Minimal Model Generation Based on E-Hyper Tableaux

Wenjin Lu

20/96

Minimal Model Generation Based on E-hyper Tableaux

Wenjin Lu
Dept. of Computer Science, University of Koblenz-Landau
Rheinau 1, D-56075 Koblenz
Germany

Abstract

Minimal model generation has received great attention as people have deeper and deeper understanding in the semantics of the logic programming, deductive database, non monotonic reasoning and the relationships among them. But most proposed minimal model generation procedures in literature are inappropriate in the sense that while generating minimal model, they also generate non-minimal models. This means an explicit minimization has to be employed to obtain minimal models. This may be a great factor of inefficiency. In this paper we develop an approach to generate the minimal models, without explicit minimization process, based on E-hyper tableau which is a variant of hyper tableaux. The soundness and completeness of the procedure are proved.

1 Introduction

Minimal model generation has received great attention as people have deeper and deeper understanding in the semantics of the logic programming, deductive database, non monotonic reasoning and the relationships among them. Every proposed semantics seems in some extent to be dependent on the minimal models, such as GCWA[11] for disjunctive logic program, static semantics[6] and D-WFS[2] for disjunctive normal logic program.

Several minimal model reasoning approaches have been proposed in the literature [7, 8, 4, 3, 5]. Among them the tableaux method seems to offer a promising basis for minimal model reasoning [9]. But as a minimal model generation procedure, most of them seem to be inappropriate in that: while generating minimal model, they also generate non-minimal models. This means that an explicit minimization process has to be employed to obtain minimal model. It can be a great factor of inefficiency. This deficiency lead us to consider new minimal model generation procedures. In this paper we describe a minimal model generation procedure based on E-hyper tableau which is a variant of hyper tableaux introduced in [10]. It generates all minimal models without explicit minimization process. This procedure is sound and complete in the sense that when the procedure terminated, the generated E-hyper tableau is such that each open branch in it corresponds to a minimal model and each minimal model is represented by a unique open branch.

Hyper tableau[10] is a variants of clause normal form tableaux. It combines the idea from hyper resolution and from analytic tableaux. Since the forward-chaining way of proving theorems, as it is employed by hyper tableaux, can be seen as a model generation procedure, it is very obvious to relate hyper-tableaux calculus to work done in semantics of disjunctive logic program and in disjunctive deductive database. Indeed, in [10] some basic relationships between hyper tableaux and fixpoint iteration techniques from work done in semantics of disjunctive logic program have been established. But directly taking it as a minimal model generation procedure would share the same deficiency, that is, too many redundant models are generated and an explicit minimization process has to be employed to get minimal models.

The key problem in generating minimal models of clause theories is to deal with the atoms occurring in the head of clause. In hyper tableau, there is almost no restriction on the extending step which results in great redundancy. To overcome this problem, we introduce the concept of E-hyper tableau which is a variant of hyper tableau. The basic idea comes from a fixpoint characterizations of minimal model. In hyper tableau language, it says if an open branch is corresponding to a minimal model and the branch contains more than one head atoms of one extending clause, then there is another open branch which corresponds to the same minimal model. Based on this characterizations in each extending step, instead of attaching just one atom to an extending node, we attach a atom with related "evidence" which consists of other head atoms in the current extending clause. The evidence is used to test if a branch contains more than one head atom from one extending clause. In this way, no non-minimal model will be generated, but some minimal models are lost. To mend this deficiency, a conversion procedure is developed which converts a E-hyper tableau to a new E-hyper tableau. The whole minimal model generation procedure consists of a series of extending and converting.

The rest of the paper is organised as follows. In section 2 we give definitions and background material related to disjunctive logic programming and hyper tableau. In section 3 we present a fixpoint characterization of minimal models, which provides a basis for our minimal model generation procedure. Section 4 defines E-hyper tableau and some properties about E-hyper tableau are discussed. Minimal model generation procedure based on E-hyper tableau and its soundness and completeness are given in section 5. Section 6 concludes this paper and discuss some further work.

2 Preliminaries

In what follows, we assume that the reader is familiar with the basic concepts of first-order logic. Given a first language, a disjunctive logic program P consists of logical clauses of the form

$$A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$$

where A_i $(1 \leq i \leq n)$ and B_j $(1 \leq j \leq m)$ are atoms in the language. A clause is called a fact if $m = 0$. In this paper we only consider the disjunctive logic program in those language without function symbol. We assume that all clauses in disjunctive logic program are ground. Each clause of disjunctive logic program represents the following first order formula:

$$A_1 \vee ... \vee A_n \vee \neg B_1 \vee ... \vee \neg B_m$$

If $n = 1$ for all clauses in P, then P is called a *Horn Program*.

Definition 2.1 *Given a disjunctive logic program P the Herbrand base of P, HB_P, is the set of all ground atoms that can be formed using the predicate symbols and constants in the first order language. A Herbrnd interpretation is any subset of HB_P. A Herbrand interpretation is called a Herbrand model of P if every clause in P is true in the interpretation. A model M is called minimal if no proper subset of M is a model of P.*

A literal tree T is a pair (t, λ) consisting of a finite, ordered tree t and a labelling function λ that assigns a literal to every non-root node of t. A branch of T is a path from root to a leaf. Let b be a branch in T, we denote the set of positive literals by $lit(b)$, then $lit(b)$ can be regard as an Herbrand interpretation. A hyper tableau for a disjunctive logic program P is a special literal tree in which branches can be marked as open or closed. In our case it can be inductively defined as follows.

