View Updates in Disjunctive Deductive Databases Based on SLD-Resolution*

Wenjin Lu
Department of Computer Science, University of Koblenz-Landau
Rheinau 1, D-56075 Koblenz, Germany
lue@uni-koblenz.de

Abstract

This paper discusses view updates in disjunctive deductive databases. In contrast to existing solutions in [2, 5, 3] where either the syntax of discussed databases is restricted [2] or a complicated calculus is employed [5, 3], we show, in this paper, how the problem can be solved in a framework based on SLD-resolution.

1 Introduction

A disjunctive deductive database (DDDB) is a collection of data separated into two parts. One part is called the extensional database (EDDB): it represents the set of (possibly indefinite) facts know about the world. It is a finite set of ground positive disjunctions know to be true. The second part is the intensional database (IDDB): it consists of a finite set of disjunctive program clauses that represent general rules or conditions from which new facts can be deduced.

A view of a relation R in a DDDB consists of the set of rules that define R. The definition of R is called a view since views in traditional relational databases correspond to rules in DDDBs. All predicates that are defined in IDDB are referred to as view predicates and those defined in EDDB are referred to as base predicates. Extending this notion, an atom with a view predicate is called to be a view atom, and similarly an atom with base predicate is a base atom. Further we assume that no predicate defined in a given DDDB is both view and base. For the sake of simplicity we also assume that there is no integrity constraints associated with the given database. In addition, our discussion is restricted to ground DDDBs.

*__Acknowlegement:__ The Author thanks anonymous referees for their helpful suggestions.

Example 1 *Below is a disjunctive deductive database.*

$$
\begin{array}{lrl}
IDDB: & p \vee q & \leftarrow r, e \\
 & p \vee q & \leftarrow r, f \\
 & r & \leftarrow a \\
 & r & \leftarrow b \\
 & p & \leftarrow q \\
 & q & \leftarrow t
\end{array}
\qquad
\begin{array}{ll}
EDDB: & a \vee b \vee e \\
 & e \vee f \vee a
\end{array}
$$

where the view atoms are p, q, r *and base atoms are* a, b, e, f.

The view update problem is concerned with the translation of an update of an intensional relation by modifying the underlying relations of the extensional part of the database. Two kinds of view updates can be carried out on a DDDB: An atom, that does not currently follow from DDDB, can be *inserted*; or, an atom, that currently follows from DDDB, can be *deleted*.

To update a view in a deductive database, a basic calculus is usually employed to generate explanations for the atom to be inserted or deleted [1]. In the case of definite datalog databases, algorithms based on SLD-tree have been proposed. View updates for DDDB have been discussed in [5, 2, 3]. In [2], to use SLD-resolution, the syntax of a DDDB is restricted such that its intensional database is a Horn program. Solutions for general DDDB have been discussed in [5, 3], however, SLD-resolution is replaced by more complicated SLI-resolution or deductive tree. In this paper we do not consider abstract philosophical problems concerning view updates, such as rationality of change [1]. Our emphasis is put on how the problem of view updates in a DDDB can be solved in a framework based on SLD-resolution.

The paper is organized as follows: After introducing CH-Prolog, we provide a algorithm basis for view updates of DDDBs in Section 3. Algorithms for view insertion and view deletion are presented in Section 4. We conclude this paper in Section 5 with a discussion on further work.

2 CH-Prolog

CH-Prolog, first introduced in [7, 8], is a first order proof procedure for disjunctive logic programming. It is based on the CH-transformation on disjunctive logic programs [4, 7]. For the purpose of this paper, we present here only its propositional version.

