add an atom $A \notin M$ of $Head(C)$ to M'. The resulting M'' is a superset of M', a model of DB, and $M'' \not\models Q$.
- If $\forall M \in \mathcal{MM}(IDB^d_Q)$, EDB$\vdash C_M$ we show that $DB \vdash Q$. If not, $\exists N \in \mathcal{MM}(DB)$ such that $N \not\models Q$. We describe how to extend Q^d into an $M \in \mathcal{MM}(IDB^d_Q)$ such that $N \cap M = \emptyset$ and therefore $N \not\models C_M$.
  1. Let $i := 0$ and Let $M^0 := Q^d$;
  2. Clearly, $N \cap M^i = \emptyset$ (recall that Q is a disjunctive query).
     If M^i is subsumed by an element of $\mathcal{MM}(IDB^d_Q)$ then exit with M equal the element of $\mathcal{MM}(IDB^d_Q)$ subsuming M^i.
     Otherwise there exists a clause $C_i \in IDB$ such that $Head(C_i) \cap M^i = Head(C_i)$. Since $N \not\models Body(C_i)$ there must be an atom $A_i \in Body(C_i)$ such that $A_i \notin N$.
     Now let $M^{i+1} := M^i \cup \{A_i\}$; $i := i + 1$ and go to step 2.

  Since all models are finite, the process terminates generating an $M \in \mathcal{MM}(IDB^d_Q)$ such that $N \cap M = \emptyset$, $N \models DB$ and $N \not\models C_M$. A contradiction.

■

Example 4 *Consider the theory of Example 2: $DB = \{C_1 = a \vee b, C_2 = c \vee d, C_3 = a \wedge d \rightarrow e \vee b, C_4 = b \wedge d \rightarrow f \vee c, C_5 = c \rightarrow g\}$. Let $Q_1 = b \vee e \vee g$, $Q_2 = b \vee c \vee f$ and $Q_3 = g \vee f$.*
$IDB = \{C_3 = a \wedge d \rightarrow e \vee b, C_4 = b \wedge d \rightarrow f \vee c, C_5 = c \rightarrow g\}$.
$IDB^d = \{C^d_3 = e \wedge b \rightarrow a \vee d, C^d_4 = f \wedge c \rightarrow b \vee d, C^d_5 = g \rightarrow c\}$.
$IDB^d_{Q_1} = IDB^d \cup Q^d_1 = IDB^d \cup \{b, e, g\}$. $\mathcal{MM}(IDB^d_{Q_1}) = \{\{a, b, c, e, g\}, \{b, c, d, e, g\}\}$. Both $a \vee b \vee c \vee e \vee g$ and $b \vee c \vee d \vee e \vee g$ are subsumed by clauses in EDB. Q_1 is an answer.
$IDB^d_{Q_2} = IDB^d \cup Q^d_2 = IDB^d \cup \{b, c, f\}$. $\mathcal{MM}(IDB^d_{Q_2}) = \{\{b, c, f\}\}$. $b \vee c \vee f$ is not derivable from EDB. Q_2 is not an answer.
$IDB^d_{Q_3} = IDB^d \cup Q^d_3 = IDB^d \cup \{g, f\}$. $\mathcal{MM}(IDB^d_{Q_3}) = \{\{b, c, f, g\}, \{c, d, f, g\}\}$. $c \vee d \vee f \vee g$ is subsumed by $(c \vee d) \in EDB$ while $b \vee c \vee f \vee g$ is not and so Q_3 is not an answer.

The fact that we are characterizing the goal set in terms of the minimal models of a dualized theory allows us to use a (minimal) model generation procedure applied to the dual theory and query to compute that set.

This approach separates the stage of generating the checks from the stage of actual checking. Efficiency gains may be achieved from optimizing access to EDB on external memory. However, the processes of generation and checking can be integrated so that derivability is detected as soon as it occurs, even before the generation of the entire clause. The following theorem shows how to do that.