Definition 2.2 *Let P be a disjunctive logic program and f be a selection function.*

- *Initialisation: A one node evidential tree is a basic E-hyper tableau of P. its unique branch is marked as "open".*
- *Extending: Let T be an hyper tableau for P with some open branches.*
  - *if $f(T) = b$ with leaf node N and*
  - *$C = A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$ is a clause from P such that $B_1 \wedge ... \wedge B_m$ is true but $A_1 \vee ... \vee A_n$ is false in b.*

  Then the literal tree T' is a hyper tableau for P, where T' is obtained from T by attaching $m+n$ child nodes M_1,...M_n, N_1,...,N_m to b with respective labels

  $$A_1, .., A_n, \neg B_1, ..., \neg B_m$$

  Marking every new branch $(b,\ M_1)$,...$(b,\ M_m)$ with positive leaf as "open" and Marking every new branch $(b,\ N_1)$,...$(b,\ N_n)$ with negative leaf as "closed"

Nodes M_1,...M_n, N_1,...,N_m are also called successor sequence of a node N in T'.

Different from the original definition in [10], here we require that the extending clause is not redundant because we are only interested in minimal models. It is clear that this requirement has no effect on minimal model of disjunctive logic program. That is the following fact holds[4].

Fact 1 *Let P be a disjunctive logic program, T be a hyper tableau for P such that for any open branch in T no extending step can be carried out. Then for every minimal model M of P there is an open branch b in T such that $M = lit(b)$.*

The Open branches corresponding to a minimal model are called minimal branches.

3 A Characterization of Minimal Models

In this section, we present a characterization of minimal model which provides the basis for our approach of minimal model generation. The characterization is based on the following program transformation.

Definition 3.1 *Given a disjunctive logic program P, let I be an interpretation of P and $C = A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$. Then the Horn transformation of C wrt I, denoted by $Horn(C, I)$, is the set of clauses.*

$$Horn(C, I) = \begin{cases} \emptyset : \quad \textit{if} \quad B_1 \wedge ... \wedge B_m \quad \textit{is false} \\ \{A \leftarrow B_1 \wedge ... \wedge B_m \quad | \quad A \in I \cap \{A_1, ..., A_n\}\} : \quad \textit{otherwise} \end{cases}$$

The Horn transformation of P wrt I, denoted by $Horn(P, I)$, is the set of Horn programs defined by
$Horn(P, I) = \{P' \quad | \quad P' \subseteq \bigcup_{C \in P} Horn(C, I)$ *and* $|P' \cap Horn(C, I)| = 1$ *for all* $C \in P, Horn(C, I) \neq \emptyset\}$

Definition 3.2 *Let P be an Horn logic program and let HB_P be the Herbrand base of P. Then the immediate consequence operator $T_P : \quad 2^{HB_P} \rightarrow 2^{HB_P}$ is defined as:*
$T_P(I) = \{A \in HB_P \quad | \quad A \leftarrow B_1 \wedge ... \wedge B_m$ *is a ground instance of a program clause in P and $B_1, ..., B_m$ are in I*$\}$

It is well known that T_P is monotonic and has the least fixpont $T_P \uparrow \omega$ which is the smallest model of P

Lemma 1 *For any interpretation I and any $P' \in Horn(P, I)$, $T_{P'} \uparrow \omega \subseteq I$.*

Proof: It follows from the fact that the head of any clause in P' is in I. ■

The following is the least fixpoint characterization of minimal model of disjunctive logic program.

Theorem 2 *Let M be a model of disjunctive logic program P, then M is a minimal model of P iff for any $P' \in Horn(P, M)$, $M = T_{P'} \uparrow \omega$.*

PROOF: ($\Rightarrow$) let M be a minimal model of P, we want to prove $M = T_{P'} \uparrow \omega$. To do so, by Lemma 1 we need only to show $T_{P'} \uparrow \omega$ is a model of P. If not so, then there is a clause

$$A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$$

in P such that $\{B_1, ..., B_m\} \subseteq T_{P'} \uparrow \omega$ but for any i, $A_i \notin T_{P'} \uparrow \omega$. By Lemma 1, we have $\{B_1, ..., B_m\} \subseteq M$. Because M is a model of P, $\{A_1, A_2, ..., A_n\} \cap M \neq \emptyset$. By the definition of Horn transformation, there is some i, $(1 \leq i \leq n)$ such that $A_i \leftarrow B_1 \wedge ... \wedge B_m$ in P'. So $A_i \in T_{P'} \uparrow \omega$. Contradiction.

($\Leftarrow$) Let M be a model of P and for any $P' \in Horn(P, M)$, $M = T_{P'} \uparrow \omega$, we want to prove M is a minimal model of P. If not so, let $M' \subset M$ be a minimal model of P. We construct a definite program P' as follows. For any clause of the form

$$A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$$

in P,

- If M' satisfies $B_1 \wedge ... \wedge B_m$, because M' is a model of P, there is some $A_i \in M' \subseteq M$, let $A_i \leftarrow B_1 \wedge ... \wedge B_m \in P'$.
- If M' does not satisfy $B_1 \wedge ... \wedge B_m$ but M does, then let $A_i \leftarrow B_1 \wedge ... \wedge B_m \in P'$, for some $A_i \in M$.

It is clearly that this P' exists and $P' \in Horn(P, M)$. By the construction of P', M' is a model of P'. But $M = T_{P'} \uparrow \omega$ is the least model of P'. So $M = T_{P'} \uparrow \omega \subseteq M'$. Contradiction. ■

Below we tranlate this characterization of minimal model into hyper tableau language, it provides a basis for our minimal model generation procedure.