Definition 2 *Let P be a disjunctive logic program. For each clause $C \in P$ of the form:*

$$a_1 \vee ... \vee a_n \leftarrow b_1, ..., b_m$$

The CH-transformation of C, denoted by $CH(C)$, is a set of clauses defined by:

$$CH(C) = \begin{cases} \{C\}: & C \text{ is a Horn clause} \\ C_{ch}: & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} C_{ch} \quad = \{ \quad & C_1 \vee C_2 \vee ... \vee C_n, \\ & a_1 \leftarrow b_1, ..., b_m, C_1, \\ & ...... \\ & a_n \leftarrow b_1, ..., b_m, C_n \ \} \end{aligned}$$

C_1, ..., C_n are new propositional atoms not in P, they are called control atoms, the disjunctive fact $C_1 \vee ... \vee C_n$ is called a control clause. The CH-transformation of P, denoted by $CH(P)$, is defined by

$$CH(P) = \bigcup_{C \in P} CH(C)$$

The set of all control atoms in $CH(P)$ is denoted by $\mathcal{C}_\mathcal{P}$

By definition, the clause in $CH(P)$ is either a Horn clause or a disjunctive fact (control clause). Therefore $CH(P)$ can be represented as follows:

$$CH(P) = P_H + P_C$$

where P_C is the program consisting of all control clauses in $CH(P)$ and P_H is the Horn program consisting of all Horn clauses in $CH(P)$. P_C is called the control program of P_H.

Example 3 *Note the fact that the IDDB in Example 1 is a disjunctive logic program. Applying CH-transformation to the IDDB produces a Horn program $IDDB_H$ and a control program $IDDB_C$ as follows:*

$$\begin{array}{lll} IDDB_H: & p & \leftarrow r, e, C_1 \\ & q & \leftarrow r, e, C_2 \\ & p & \leftarrow r, f, C_3 \\ & q & \leftarrow r, f, C_4 \\ & r & \leftarrow a \\ & r & \leftarrow b \\ & p & \leftarrow q \\ & q & \leftarrow t \end{array} \qquad \begin{array}{ll} IDDB_C: & C_1 \vee C_2 \\ & C_3 \vee C_4 \end{array}$$

Where C_1, C_2, C_3, and C_4 are new control atoms.

Next we turn to the proof procedure *CH-Prolog* for disjunctive logic programs. The prefix "CH", on one hand, hints that the procedure is based on the CH-transformation, and on the other hand it also indicates that CH-Prolog is a Prolog with a "CHecking". Following we assume that reader knows the basic concepts from SLD-resolution which can be found in [6].

Definition 4 (Restricted SLD-tree) *Let P be a disjunctive program and $CH(P) = P_H + P_C$ be the CH-transformation of P. Let G be a goal. Assume R to be a computation rule that only selects non control atoms. A restricted SLD-tree for $P \cup \{G\}$ is the SLD-tree for $P_H \cup \{G\}$ via R. A restricted success branch in a restricted SLD-tree is the branch ending either with an empty clause $\square$ or with a subgoal consisting of only control atoms.*

Given a disjunctive program P, let $CH(P) = P_H + P_C$ be the CH-transformation of P. The calculus then is defined by simulating SLD-tree building for $P_H \cup \{\leftarrow Q\}$. When restricted to propositional case, the basic derivation in CH-Prolog can be stated as follows:

Definition 5 (SLDCH-derivation) *Let P be a disjunctive program and $CH(P)=P_H + P_C$. Let G be a goal. Assume R to be a computation rule that only selects non control atoms. An SLDCH-derivation is a sequence $L_0, L_1, \ldots, L_n \ldots$, where*

1. *L_0 is the given goal G and*
2. *If $L_i = G_i \# C_1 \# \ldots \# C_k$ and G_i contains a non control atom, then*

   $$L_{i+1} = G_{i+1} \# C_1 \# \ldots \# C_k,$$

   where G_{i+1} is derived from G_i and some clause $C_{i+1} \in P_H$ via R.
3. *If $L_i = G_i \# C_1 \# \ldots \# C_k$ and G_i consists of only control atoms, then*

   Checking: *if $(C_1 \vee \ldots \vee C_k \vee C_{k+1})$ is a logic consequence of P_C, where $C_{k+1} = \neg G_i$, then $L_{i+1} = \square$. Otherwise*

   Restart: $L_{i+1} = G \# C_1 \# \ldots \# C_k \# C_{k+1}$.