Theorem 5 *Let DB be a ground DDDB (with no denial constraints) and $Q = q_1 \vee ... \vee q_n$ be a disjunctive query. If $DB^d_Q = \{q_1, ..., q_n\} \cup \{C^d | C \in IDB\} \cup \{Head(C) \rightarrow \bot | C \in EDB\} = IDB^d_Q \cup \{C^d | C \in EDB\} = DB^d_Q$. Then[3]: Q is derivable from DB if and only if $\mathcal{T}^g_{DB} \uparrow \omega(\{Q\}) = \emptyset$; or equivalently iff $\mathcal{MM}(DB^d_Q) = \emptyset$.*

Proof: A clause corresponding to $M \in \mathcal{MM}(IDB^d_Q)$ is derivable from DB if and only if it is subsumed by a clause C in EDB. M will be eliminated by the presence $Head(C) \rightarrow \bot$ in DB^d_Q. Since M is arbitrary, $\mathcal{MM}(DB^d_Q) = \emptyset$. The result follows immediately from Theorem 4. ∎

Note that elements of EDB produce negative clauses of DB^d.

Corollary 2 *Let DB be a ground DDDB (with no denial constraints) and $Q = q_1 \vee ... \vee q_n$ be a disjunctive query. Let $Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q\})) \neq \emptyset$ (That is $\mathcal{MM}(DB^d_Q) \neq \emptyset$). Then:*

1. *Q is not derivable from DB.*
2. *$DB_u \vdash Q$ where $DB_u = DB \cup \mathcal{S}$ and $\forall M \in \mathcal{MM}(DB^d_Q) \exists C \in \mathcal{S}$ s.t. C subsumes M. That is, DB_u is achieved by adding to DB (actually to its EDB component) a set of clauses $\mathcal{S}$ subsuming all the minimal models of DB^d_Q.*
3. *$\mathcal{S} = \mathcal{MM}(DB^d_Q)$ is the weakest such set that can be added to DB to guarantee the derivability of Q from the updated database DB_u.*

Proof:

1. Immediate.
2. Needed to guarantee the condition of theorem 5.
3. Consider $M \in \mathcal{MM}(DB^d_Q)$. Clearly any clause C that is subsumed by M, $(M \subset C)$, will not remove M from the model set when its corresponding denial $(Head(C) \rightarrow \bot)$, is added to DB^d_Q. Any clause that properly subsumes M $(C \subset M)$ can be weakened by augmenting it with the remaining elements of M (elements of $M \setminus C$) and still guarantee the removal of M from the model set.

∎

Once more the characterization of the goal set in terms minimal models makes it possible to use a (minimal) model generation procedure to compute answers in a top down mode.

Example 5 *For the database and queries of Example 4,*
$DB = \{C_1 = a \vee b, C_2 = c \vee d, C_3 = a \wedge d \rightarrow e \vee b, C_4 = b \wedge d \rightarrow f \vee c, C_5 = c \rightarrow g\}$.
Let $Q_1 = b \vee e \vee g$, $Q_2 = b \vee c \vee f$ and $Q_3 = g \vee f$.
$DB^d = \{C^d_1 = a \wedge b \rightarrow \bot, C^d_2 = c \wedge d \rightarrow \bot, C^d_3 = e \wedge b \rightarrow a \vee d, C^d_4 = f \wedge c \rightarrow b \vee d, C^d_5 = g \rightarrow c\}$.
$DB^d_{Q_1} = DB^d \cup Q^d_1 = DB^d \cup \{b, e, g\}$. $\mathcal{MM}(DB^d_{Q_1}) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q_1\})) = \emptyset$. *$Q_1$ is an answer.*
$DB^d_{Q_2} = DB^d \cup Q^d_2 = DB^d \cup \{b, c, f\}$. $\mathcal{MM}(DB^d_{Q_2}) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q_2\})) = \{\{b, c, f\}\}$.
$DB \not\vdash b \vee c \vee f$. *$Q_2$ is not an answer. It will become an answer by adding $b \vee c \vee f$ to EDB.*
$DB^d_{Q_3} = DB^d \cup Q^d_3 = DB^d \cup \{g, f\}$. $\mathcal{MM}(DB^d_{Q_3}) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q_3\})) = \{\{b, c, f, g\}\}$. *$Q_3$ is not an answer but adding $b \vee c \vee f \vee g$ will make Q_3 an answer.*

[3] The extension of Theorem 3 to DB^d is easily established: $\mathcal{MM}(DB^d_Q) \subseteq \mathcal{T}^g_{DB} \uparrow \omega(\{Q\})$ and $\mathcal{MM}(DB^d_Q) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q\}))$.