Definition 3.3 *Let P be a disjunctive logic program and T be a hyper tableau for P. For a given open branch b of T, the Branch transformation of P w.r.t. b, denoted by $Branch(P, b)$, is a disjunctive program defined as follows. For any $C = A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m \in P$,*

- *If in some extending step of T, a prefix b' of b is selected as the extended branch with the extending clause C and $b' \circ A_i$ is a prefix of b, then $A_i \leftarrow B_1 \wedge ... \wedge B_m \in Branch(P, b)$. Otherwise*
- *$C \in Branch(P, b)$*

Nothing else in $Branch(P, b)$.

Lemma 3 *Let P be a disjunctive logic program and T be a finished hyper tableau for P and let b be a branch in T. If in some extending step of T, a prefix of b is selected as the extended branch with the the extending clause*

$$C = A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$$

and b contains more than one head atom of C, say, A_1, ..., A_k and b first goes through node N_1 labelled with A_1. Then there are $k-1$ branches b_2, ..., b_k in T which go through nodes N_2, ..., N_k (resp.) such that $lit(b_i) \subseteq lit(b)$ $(i = 2, ..., k)$, where N_2, ..., N_k are brother nodes of N_1 and are labelled with A_2, ..., A_k, respectively, in T.

PROOF: Let b' be the prefix of b selected as the extended branch with the extending clause C in some extending step of T. Let N be the last node of b', then by the definition of hyper tableau, the successor sequence N_1, ..., N_n, M_1, ..., M_m of N are labelled with A_1, ..., A_n, $\neg B_1$, ..., $\neg B_m$, respectively, b go through N_1. Now we prove that for any i $(2 \leq i \leq k)$ there is a open branch b_i of T which goes through N_i such that $lit(b_i) \subseteq lit(b)$. Cosidering program $P_{N_i} = Branch(P, b' \circ A_i)$. Then the subtree T_{N_i} obtained from T by cutting all branches not going through node N_i is a hyper tableau for P_{N_i}. $lit(b)$ is a model of P_{N_i}. Then $lit(b)$ contains a minimal model of P_{N_i} which should correspond to a minimal branch b_i in T_{N_i} by Fact 1. This means $lit(b_i) \subseteq lit(b)$.

■

Theorem 4 *In addition to the assumption of Lemma 3, if b is a minimal branch of T, then $lit(b) = lit(b_i)$ $(i = 1, ..., k)$.*

PROOF: It follows Lemma 3 and the fact that every open branch is a model of P. ■

Based on this result, in the next section we will introduce a variant of hyper tableau in which the branches containing more than one head atoms from one extending clause can be effectively tested.

4 E-hyper Tableaux

In this section, we introduce E-hyper tableau, which is a variant of hyper tableau introduced in [10]. However different from hyper tableau, instead of attaching an atom during extending step, we will attach an atom with related "evidences". With the evidences it would be easy to test if a branch contains more than one head atoms of one extending clause. To distinguish this tableau from hyper tableau we call it E-hyper tableau.

Definition 4.1 *Let P be a disjunctive logic program and let C be a clause of the form*

$$A_1 \vee ... \vee A_i \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$$

in P. Then an evidential head of C is a triple

$$(A, Nevid, Pevid)$$

where $A \in \{A_1, ..., A_n\}$ is called evidenced atom, $Nevid \subset \{A_1, ..., A_n\}$, $Pevid \subseteq \{A_1, ..., A_n\}$ and satify

- $Nevid \cap Pevid = \emptyset$
- $Nevid \cup Pevid = \{A_1, ..., A_n\}$
- $A \notin Nevid$

Especially, the evidential head of the form

$$(A_i, \{A_1, ...A_{i-1}, A_{i+1}, ...A_n\}, \{A_i\})$$

is called a normal evidential head of C. For the given clause C there is exactly n different normal evidential heads:

$$(A_1, \{A_2, ..., A_n\}, \{A_1\})$$

$$(A_2, \{A_1, ..., A_n\}, \{A_2\})$$

$$......$$

$$(A_n, \{A_1, ..., A_{n-1}\}, \{A_n\})$$

Definition 4.2 (Evidential Tree) *A evidential tree T is a pair (t, λ) consisting of a finite, ordered tree t and a labelling function λ that assigns an evidential atom or a negative atom to every non-root node of t.*
A branch of T is a path N_0,...,N_n $(n \geq 0)$ from root N_0 to leaf N_n in t and each node N_i is with respective label. Given branch $b = N_0, ..., N_n$, the branch literals $lit(b)$, negtive branch evidence $nevid(b)$ and positive branch evidence $pevid(b)$ are sets defined as belowe:

$$lit(b) = \bigcup_{(A,Nevid,Pevid)\in b} \{A\}$$

$$nevid(b) = \bigcup_{(A,Nevid,Pevid)\in b} Nevid$$

$$pevid(b) = \bigcup_{(A,Nevid,Pevid)\in b} Pevid$$

For a given branch b, when we regard b as an Herbrand interpretation we means that the interpretation consists of the atoms in $lit(b)$. When we say a formula F is true(false) in b we always means that F is true(false) in the Herbrand interpretation determined by $lit(b)$.
A branch in T can be marked as open, closed or possible closed. T is called closed if each branch of T is marked as closed or possible closed, otherwise it is open. A selection function is a total function which maps an open evidential tree to one of its open branches. If $f(T) = b$ we also say b is selected by f.

4.1 Basic E-hyper Tableau

Let P be a disjunctive logic program, then the basic E-hyper tableau for P is a special evidential tree which can be defined inductively as follows.