An SLDCH-derivation is called an SLDCH-refutation if it contains a empty clause $\square$.

Roughly speaking, except the checking, SLDCH-derivation works exactly same as SLD-derivation. Therefore in CH-Prolog, high inner-loop speed can be expected. CH-Prolog is a sound and complete proof procedure for disjunctive logic programming [7, 8].

Theorem 6 *Let P be a disjunctive logic program and Q is a query. Then $P \models Q$ iff there is a SLDCH-refutation for P and goal $\leftarrow Q$.*

3 Disjunctive Explanations

Given a DDDB and view atom, we show, in this section, how to compute an explanation for the view atom.

Recall that for a given disjunctive logic program P, to prove an atom q, CH-Prolog works as follows:

1. transform P into $P_H + P_C$,
2. construct an SLD-tree for $P_H \cup \{\leftarrow q\}$,
3. let $\leftarrow C_1, \ldots, \leftarrow C_m$ be all leaf nodes consisting of only control atoms, then q is a logic consequence of P iff $C_1 \vee \ldots \vee C_m$ is a logic consequence of P_C.

Note that C_i $(1 \leq i \leq m)$ is a conjunction of control atoms. $C_1 \vee \ldots \vee C_m$ is a disjunctive normal form and P_C is a conjunctive normal form. Let $D_1 \wedge \ldots \wedge D_{m'}$ be a conjunctive normal form of $C_1 \vee \ldots \vee C_m$, then $C_1 \vee \ldots \vee C_m$ is a logic consequence of P_C iff each D_i $(1 \leq i \leq m')$ is subsumed by some control clause.

Let $DB = EDDB \cup IDDB$ be a DDDB. By definition $EDDB$ consists of only disjunctive facts, the extensional atoms never occur in the head of rules in $IDDB_{DB}$, therefore the $EDDB$ in DB plays a role like a control program. To address the database update, we apply CH-transformation to the intensional database $IDDB$. Let

$$CH(IDDB) = IDDB_H + IDDB_C$$

and define

$$DB_{CH} = EDDB_C \cup IDDB_H$$

where $EDDB_C = EDDB \cup IDDB_C$, then DB_{CH} is a new DDDB with a definite intensional database $IDDB_H$. The following theorem follows from results in [7, 8].

Theorem 7 *Let $DB = IDDB \cup EDDB$ be a DDDB, and $DB_{CH} = IDDB_H \cup EDDB_C$ be defined as above. Then for any view atom q, $DB \models q$ iff $DB_{CH} \models q$.*

Logically, DB and DB_{CH} are equivalent, but when considering view updates, DB and DB_{CH} are very different. That is, a view update request on DB is not equivalent to the same view update request on DB_{CH}. This is because after transformation, the set of base atoms has been enlarged. Next we discuss how a view update request on DB can be addressed by updating DB_{CH}. The following definition extends the notion of *explanation* from the definite case to disjunctive case.

Definition 8 (Disjunctive Explanation) *Let $DB = EDDB \cup IDDB$ be a DDDB and A be a view atom. Then a disjunctive explanation of A with respect to DB is a set $\mathcal{E}$ of disjunctive facts that consist of extensional atoms such that $IDDB \cup \mathcal{E} \models \mathcal{A}$. Given two explanations $\mathcal{E}$ and $\mathcal{E}'$, we say $\mathcal{E}$ is weaker than $\mathcal{E}'$ iff $\mathcal{E}' \models \mathcal{E}$ but $\mathcal{E} \not\models \mathcal{E}'$. A disjunctive explanation $\mathcal{E}$ is called a weakest explanation if there exist no explanations that is weaker than $\mathcal{E}$.*

By definition, weakest explanations are unique up to logical equivalence. As the careful reader has already realized, the weakest disjunctive explanations will play a critical role in view updates of DDDB, however, this explanation does not always exist. The following theorem shows that if a view atoms has an explanation with respect to DB, then it has a weakest explanation with respect to DB.