4 Extentions and Interpretations

4.1 Compound Queries

While we considered only disjunctive queries, the results can be extended to general positive queries. An answer to such a query is affirmative if every clause has a *yes* answer and is negative otherwise.

To answer such a query one could run each clause separately as a disjunctive query and combine the results. Alternatively, one may run a single process with the elements of each clause being the starting set for an initial branch. The resulting minimal set which may be more compact than the union of the minimal sets for individual clauses, will represent the clauses that need to be true in EDB for the compound query to have a *yes* answer. The compactness is the result of exploiting the shared information between the processes corresponding to individual disjunctive components of the query. Consider the following example:

Example 6 *Let* $DB = \{C_1 = a \vee b, C_2 = c \vee d, C_3 = a \wedge d \rightarrow e \vee b, C_4 = b \wedge d \rightarrow f \vee c, C_5 = c \rightarrow g\}$. *Let* $Q_6 = a \vee g \vee f$ *and* $Q_7 = a \vee c \vee f$. $(Q_6 \wedge Q_7)^d = \{a, f, g \vee c\}$.
$\mathcal{MM}(IDB^d_{Q_6}) = \{\{a, b, c, f, g\}, \{a, c, d, f, g\}\}$ *and* $\mathcal{MM}(IDB^d_{Q_7}) = \{\{a, b, c, f\}, \{a, c, d, f\}\}$.
$\mathcal{MM}(IDB^d_{Q_6 \wedge Q_7}) = \mathcal{MM}(IDB^d_{Q_7}) = \{\{a, b, c, f\}, \{a, c, d, f\}\}$.

4.2 Denial Constraints

Denial constraints are rules of the form $C = Body(C) \rightarrow \perp$. In [3] the following result was proved:

Lemma 2 *Let* $\mathcal{S}$ *be a set of clauses and* $A_1, ..., A_n (n \geq 1)$ *be atoms of the Herbrand base of* $\mathcal{S}$.

1. *If* M *is a minimal model of* $\mathcal{S}$ *such that* $M \not\models A_1 \wedge ... \wedge A_n$, *then* M *is a minimal model of* $\mathcal{S} \cup \{A_1 \wedge ... \wedge A_n \rightarrow \perp\}$.
2. *If* M *is a minimal model of* $\mathcal{S} \cup \{A_1 \wedge ... \wedge A_n \rightarrow \perp\}$, *then* M *is also a minimal model of* $\mathcal{S}$.

Clearly, if for a positive query Q, $DB \vdash Q$ then it is also true that $(DB \cup \mathcal{C}) \vdash Q$. However, it is possible that Q is not derivable from DB but is derivable from $DB \cup \mathcal{C}$ if the *bad* models are in the set $(\mathcal{MM}(DB) \setminus \mathcal{MM}(DB \cup \mathcal{C}))$. This was demonstrated by Example 1. So, in a sense, the presence of denial rules must enhance the potential derivability for positive queries. The form this enhancement can take is to expand the clauses in the goal clause set of Q so as to contain more atoms. Indeed, that is what happens. Formally we have the following result:

Theorem 6 *Let* DB *be a ground DDDB,* $\mathcal{C}$ *be the set of denial rules in* DB *and* Q *be a positive query. Then:* $(DB \cup \mathcal{C}) \vdash Q$ *if and only if the formula* $(Body(C) \vee Q) = \{(B \vee Q) | B \in Body(C)\}$ *is derivable from* DB *for some* $C \in \mathcal{C}$.

Proof: If $DB \vdash (Body(C) \vee Q)$ for some $C \in \mathcal{C}$ then the minimal models of DB in which Q is not satisfied will have to satisfy $Body(C)$ and will be nonmodels of $(DB \cup \mathcal{C})$ by an application of C making $(DB \cup \mathcal{C}) \vdash Q$.