Definition 4.3 *Let P be a disjunctive logic program and f be a selection function.*

- *Initialization: A one node evidential tree is a basic E-hyper tableau of P. its unique branch is marked as "open".*
- *Extending and Marking: Let T be an open hyper tableau for P.*
  - *Extending:*
    - *if $f(T) = b$ with leaf node N and*
    - *$C = A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$ is a clause from P such that $B_1 \wedge ... \wedge B_m$ is true in b but $A_1 \vee ... \vee A_n$ is false in b.*

    Then the evidential tree T' is basic hyper tableau for P, where T' is obtained from T by attaching n nodes M_1,...M_n with n normal evidential head of C and m nodes N_1,...,N_m with m negative literals $\neg B_1$,..., $\neg B_m$ to b.
  - *Marking:*
    - *Marking every new branch $b \circ N_1$,...$b \circ N_m$ with negative leaf as "closed"*
    - *For each new branch $b \circ N_i$, where N_i is labelled with $(A_i, \{A_1, ..., A_{i-1}, A_{i+1}, ..., A_n\}, \{A_i\})$, if $A_i \in nevid(b)$, then marking this new branch as "possible closed".*
    - *Marking the rest branches "open"*

In the above definition, b is called extended branch of T and C extending clause. We say C is applicable to b in this context.

E-hyper tableau is really a slight modification of hyper tableau. Given a E-hyper tableau T_e for P, we can get a unique hyper tableau T for P by replacing label $(A, Nevid, Pevid)$ with the atom A. Then T_e and T have same branches (but with different labels). And for each branch b the branch literals of b in T_e and in T are same. In the following we call T is the hyper tableau projection of T_e.

Definition 4.4 *A branch in a basic E-hyper tableau is called extending finished iff either b is marked "closed" or "possible closed" or no clause in P is applicable to b. A basic E-hyper tableau T is called extending finished iff all branches in T are extending finished.*

The following lemma indicates that the basic E-hyper tableau is sound about the minimal model of disjunctive logic program.

Lemma 5 *Let P be a disjunctive logic program. If T is finished basic E-hyper tableau for P, then every open branch b in T, $lit(b)$ is a minimal model of P.*

Proof: Let b is an open branch in T, we want to prove $lit(b)$ is a minimal model of P. By the building process of basic E-hyper tableau, we know that $lit(b)$ is a model. For any $P' \in Horn(P, lit(b))$ we want to prove $T_{P'} \uparrow \omega = b$. By theorem 4, we have $T_{P'} \uparrow \omega \subseteq b$. Now we show the other direction. Notice that in basic E-hyper tableau for a given branch b and any atom $A \in lit(b)$, there is one and only one node N in b labelled with the evidential head in which A is the evidenced atom. So it is reasonable to define the distance of A in b, denoted by $d(b, A)$, to be the number of nodes in the path from root to node N, where N is labelled with the evidential head in which A is the evidenced atom.

For any $A \in lit(b)$, we prove $A \in T_{P'} \uparrow \omega$ by induction on the distance $d(b, A)$ of A in b.

- Base case: $d(b, A) = 1$. Then A is in a fact in P, say $A_1 \vee ... \vee A_n$ and $lit(b) \cap \{A_1, ..., A_n\}$ contains just one element A, otherwise it must be closed. By the Horn transformation, $A \in P'$, so $A \in T_{P'} \uparrow \omega$.

- Induction hypothesis: Assume for any $A \in b$ with $d(b, A) < k$, we have $A \in T_{P'} \uparrow \omega$.

- Induction step: If $d(b, A) = k$, then by hyper extension, there is a clause

  $$A_1 \vee ... \vee A_n \leftarrow B_1 \wedge ... \wedge B_m$$

  such that $B_1 \wedge ... \wedge B_m$ is true in b and $d(b, B_i) < k$ $(1 \leq i \leq m)$. Because b is an open branch, we have $\{A_1, ..., A_n\} \cap b$ contains just one element, say A_1, then by the Horn transformation, $A_1 \leftarrow B_1 \wedge ... \wedge B_m \in P'$. By induction hypothesis, B_1,...,B_m are in $T_{P'} \uparrow \omega$. By the definition of $T_{P'}$, we have $A \in T_{P'} \uparrow \omega$.

By induction, this proves the theorem. ■

As a minimal model generation procedure E-hyper tableau is not complete in the sense that it can only generate some of minimal models and generally can not generate all minimal model of P.

Example 4.5 *Let $P = \{A \vee B, A \leftarrow B, B \leftarrow A\}$.*

A basic E-hyper tableau for P is shown in figure 1. $\{A, B\}$ is a minimal model of P but corresponds to no open branch in figure 1.

In fact, basic hyper tableau generates only those minimal models which contains no more than one head atom of one extending clause. To obtain all minimal models of P using E-hyper tableau, in next subsection we introduce a new operation on extending finished E-hyper tableau with some possible closed branches.

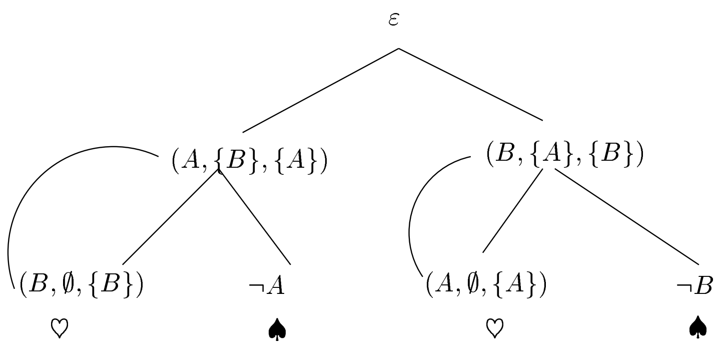

Figure 1: A basic E-hyper tableau for P without open branch. "♡" marks possible closed branch and "♠" marks closed branch

4.2 E-hyper Tableau Conversion

The example in the last subsection shows that the basic E-hyper tableau as a minimal model generation procedure is not complete because it generates only those minimal models which contains no more than one head atom of extending clause. By the building of basic E-hyper tableau, if a branch contains two head atoms from one extending clause then it must be marked as a possible closed branch.