Theorem 9 *Let $DB = EDDB \cup IDDB$ be a disjunctive database, and $DB_{CH} = EDDB_C \cup IDDB_H$ be defined as above. Let A be a view atom and let T be an SLD-tree[1] of $IDDB_H \cup \{\leftarrow A\}$. Suppose that $\leftarrow C_1, \ldots, \leftarrow C_l$ are all leaf nodes which consist of only control or extensional atoms. Let $D_1 \wedge \ldots \wedge D_m$ be the conjunctive normal form of $C_1 \vee \ldots \vee C_l$ such that no clause D_i is subsumed by D_j $(j \neq i)$ or a control clause in $IDDB_C$. Then*

1. *if there is a clause D_j consisting of only control atoms, then A has no disjunctive explanation with respect to DB. Otherwise*
2. *$\{E_1, \ldots, E_m\}$ is a weakest disjunctive explanation of A with respect to DB, where E_i is obtained by deleting all control atoms in D_i (if any).*

This theorem, in fact, provides a means of computing a weakest explanation. We summarize it in Algorithm 1.

Algorithm 1 : Compute a weakest explanation

Input: A database $DB = EDDB \cup IDDB$ and a given view atom A.
Output: A weakest explanation $\mathcal{E}$ of A with respect to DB whenever it exists.
begin

1. Transform DB into $DB_{CH} = EDDB_C \cup IDDB_H$.
2. construct a complete restricted SLD-tree for $\leftarrow A$ and $IDDB_H$.
3. Let $\leftarrow C_1, \ldots, \leftarrow C_m$ be the leaf nodes containing only conjunctions of extensional atoms or control atoms. Construct a conjunctive normal form of $C_1 \vee \ldots \vee C_n$, $D_1 \wedge \ldots \wedge D_n$, such that no clause D_i is subsumed by a control clause.
4. If there is D_i which consists of only control atoms and is not subsumed by any control clause, then return "No explanation". Otherwise
5. Return $\mathcal{E}= \{E_{i_1}, \ldots, E_{i_l}\}$, where E_{i_j} $(1 \leq j \leq l)$ is obtained by removing all control atoms from D_{i_j}, and is not subsumed by E_{i_k} $(k \neq j)$.

end

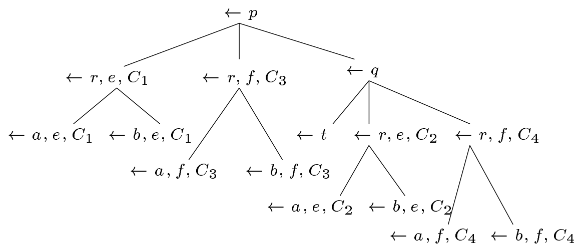
---

[1] We always assume that the computation rule selects only intensional atoms.

Example 10 *Applying CH-transformation to IDDB in Example 1 produces DB_{CH} in which its intensional part is just the $IDDB_H$ in Example 3 and its extensional part is just the union of EDDB and the control program in Example 3, that is,*

$$EDDB_C = IDDB_C \cup EDDB$$

An SLD-tree for $IDDB_H$ and $\leftarrow p$ is depicted below.



From the SLD-tree we get the subgoals

$$\leftarrow a, e, C_1; \quad \leftarrow b, e, C_1; \quad \leftarrow a, f, C_3; \quad \leftarrow b, f, C_3;$$
$$\leftarrow a, e, C_2; \quad \leftarrow b, e, C_2; \quad \leftarrow a, f, C_4; \quad \leftarrow b, f, C_4$$

which consist of only extensional atoms and control atoms. Note that subgoal $\leftarrow t$ is not included because it contains a intensional atom t. Translate the following disjunctive normal form:

$$(a \wedge e \wedge C_1) \vee (b \wedge e \wedge C_1) \vee (a \wedge f \wedge C_3) \vee (b \wedge f \wedge C_3) \vee$$
$$(a \wedge e \wedge C_2) \vee (b \wedge e \wedge C_2) \vee (a \wedge f \wedge C_4) \vee (b \wedge f \wedge C_4)$$

into a conjunctive normal form

$$(a \vee b) \wedge (e \vee f) \wedge (e \vee C_3 \vee C_4) \wedge (f \vee C_1 \vee C_2) \wedge (C_1 \vee C_2 \vee C_3 \vee C_4).$$

Delete all clauses which are subsumed by a control clause, finally, we get a weakest explanation $\mathcal{E}=\{a \vee b,\ e \vee f\}$ of the view atom p.