Assume $(DB \cup \mathcal{C}) \vdash Q$. By Lemma 2, for any model $M \in \mathcal{MM}(DB)$ s.t. $M \notin \mathcal{MM}(DB \cup \mathcal{C})$ there must exist a clause $C \in \mathcal{C}$ such that $Body(C) \subseteq M$. Clearly, $M \models (Body(C) \vee Q)$. ∎

This theorem is basically suggesting that denial rules expand the goal clause set unconditionally by extending the query Q with atoms from $Body(C)$, one at a time. Since, according to our definitions, denial rules of DB will convert into positive clauses in the dual database DB^d , it is clear that these rules can be treated on the same footing as others in the theory. It is straightforward to extend all the results established so far to the case of databases containing denial rules. The result is summarized in the following theorem[4]:

Theorem 7 *(**general case**) Let DB be a ground DDDB with the set of denial rules $\mathcal{C}$: $DB = EDB \cup IDB \cup \mathcal{C}$ and Q be a positive query.*

- *Let $EDB^d = \{C^d | C \in EDB\}$.*
- *Let $\mathcal{C}^d = \{C^d | C \in \mathcal{C}\}$.*
- *Let $IDB^d = \{C^d | C \in IDB\}$.*
- *Let $IDB^d_Q = Q^d \cup IDB^d = Q^d \cup \{C^d | C \in IDB\}$.*
- *Let $DB^d = EDB^d \cup IDB^d \cup \mathcal{C}^d$.*
- *Let $DB^d_Q = Q^d \cup EDB^d \cup IDB^d \cup \mathcal{C}^d$.*

Then

1. *$DB \vdash Q$ iff EDB derives the clause set $\mathcal{MM}(\mathcal{C}^d \cup IDB^d_Q) = Min(\mathcal{T}^g_{\mathcal{C} \cup IDB} \uparrow \omega(\{Q\}))$.*
2. *$DB \vdash Q$ if and only if $\mathcal{MM}(DB^d_Q) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q\})) = \emptyset$.*
3. *If $\mathcal{MM}(DB^d_Q) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q\}))$ is nonempty then:*
   (a) *Q is not derivable from DB.*
   (b) *Q becomes derivable from the updated database DB_u achieved by adding to DB the set of clauses $\mathcal{S}$ such that $\forall M \in \mathcal{MM}(DB^d_Q) \exists C \in \mathcal{S}$ s.t. C subsumes M.*
   (c) *$\mathcal{S} = \mathcal{MM}(DB^d_Q) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q\}))$ is the weakest such set that can be added to DB to guarantee the derivability of Q from the updated database.*

Proof: Along the lines of earlier proofs (omitted for space considerations). ∎

Example 7 *Let $DB = \{C_1 = a \vee b, C_2 = c \vee d, C_3 = a \wedge d \rightarrow e \vee b, C_4 = b \wedge d \rightarrow f \vee c, C_5 = c \rightarrow g, C_6 = a \wedge d \rightarrow \bot\}$. Let $Q_1 = b \vee e \vee g$, $Q_2 = b \vee c \vee f$ and $Q_3 = g \vee f$. $\mathcal{C} = \{C_6 = a \wedge d \rightarrow \bot\}$.*
$IDB^d \cup \mathcal{C}^d = \{C^d_3 = e \wedge b \rightarrow a \vee d, C^d_4 = f \wedge c \rightarrow b \vee d, C^d_5 = g \rightarrow c, C^d_6 = \top \rightarrow a \vee d\}$.
$IDB^d_{Q_1} \cup \mathcal{C}^d = \{C^d_3 = e \wedge b \rightarrow a \vee d, C^d_4 = f \wedge c \rightarrow b \vee d, C^d_5 = g \rightarrow c, C^d_6 = \top \rightarrow a \vee d, b, e, g\}$
$IDB^d_{Q_2} \cup \mathcal{C}^d = \{C^d_3 = e \wedge b \rightarrow a \vee d, C^d_4 = f \wedge c \rightarrow b \vee d, C^d_5 = g \rightarrow c, C^d_6 = \top \rightarrow a \vee d, b, c, f\}$.
$IDB^d_{Q_3} \cup \mathcal{C}^d = \{C^d_3 = e \wedge b \rightarrow a \vee d, C^d_4 = f \wedge c \rightarrow b \vee d, C^d_5 = g \rightarrow c, C^d_6 = \top \rightarrow a \vee d, g, f\}$.
$\mathcal{MM}(IDB^d_{Q_1} \cup \mathcal{C}^d) = \{\{a, b, c, e, g\}, \{b, c, d, e, g\}\}$. Both $a \vee b \vee c \vee e \vee g$ and $b \vee c \vee d \vee e \vee g$ are subsumed by clauses in EDB. Q_1 is an answer. It is not affected by the added constraint C_6.
$\mathcal{MM}(IDB^d_{Q_2} \cup \mathcal{C}^d) = \{\{a, b, c, f\}, \{b, c, f, d\}\}$. Both clauses are derivable from EDB and therefore