Definition 4.6 *Let $b = b_1 \circ N \circ b_2 \circ M$ is a possible closed branch, N is labelled with $(A_N, Nevid_N, Pevid_N)$ and M is labelled with $(A_M, Nevid_M, Pevid_M)$. If $A_M \in Nevid_N$ and $A_M \notin \bigcup_{(A,Nevid,Pevid)\in b_2} Nevid$ then we say nodes N, M is the current connection (short for connection) in b, denote it as $(N, M)_b$.*

Let b and b' be two possible closed branches with connections $(N, M)_b$ and $(N', M')_{b'}$ respectively. We say branch b is delegable to branch b' iff the following conditions hold:

- *nodes N and N' are brothers in T.*
- *M is labelled with $(A_M, Nevid_M, Pevid_M)$ and N' is labelled with $(A_{N'}, Nevid_{N'}, Pevid_{N'})$ and $A_M = A_{N'}$.*

To obtain all minimal models, we have to "open" some possible closed branch. The basic idea in E-hyper tableau conversion is based on the theorem 2 of minimal model and its connection with hyper tableau stated in section 3. Let T_e be a extending finished basic E-hyper tableau and T be a finished hyper tableau obtained from the hyper tableau projection of T_e by hyper tableau extending. Let b_1 be a possible closed branch and $(N_1, M_1)_{b_1}$ be a connection in b_1. Let N_1 be labelde with $(A_{N_1}, Nevid_{N_1}, Pevid_{N_1})$ and M_1 is labelled with $(A_{M_1}, Nevid_{M_1}, Pevid_{M_1})$. Then A_{N_1} and A_{M_1} occur in the head of one extending clause. By theorem 4, if there is a miniaml branch b_1' through N_1 and M_1

(therefore $b_1 \subseteq b'_1$) in T, then in T there is another minimal branch b'_2 through some N's brother node N_2 labelled with A_{M_1} such that $lit(b'_1) = lit(b'_2)$. But some prefix b_2 of b'_2 will be closed in T_e with connection (N_2, M_2) because node N_2 is labelled with $(A_M, Nevid_{N_2}, Pevid_{N_2})$ and A_{N_1} in b'_2. So in T_e if we can open one of b'_1 and b'_2, then no minimal model will be lost. The E-hyper conversion serves for this purpose.

Definition 4.7 (E-hyper Tableau Conversion) *Let T be a E-hyper tabeleau. Let b and b' be two possible closed branches in T with connections $(N, M)_b$ and $(N', M')_{b'}$ respectively, and b is delegable to b'. N is labelled with $(A_N, Nevid_N, Pevid_N)$ and N' is labelled by $(A_{N'}, Nevid_{N'}, Pevid_{N'})$. Then the $(c.c')$-conversion of T is a E-hyper tableau T' defined as follows.*

- *Marking b closed.*
- *Labeling node N' with $(A_{N'}, Nevid'_{N'}, Pevid'_{N'})$, where*
  - $Nevid'_{N'} = Nevid_{N'} \setminus Pevid_N$
  - $Pevid'_{N'} = Pevid_{N'} \cup Pevid_N$

  If $A_k \in nevid(b)$, then marking b possible closed, otherwise open.

(c, c') is called a convertible connection pair of T. We denote the fact that T' is obtained from T by $(c.c')$-conversion as $T \rightarrow_{(c,c')} T'$.

Definition 4.8 *Let P be a disjunctive logic program, then an E-hyper tableau for P is evdential tableau obtained by initialization and extending and marking and conversion.*

Definition 4.9 *Let T be an extending finished E-hyper tableau, T is said converting finished iff T contains no convertible connection pair.*
If T is both extending finished and conveting finished, then T is called finished.

The following lemma says E-hyper tableau is complete about the minimal model.

Lemma 6 *Let P be a disjunctive logic program, T be the finished E-hyper tableau. Then for any minimal model M of P, there is an open branch b in T such that $lit(b) = M$.*

PROOF: Let T be a finished hyper tableau obtained from the hyper tableau projection of T_e by hyper tableau extending. By the the Fact 1, every minimal model of P is corresponding to a minimal branch in T. Let b_1 be a minimal branch in T, we want to prove there is open branch b_e in T_e such that $lit(b_1) = lit(b_e)$.

If b_1 is open in T_e, then let $b_e = b_1$, we have found it, otherwise let b'_1 be the prefix of b_1 which is closed in T_e with connetion $c_1 = (N_1, M_1)_{b'_1}$. Notice that b_1 is a miniaml branch, by theorem 4, in T there is a minimal branch b_2 ($b_1 \neq b_2$) which goes through one of N_1's brother N'_1 and $lit(b_1) = lit(b_2)$ in T.

If b_2 is open in T_e, then let $b_e = b_2$, we have found it, otherwise let b'_2 be the prefix of b_2 which is closed in T_e with a connection $c_2 = (N_2, M_2)_{b'_2}$, Simular argument as above we can find minimal branches each of which goes through one of N_2's brother in T and has same branch literals as b_2. We want to prove that among them there exists b_3 such that $(b_3 \neq b_1)$. Cosidering connection $c_2 = (N_2, M_2)_{b'_2}$,

- Case 1: If $N_2 = N'_1$, let N_1 be labelled with $(A_{N_1}, Nevid_{N_1}, Pevid_{N_1})$, M_2 be labelled with $(A_{M_1}, Nevid_{M_1}, Pevid_{M_1})$, then $A_{N_1} \neq A_{M_1}$. Otherwise c_2, c_1 form a convertable pair in T_e. This contradicats T_e is a converting finished E-hyper tableau. Therefore there is N_2's brother N'_2, $N'_2 \neq N_1$. In T there is minimal branch b_3 going through N'_2 such that $lit(b_3) = lit(b_2)$ but $b_3 \neq b_1$.
- Case 2: If N_2 is a successor of N'_1, then we can find such b_3 which goes through N'_1, therefore $b_3 \neq b_1$.
- case 3: If N_2 is an ancester of N'_1, then N_2 is an ancester of N_1. N'_2 is a brother of N_2, we can find such b_3 which goes through N'_2, therefore b_3 can not goes through N_1. So we also have $b_3 \neq b_1$.