This algorithm provides a basis for the algorithms presented in the next two sections.

4 View Updates

In this section we present algorithms for view updates in DDDB. Given a DDDB $DB = IDDB \cup EDDB$. Let $DB_{CH} = EDDB_C \cup IDDB_H$ be defined as above, then DB_{CH} is a special kind of disjunctive databases. The view update problem of databases in this case has been discussed in [2]. Therefore it is not surprising that the algorithms presented below are obtained by modifying the related algorithms in [2] and their correctness can also be proved similarly.

We first discuss view insertion. The following formal definition of view insertion is taken from [2, 5].

Definition 11 *Let DB be a DDDB and A be a view atom such that $DB \not\models A$.*

- *An insertion of the atom A into DB is a disjunctive database DB' such that $DB' \models A$.*
- *DB' is called a minimal insertion if there is no database DB'' which is a insertion of A such that $DB \subseteq DB'' \subset DB'$.*
- *DB' is called a weakest insertion if there is no database DB'' which is an insertion of A such that $DB' \models DB''$ and $DB' \neq DB''$* .

Algorithm 2 addresses view insertion.

Algorithm 2 : View insertion algorithm

Input: A database $DB = EDDB \cup IDDB$ and a given view atom A to be inserted. $DB \not\models A$.
Output: A database $DB' = EDDB' \cup IDDB$ such that $DB' \models A$ whenever the insertion is possible.
begin

1. Call Algorithm 1 for input DB and A.
2. If Algorithm 1 outputs "no explanation", then output "insertion impossible". Otherwise let $\mathcal{E}$ be the output.
3. Delete all subsumed clauses from $\mathcal{E} \cup EDDB$, let $EDDB'$ be the resulting subset of $\mathcal{E} \cup EDDB$.
4. Output $DB' = EDDB' \cup IDDB$.

end

Example 12 *Let DB be the DDDB in Example 1. Consider insertion of view atom p. By Example 1, a weakest explanation of p is $\mathcal{E}= \{a \vee b, e \vee f\}$. Hence DB', the database produced by Algorithm 2 after the insertion of p is as follows:*

$$\begin{array}{llll} IDDB: & p \vee q & \leftarrow r, e & \qquad EDDB': \quad a \vee b \\ & p \vee q & \leftarrow r, f & \qquad \qquad \qquad \quad\; e \vee f \\ & r & \leftarrow a & \\ & r & \leftarrow b & \\ & p & \leftarrow q & \end{array}$$

Next we turn to view deletion problem. The following formal definition of view deletion is taken from [2, 5].

Definition 13 *Let DB be a disjunctive database and A be a view atom such that $DB \models A$. Then*

1. *A deletion of the atom A from DB is a disjunctive database DB' such that $DB' \not\models A$.*
2. *A deletion DB' of A from DB is called a minimal deletion if DB' is a maximal subset of DB that does not imply A. That is, $DB' \subseteq DB$ and $DB' \not\models A$ and for any other disjunctive database $D'' \subseteq DB$ if $DB' \subset DB''$, $DB'' \models A$.*

Algorithm 3 addresses view deletion.

Algorithm 3 : View deletion algorithm

Input: A database $DB = EDDB \cup IDDB$ and a given view atom A to be deleted.