[4] The result is quite natural. The empty head $\bot$ of the denial clause trivially subsumes every positive clause and therefore the goal set of the query can be expanded using such a clause, unconditionally. As expected, in the transformed theory such clauses are converted into facts where they can be used for this unconditional expansion.

Q_2 is an answer as a result of adding C_6.
$\mathcal{MM}(IDB^d_{Q_3} \cup \mathcal{C}^d) = \{\{a,b,c,f,g\},\{c,d,f,g\}\}$. $c \vee d \vee f \vee g$ is subsumed by $(c \vee d) \in EDB$ and $a \vee b \vee c \vee f \vee g$ is subsumed by $(a \vee b) \in EDB$ and so Q_3 is an answer as a result of adding C_6.
Note that $\mathcal{MM}(DB^d_{Q_i}) = Min(\mathcal{T}^g_{DB} \uparrow \omega(\{Q_i\})) = \emptyset$, for $i \in \{1,2,3\}$. Comparing the results with Example 5 demonstrates that adding the constraints contributed to deriving more yes answers.

4.3 Nonground Rules

While we described our procedure for ground DDDBs, lifting them to the case of nonground databases is possible. The problem is if the bottom-up procedure can find all the required clauses of the theory (models of the dual) starting from the set of facts representing the dual of the query. A point to note is that variables in the heads of dual clauses are treated as existentially quantified and multiple copies can be used for a refutation reflecting the indefiniteness of the answer. The detailed treatment of this issue is omitted for space considerations.

Example 8 *[9] Let $DB = \{C_1 = Person(x) \wedge Cold(x) \rightarrow Sneeze(x), C_2 = Person(x) \wedge HayFever(x) \rightarrow Sneeze(x), C_3 = \top \rightarrow Person(Tom), C_4 = Person(x) \wedge Cold(x) \wedge HayFever(x) \rightarrow \bot\}$.*
Let $Q = Sneeze(Tom)$. $Q^d = Sneeze(Tom)$.
$DB^d = \{C^d_1 = Sneeze(x) \rightarrow Person(x) \vee Cold(x), C^d_2 = Sneeze(x) \rightarrow Person(x) \vee HayFever(x), C^d_4 = \top \rightarrow Person(x) \vee Cold(x) \vee HayFever(x), C^d_3 = Person(Tom) \rightarrow \bot\}$.
The only minimal model of DB^d_Q is $\{Sneeze(Tom), Cold(Tom), HayFever(Tom)\}$. Q is not a yes answer and $Sneeze(Tom) \vee Cold(Tom) \vee HayFever(Tom)$ is the possible update. $\{Cold(Tom) \vee HayFever(Tom)\}$ is the nontrivial update (explanation).

4.4 Interpretations of the Duality Approach

Usually, proving the query is done by trying to refute the theory augmented by the negation of the query [11]. When all clauses are represented as disjunctions of literals, the dual transformation has the effect of consistently reversing the polarity of each literal of both the theory and the negation of the query. This is so since $C = Head(C) \vee \neg Body(C)$ while $C^d = \neg Head(C) \vee Body(C)$. Clearly, this syntactic transformation preserves the consistency properties. So[5] $DB \cup \{Neg(Q)\} \vdash \Box$ if and only if $DB^d \cup \{Q^d\} \vdash \Box$. The change in efficiency can be attributed to the fact that the bottom-up computational procedures treat positive and negative literals asymmetrically. For example, model generation provers are generally driven by positive facts that are then used to generate new facts through theory clauses. Negative clauses are only used to close branches when applicable [3]. Therefore, working with the transformed theory DB^d can affect the performance of the algorithm by reducing the number of positive literal occurrences and thus limiting the number of possible expansions. Under favorable circumstances the overall effect may be faster refutations .