Now we find different branches b_1, b_2, b_3 such that $lit(b_1) = lit(b_2) = lit(b_3)$. If b_3 is open in T_e, then let $b_e = b_3$, we have found it, otherwise let b_3 be closed in T_e with a connection $c_3 = (N_3, M_3)$. Then with simular argument we can find b_4 such that $b_i \neq b_j$ $(i \neq j, i, j = 1, 2, 3, 4)$ and $lit(b_1) = lit(b_2) = lit(b_3) = lit(b_4)$...., In this way either we can find b_e such that $lit(b) = lit(b_e)$ or we find a infinite sequence of different minimal branches b_1, b_2, b_3,..., in T such that $lit(b_i) = lit(b_{i+1})$, $(i = 1, 2, ...)$. But the late is impossible because T is finite. This means we can find b_e in T_e such that $lit(b_e) = lit(b)$. ■

Example 4.10 *Given program,* $P_1 = \{A \vee B, \quad A \vee H, \quad B \vee H_1, \quad H\}$.

A basic E-hyper tableau for P_1 is shown in figure 2. In figure 2 there are two possible closed branches $b_1 = (B, \{A\}, \{B\}) \circ (A, \{H\}, \{A\})$ with connection $c_1 = ((B, \{A\}, \{B\}), (A, \{H\}, \{A\}))_{b_1}$ and $b_2 = (A, \{B\}, \{A\}) \circ (H, \emptyset, \{H\}) \circ (B, \{H_1\}, \{B\})$ with connection $c_2 = ((A, \{B\}, \{A\}), (B, \{H_1\}, \{B\}))_{b_2}$. After $(c_1, c_2) - conversion$, we get figure 3 in which branch $b = (A, \emptyset, \{B, A\}) \circ (H, \emptyset, \{H\}) \circ (B, \{H_1\}, \{B\})$ is an open branch but $lit(b) = \{A, H, B\}$ is not a minimal model of P_1.

5 Minimal Model Generation

The example in tha last section indicates that E-hyper tableau can not be used directly as a minimal model generation procedure because it also generates non minimal model while generating all minimal models. To obtain a sound and complete minimal model generation procedure we have to put some restriction

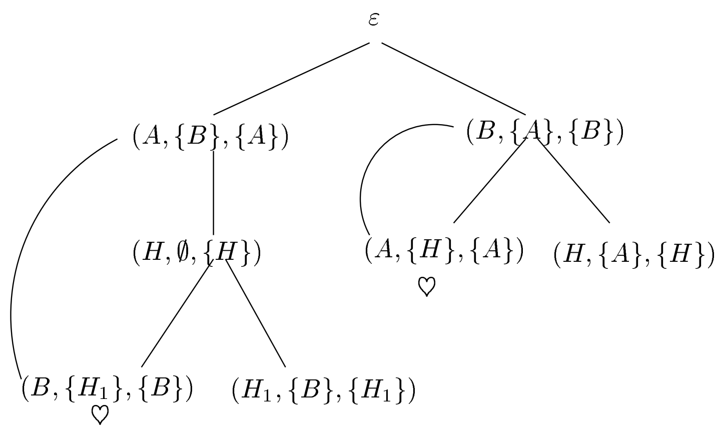

Figure 2: A Basic E-hyper tableau for P_1

on the application of conversion. This restriction is based on the following observation.

For a given disjunctive logic program P, by lemma 5, all open branches in the basic E-hyper tableau for P are minimal models. This means the non-minimal models in a E-hyper tableau are introduced by E-hyper tableau conversion. Let $c = (N, M)_b$, $c' = (N', M')_{b'}$ be a convertable connection pair, c is delegetable to c'. Let N be labelled with $(A_N, Nevid_N, Pevid_N)$ and N' be labelled with $(A_{N'}, Nevid_{N'}, Pevid_{N'})$. Then after $(c, c') - conversion$, N is labelled with a new label $(A_N, Nevid'_N, Pevid'_N)$ in which $A_{N'} \notin Nevid'_N = Nevid_N \setminus Pevid_{N'}$ This makes it possible that the open branch through N is a non-minimal branch.

Lemma 7 *Let P be a disjunctive logic program and T be a E-hyper tableau for P. Let N be a node with label $(A_N, Nevid_N, Pevid_N)$. Then the open branches through node N can not contain the open branches through node N', where N' is a brother of N and with label $(A_{N'}, Nevid_{N'}, Pevid_{N'})$ and $A_{N'} \in Nevid_N$.*

PROOF: Let b_N be a open branch through node N. then $A_{N'}$ must not occurs in b_N, otherwise b_N can not be open. But any open branch $b_{N'}$ through node N' contains $A_{N'}$. So $lit(b'_N) \not\subseteq lit(b_N)$. ■

In other words, if b_N contains an open branch $b_{N'}$, then the evidenced head $A_{N'}$ in the label of N' must occur in $Pevid_N$. But notice that, in basic E-hyper tableau T for P all nodes are labelled with normal evidential head in which $Pevid$ contains only one atom, the evidenced head. Only conversion can change $Nevid$ and $Pevid$. This suggests that we have to put some restriction on the conversion procedure to generate only minimal models. Fortunately, the idea presented in [12] can be also used in our case. The idea is based on the following theorem [12].