Output: A database $DB' = EDDB' \cup IDDB$ such that $DB' \not\models A$.

begin

1. Compute the weakest explanation $\mathcal{E}'$ using Algorithm 1.
2. Delete all subsumed clauses from $\mathcal{E}'$, let $\mathcal{E}= \{E_1, \ldots, Em\}$ be the resulting subset of $\mathcal{E}'$.
3. Let S_i be the sets of clauses in DB which subsume some $E_i \in \mathcal{E}$, $1 \leq i \leq m$.
4. For each j, $1 \leq j \leq m$ such that there is no i, $i \neq j$ and $1 \leq i \leq m$, $S_i \subset S_j$ obtain $DB_j = (DB \setminus S_j)$. Without loss of generality assume that the DB_j's are such that $1 \leq j \leq k$ for some k, $k \leq m$.
5. Let DB'' be $\bigvee_{j=1}^{k} DB_j$.
6. Delete any redundant clause (clause subsumed by another clause) in DB'', let DB' be the resulted disjunctive database.

end

Example 14 *Let DB be the following DDDB, $DB \models p$.*

$$
\begin{array}{llll}
IDDB: & p \vee q \leftarrow r, e & EDDB: & a \vee b \\
 & p \vee q \leftarrow r, f & & e \vee f \\
 & r \leftarrow a & & a \vee f \\
 & r \leftarrow b & & b \vee e \\
 & p \leftarrow q & &
\end{array}
$$

Consider deletion of view atom p. By Example 1, a weakest explanation of p is $\mathcal{E}= \{a \vee b, e \vee f\}$. For $a \vee b$ (respectively $e \vee f$), by Algorithm 3 we get a database DB_1 with the same IDDB as DB but a new $EDDB_1$ (respectively $EDDB_2$) as follows:

$$
\begin{array}{llll}
EDDB_1: & e \vee f & EDDB_2: & a \vee b \\
 & a \vee f & & a \vee f \\
 & b \vee e & & b \vee e
\end{array}
$$

Make a disjunction of $EDDB_1$ and $EDDB_2$. By Algorithm 3, after the deletion of p, the new database has the same IDDB but a new $EDDB'$:

$$
\begin{array}{ll}
EDDB': & a \vee f \\
 & b \vee e
\end{array}
$$

5 Conclusion

This paper discusses view update problem in DDDB. This work is developed along the clue presented in [5, 2]. Abstract philosophical problems concerning view updates, such as rationality of change in [1], are not considered here. The emphasis is put on how the view update problem in DDDB can be addressed in a framework based on SLD-resolution. We achieve this by using CH-Prolog, a sound and complete proof procedure for disjunctive logic programming: Given a disjunctive deductive database DB, DB is first transformed into a special disjunctive deductive database DB' such that its intensional part is a Horn program, then the view update algorithms are obtained by modifying the related algorithms in [2]. As CH-transformation has been extended to deal with programs with nonmonotonic negation in [7], we plan to extend the method presented here to address view update problem in deductive databases with nonmonotonic negation. In addition, we are also plan to extend this framework to first order case.

References

[1] Chandrabose Aravindan and Peter Baumgartner. A rational and efficient algorithm for view deletion in databases. In Jan Maluszynski, editor, *Proceedings of International Logic Programming Symposium*. The MIT Press, 1997.

[2] John Grant, John Horty, Jorge Lobo, and Jack Minker. View updates in stratified disjunctive databases. *Journal of Automated Reasoning*, 11(2):249–267, October 1993.

[3] C. Johnson. Deduction Trees and the View Update Problem in Indefinite Deductive Databases. *Journal of Automated Reasoning* 19: 31-85, 1997

[4] Wenjin Lu and Ulrich Furbach. Disjuctive logic program = Horn program + Control program. In *Proceedings of JELIA'98*, number 1489 in Lecture Notes in Artificial Intelligence, 33–46. Springer-Verlag, 1998.

[5] Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT-Press, 1992.

[6] John W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1987. 2nd edition.

[7] Wenjin Lu. Nonmonotonic Reasoning Based on Minimal Models and Its Implementation, Ph.D. Thesis, University of Koblenz, Germany, 1998.

[8] Wenjin Lu. CH-Prolog: A Proof Procedure for Positive Disjunctive Logic Programming. In *Proceedings of The 12th International FLAIRS Conference*, 268–272. The AAAI Press, 1999.