Another view of the dual transformation is to interpret it as using the rules to (backward) propagate the *falsity* of the head atoms to the body atoms of each clause, initiated by the query. I.e. to specify the sets of atoms that need to be *false* in order for the query to be *false*. This is in contrast to the (forward) propagation of the *truth* of the body atoms to the head atoms when the bottom-up procedure is applied to the original clauses, initiated by elements of EDB.

Example 9 *Let $DB = \{C_1 = b \vee c, C_2 = b \rightarrow a \vee e, C_3 = c \rightarrow a \vee d, C_4 = b \wedge e \rightarrow \bot\}$. Let $Q = a \vee d$.*
$DB^d = \{C^d_1 = b \wedge c \rightarrow \bot, C^d_2 = a \wedge e \rightarrow b, C^d_3 = a \wedge d \rightarrow c, C^d_4 = \top \rightarrow b \vee e\}$.

[5] Recall footnote 2 on page 6 detailing the relationship between , $Neg(Q)$ and Q^d.

The clausal representation of $DB \cup \{Q \rightarrow \bot\}$ *and* $DB^d \cup Q^d$ *are*
$\{C_1 = b \vee c, C_2 = \neg b \vee a \vee e, C_3 = \neg c \vee a \vee d, C_4 = \neg b \vee \neg e\} \cup \{\neg a, \neg d\}$ *and*
$\{C_1^d = \neg b \vee \neg c, C_2^d = b \vee \neg a \vee \neg e, C_3^d = c \vee \neg a \vee \neg d, C_4^d = b \vee e\} \cup \{a, d\}$, *respectively.*
The correspondence between the two sets and the reversal of polarities is clear.

Note that any resolution of two clauses in DB *can be simulated by a resolution of the corresponding clauses in* DB^d *with the resulting resolvents having reversed polarities of their literals. E.g.* $Resolvent(C_1, C_2) = a \vee c \vee e$ *and* $Resolvent(C_1^d, C_2^d) = \neg a \vee \neg c \vee \neg e$.

Elements of $\mathcal{MM}(DB) = \{\{a, b\}, \{a, c\}, \{c, d\}\}$ *are sets of atoms that need to be* true *simultaneously to satisfy the theory (driven by EDB). Elements of* $\mathcal{MM}(IDB_Q^d) = \{\{a, b, c, d\}\}$ *are sets of atoms that need to be* false *simultaneously to satisfy IDB and the falsity of* Q *(driven by* Q^d *and* $\mathcal{C}^d$*).*

5 Conclusions and Future Work

We presented a simple approach to enable the use of a forward chaining procedure to process queries in a backward chaining mode. The idea is to utilize a certain version of the *duality* principle to reinterpret the clauses of the input theory so that the application of a bottom-up procedure to the transformed theory will answer the posed positive query in a top-down fashion.

From a theoretical perspective, the results constitute an elaboration on the strong connection between concepts used to characterize disjunctive theories: derivable clauses and models; minimal model set and minimal model state; model trees and clausal trees; and query answering and model generation [21, 18, 22]. We emphasized the strong connection between the set of clauses that need to be subsumed by EDB to make the query derivable and the set of minimal models of the theory dualization, with the view on using (minimal) model generating procedures for bi-directional query processing. From a practical point of view, our algorithm can result in substantial savings due to the limited search space explored and can benefit from the wealth of work and algorithms, available and under development, for efficient (minimal) model generation to perform a goal focused search for answers [3, 2, 17]. Our preliminary testing points to substantial performance improvement achievable by using a minimal model generation based query answering procedure on the dual theory to achieve top-down processing, as opposed to having the same procedure operate on the input theory in a bottom-up mode[6].