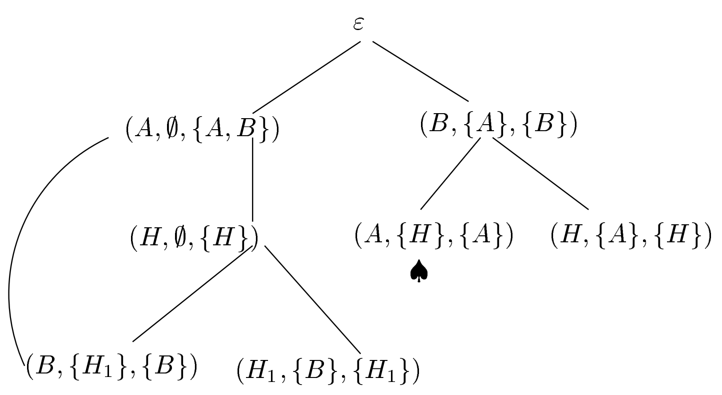

Figure 3: The E-hyper tableau for P_1 after $(c_1, c_2) - conversion$

Theorem 8 *[12] Let S be a set of clauses, M_1, ..., M_m be minimal models of S. Let $S' = S \cup \{neg(M_1), ..., neg(M_m)\}$, where $neg(M_i) = \bigvee_{A \in M_i} \neg A$ $(i = 1, ..., m)$. Then S' has the same set of minimal models as S except for the M_i $(i = 1, ..., m)$.*

As a result it is possible to restrict the generation of non minimal branch by retaining the found minimal models and adding to program the negation clause corresponding to each of these models when a conversion procedure is called.

Definition 5.1 *Let P be a disjunctive logic program and T be an extending finished E-hyper tableau for P. For any node N of T. we denote T_N the subtree with root N. T_N is called ripe iff there is no convertible connection pair in T_N.*

Intuitively, the subtree T_N is ripe if as a part of whole E-hyper tableau of P, it is "local" finished in the sense that no extending step and no conversion can be carried out in T_N.

Definition 5.2 (Minimal Model Generation Procedure)*Let P be a disjunctive logic program, then the minimal model generation procedure for P is defined as follows.*

1. *Initialization E-hyper tableau T for P.*
2. *Extending T until T is extending finished.*
3. *Marking open branch in T finished open.*
4. *If there is a convertible connection pair $c = (N, M)_b$ and $c' = (N', M')_{b'}$ and T_N is ripe, then for each open branch b through N, add $neg(b)$ to P, then do (c, c')-conversion, goto 2. If no such c, c' then stop.*

In the above definition of minimal model generation procedure, to prevent the found minimal model M from being closed by the $neg(M)$, we mark the branch corresponding to M finished open to distinguish it from open branch. After termination of the procedure, all open branches are marked as finished open, so in the following, we simple call them open branch.

Theorem 9 (Soundness) *Let P be a disjunctive logic program and T be a finished E-hyper tableau for P generated by minimal model generation procedure. Then the open branches in T are minimal branches.*

PROOF: Let T be generated as follows.

$$T_0, ..., T_{i_1} \rightarrow_{(c_1, c_1')} T_{i_1+1}, ..., T_{i_k-1} \rightarrow_{(c_k, c_k')} T_{i_k}, ..., T_{i_{k+1}} = T$$

and there be no conversion between T_0 and T_{i_1} and T_{i_l} and $T_{i_{l+1}}$ $(1 \leq l \leq k)$. We prove it by induction on the number k of conversions during the building of T.

- Base case: $k = 0$, then T_{i_1} is a basic E-hyper tableau. By lemma 5, the open branches in T_{i_1} are minimal models.
- Induction hypothesis: for $k_1 < k$, the open branches in $T_{i_{k_1}}$ are minimal models of P.
- Induction step: Let b be an open branch in $T = T_{i_{k+1}}$. We went to prove b can not properly contain any minimal model. By Lemma 7, for any N with label $(A_N, Nevid_N, Pevid_N)$ in b, if $A_{N'} \in Nevid_N$, then b can not contains the open branch through node N', where N' is a brother of N and with label $(A_{N'}, Nevid_{N'}, Pevid_{N'})$. In other word, if there is open branch $b' \subseteq b$, then there are nodes $N \in b$ with label $(A_N, Nevid_N, Pevid_N)$ and $N' \in b'$ with label $(A_{N'}, Nevid_{N'}, Pevid_{N'})$ and they are brothers in T such that $A_{N'} \in Pevid_N$. This means there is $(c_{k_1}, c'_{k_1}) - conversion$ before $(c_k, c'_k) - conversion$ in the generation procedure of T and $c_{k_1} = (N, M)_{b_1}$ and $c'_{k_1} = (N', M')_{b'_1}$, where b_1 and b'_1 are some prefixes of b and b', respectively. By the restriction on the conversion in minimal model generation procedure, the subtree $T_{N'}$ is ripe and therefore for the open branch b' , by hypothesis it is a minimal branch, $neg(b)$ is added to P, this contradicts that b is open.