In contrast to other approaches, ours is applicable to disjunctive theories [13] and avoids the explicit introduction of new predicates into the transformed theory [1, 8, 5, 16, 19]. Rather we achieve the required results by reinterpreting the clauses (and consequently the logical connectives) in a dual mode. As is the original, the transformed theory is a DDDB. The dual transformation is quite simple and involves only changing the direction of implications in all clauses. Including (negative) clauses corresponding to the elements of the extensional database, EDB, in the process makes it work in a refutation-like manner and may improve the performance by detecting clause subsumption as early as possible. The reasoning behind the duality approach was shown to be natural and based on solid logical grounds. It is equivalent to working with reversed polarities of literals and using clauses to propagate particular truth values.

The method in [10] uses special data structures (deduction trees) and algorithms to achieve top-down query answering without transforming the theory. That approach doesn't offer the refined

[6] Our testing was performed on a prototype implementation of the minimal model generator MM-Satchmo as described in [3]. The theory and the query were presented in both original form and manually transformed dual form. The gains were achieved when both ground and range-restricted (in both directions) theories were used.

query answering capabilities of the method outlined here. We specify the minimal component that needs to be satisfied to have an affirmative answer and thus produce minimal updates.

[14] outlines a method based on using SLO-resolution to modify the WAM approach so that it deals with disjunctive logic programs. The modification, Disjunctive WAM (DWAM), uses clause subsumption as the basic expansion mechanism and operates in a goal oriented fashion for query answering. In contrast with the current method, the DWAM and the deduction tree approach of [10], are applied to theories without constraints.

A more substantive difference is that our approach avoids much of the nondeterminism that causes problems for the SLO-based DWAM approach. Rather than searching for alternative ways to subsume a goal we try all possible subsumptions but without repetition. Additionally, our definition of the goal set explicitly excludes a major class of irrelevant clause expansions that can cause the search space to explode. Matching clauses are expanded only if they can contribute something new to the refutational process. No clause is used for expansion more than once in a single branch and we never expand using a clause with body atoms intersecting with the set of atoms in the current branch. This makes it possible to avoid many useless expansions.

While we still have to choose from among clauses with matching heads, our expansion of the goal tree is deterministic: e.g. a selection function always selects the leftmost goal from the first (in the given clause order) potentially useful clause and we insist on solving it (or failing) before moving (backtracking) to the next goal in that clause. In a sense this makes our search more focused: at every stage we are concerned with the solvability of a particular goal clause. Subsequent goal clauses are considered, if required, only after the decision on the current clause is made (success or failure).

The fact that our approach is based on a different clause expansion paradigm makes it possible to avoid the extensive rewriting (I-code) needed for the DWAM approach to account for the nondeterminism of subsumption checking and the indexing needed to keep track of clause usage. Rather, we use the syntactic duality transformation and that alone makes it possible to utilize already existing bottom-up procedures, with their efficiency enhancing techniques, to process queries top-down.

An added advantage of our approach is that it is able to specify the conditions under which the goal set expansion can be discontinued and the minimal updates [7] needed to derive the query. The latter is a refinement of the query answering process. However, our updates take the form of clause additions and do not include specifying clauses that need to be false to guarantee the derivability of the query [17]. The set of goal clauses returned can be used to modify the clausal structure of EDB to make the query nonderivable. For space considerations we didn't elaborate on this and related issues. It is also possible for the user to divide the query answering process into two stages: generating a complete (sufficient) and necessary set of clauses that need to be checked for derivability in the extensional component of the database and the actual checking process. This can be employed to achieve optimal access time to the EDB when it is stored in slower memory and makes it possible to localize updates to individual components of the database. Alternatively, interleaving accesses to the different components of the database will make it possible to operate the procedure in the refutation mode, where the aim is to derive the empty clause. In this case the procedure will operate on a (possibly much) larger theory that includes the transformed EDB but will tend to end the expansion sooner. The choice of the mode will depend on the relative sizes of the database components and their relative access times.

The debate over which direction for clause evaluation: bottom-up or top-down performs best was addressed extensively in the literature [4, 13, 15, 19, 20]. Our presentation is not meant to solve this issue but rather to offer the user a choice. As a matter of fact, since $(DB^d)^d = DB$ it is immediate to note that for any theory that performs better for one approach there is a theory that performs worse. A strong argument for the top-down approach is that many deductive databases fall into the

class where it is likely to perform better [19]. Model generation procedures are good candidates to be used for our approach [3, 2]. Our method will be able to utilize developments in this area to improve performance and cover a wider class of theories than considered here.

The duality approach as outlined here suggests using one direction (top-down or bottom-up) for processing all clauses of the theory. It is of interest to combine both directions for clause evaluation in a single query answering run to try to achieve *optimal* performance. Other topics of further study are the ability of the duality approach to handle query answering under various database semantics, e.g. in the presence of negation as failure in clause bodies and multiple types of negation and the use of the defined approach in database update, abduction and answering nonpositive queries.

References

[1] F. Bancilhon, Y. Sagiv, and J. Ullman. Magic sets and other strange ways to implement logic programs. In *Proceedings of the Fifth ACM SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–15, 1988.

[2] P. Baumgatrner, U. Furbach, and I. Niemmelä. Hyper tabluaux. *Proceedings of JELIA96*. LNAI Series 1126. Springer Verlag. 1996.

[3] F. Bry and A. Yahya. Minimal model generation with positive unit hyper-resolution tableaux. *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.

[4] F. Bry. Query evaluation in recursive databases: bottom-up and top-down reconciled. *Data & Knowledge Engineering*, pages 289–312, 1990.

[5] R. Demolombe. An efficient strategy for non-horn deductive data bases. *Theoretical Computer Science*, 78:245–259, 1991.

[6] J. A. Fernández and J. Minker. Bottom-up computation of perfect models for disjunctive stratified theories. *Journal of Logic Programming*, 25(1):33–50, 1995.

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

[8] R. Hasegawa, K. Inoue, Y. Ohta and M. Koshimura. Nonhorn magic sets to incorporate top-down inference into bttom-up theorem proving. *Proceedings of CADE97* PP. 1997.

[9] K. Inoue and C. Sakama. A fixpoint characterization of abductive logic programs. *Journal of Logic Programming*, 27:107–136, 1996.

[10] C.A. Johnson. Top down deduction in indifinite deductive databases. In *Journees Bases de Donnees Avancees*, pages 119–138, Toulosuse, France, 1993.

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

[12] D.W. Loveland, D. Reed, and D. Wilson. Satchmore: Satchmo with relevancy. *J. Automated Reasoning*, 14:349–363, July 1995.

[13] V. Neiman. Refutation search for horn sets by a subgoal-extraction method. *Journal of Logic Programming*, 9:267–284, 1990.

[14] A. Rajasekar and H. Yusuf. Dwam - a wam model extension for disjunctive logic programming. *Annals of Mathematics and Artifitial Intelligence*, 14:275–308, 1995.

[15] R. Ramakrishnan and S. Sudarshan. Top-down vs. bottom-up revisited. In *Proceedings of the ISLP'91*, 1991.

[16] J. Rohmer, R. Lescoeur, and J-M. Kerisit. The alexander method: a technique for the processing of recusive axioms in deductive databases. *New Generation Computing*, 4(3), 1986.

[17] V. Royer. Backward chaining Evaluation in stratified disjunctive theories. *Proceedings of PODS'90*, PP. 183–195, 1990.

[18] D. Seipel, J. Minker, and C. Ruiz. Model generation and state generation for disjunctive logic programs. *Journal of Logic Programming* 32(1), 1997. PP. 48–69.

[19] M. Stickel. Meta interpretation of the model elimination theorem proving procedure for deduction and abduction. *J. Automated Reasoning*, 13(2):189–210, October 1994.

[20] R. Treitel and M. Genesereth. Choosing directions of rules. *J. Automated Reasoning*, 3(2):395–431, October 1987.

[21] 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.

[22] A. Yahya. A goal-driven approach to efficient query processing in disjunctive deductive databases. Technical Report PMS-FB-1996-12. Department of Computer Science, Munich University. July 1996.