By induction, this proves the theorem. ■

Theorem 10 (Completeness) *Let P be a disjunctive logic program. T is a E-hyper tableau generated by minimal model generation procedure for P. Then for any minimal modal of P there is an open branch b in T such that $M = lit(b)$.*

PROOF:By Lemma 6, finished E-hyper tableau is complete. Notice that T is a special finished E-hyper tableau with extra closed branches which are closed by some minimal open branches which are corresponding to minimal models of

P. This means either the extra closed branches in T can not be minimal model or as minimal models they have occurred in some open branches. Therefore for each minimal model of P, T has a correspondent open branch b such that $M = lit(b)$. ■

6 Conclusion and Further Work

We conclude this paper by comparing the minimal model generation procedure reported here with related works and discuss some further topics related to this paper.

The minimal model generation procedure reported in this paper is based on E-hyper tableau which is a variant of hyper tableau. The theorem 2 provides a basis of our approach. This procedure consists of three basic operators: initialization, extending and conversion and without explicit minimalization process. The minimalization process is implicitly done during the building of the E-hyper tableau by putting a constraint on the conversion procedure.

There are other tableau methods for minimal model generation reported in literature. But most of them need an explicit minimalization process, which seems to cost too much. However, after writing this paper, we find the idea for minimal model generation in [1] is very similar to us. We both are using "complement splitting" as a means to reduce the generation of non-minimal model. But the complement splitting used in [1] could be regard as a "partial complement splitting" because for a disjunction of the form $A_1 \vee A_2 \vee ... \vee A_n$, the complement splitting in [1] will produce $A_1 \wedge \neg A_2 \wedge ... \wedge \neg A_n$, $A_2 \wedge \neg A_3 \wedge ... \wedge A_n$, ..., A_n. In this sense, our complement splitting could be regarded as a "complete complement splitting" because for $A_1 \vee A_2 \vee ... \vee A_n$ we produce n evidenced heads, semantically, they are equvalent to the n formulas. $A_1 \wedge \neg A_2 \wedge ... \wedge \neg A_n$, ..., $A_n \wedge \neg A_2 \wedge ... \wedge \neg A_{n-1}$. As a result, we generate minimal models not one by one and no special search order is needed. In addition, we augment P with the negation of minimal model "by need", That is, for some P, all minimal models are not added to P. But partial complement splitting does.

We have implemented a prototype for the minimal model generation procedure reported here. The initial test result shows that as a tableau method in most cases it works well. A deficiency in our method is that we have to keep the possble closed branches for conversion. it seems expensive. Further, we plan to compare our procedure with other related methods in more detail. We also plan to use the basic method in this paper to compute other semantics of disjunctive logic porgram.

Acknowledgements

The author would like to thank all the members of Artificial Intelligence Research Group at the University Koblenz.Landau for their support and useful comments. Special Thanks to Prof. U.Furbach for his advice and encouragement.

References

[1] Bry, F. and Yahya, A. Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods 1996, 143–159. Springer-Verlag.

[2] S. Brass and J. Dix. A Disjunctive Semantics Based on Bottom-Up Evaluation and Unfolding. Proceedings of the 13th World Computer Congress '94, IFIP, GI-Workshop W2 (Disjunctive Logic Programs and Disjunctive Databases). 1994

[3] M.L. Ginsberg. A Circumscriptive Theorem Prover. Artificial intelligence, 39:209-230, 1989.

[4] K. Inoue and M. Koshimura and R. Hasegawa. Embedding Negation as Failure into a Model Generation Theorem Prover. The 11th International Conference on Automated Deduction, 400–415, Saratoga Springs, NY, USA, June 1992, LNAI 607.

[5] T. C. Przymusinski. An Algorithm to Compute Circumscription. Artificial intelligence, 38: 49-73, 1989.

[6] T.C. Przymusinski. Static Semantics for Normal and Disjunctive Logic Programs. Annals of Mathematics and Artificial Intelligence, Special Issue on Disjunctive Programs, 1995.

[7] N. Olivetti. A tableaux and sequent calculus for minimal modelentailment. Journal of Automated Reasoning, 9:99-139, 1992.

[8] A. Nerode, R.T. Ng and V.S. Subrahmanina. Computing circumscriptive databases: I. theory and algorithms. Information and Computation, 116:58-80, 1995.

[9] I. Niemelä. A Tableau Calculus for Minimal Model Reasoning. Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods, 278–294, Springer-Verlag, May 1996.

[10] P. Baumgartner and U. Furbach and I. Niemelä. Hyper Tableaux. TechReport 8–96, Fachbericht Informatik, Universität Koblenz-Landau, Available at http://www.uni-koblenz.de/universitaet/fb4/publications/GelbeReihe/

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

[12] A. Yahya, J. A. Fernandez, and J. Minker. Ordered model tree: A normal form for disjunctive deductive databases. J. Automated Reasoning, 13(1):117-144, 1994.

7/95 *Jürgen Ebert, Gottfried Vossen.* I-Serializability: Generalized Correctness for Transaction-Based Environments.

6/95 *P. Baumgartner, S. Brüning.* A Disjunctive Positive Refinement of Model Elimination and its Application to Subsumption Deletion.

5/95 *P. Baumgartner, J. Schumann.* Implementing Restart Model Elimination and Theory Model Elimination on top of SETHEO.

4/95 *Lutz Priese, Jens Klieber, Raimund Lakmann, Volker Rehrmann, Rainer Schian.* Echtzeit-Verkehrszeichenerkennung mit dem Color Structure Code — Ein Projektbericht.

3/95 *Lutz Priese.* A Class of Fully Abstract Semantics for Petri-Nets.

2/95 *P. Baumgartner, R. Hähnle, J. Posegga (Hrsg.).* 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods — Poster Session and Short Papers.

1/95 *P. Baumgartner, U. Furbach, F. Stolzenburg.* Model Elimination, Logic Programming and Computing Answers.

1994

18/94 *W. Hower, D. Haroud, Z. Ruttkay (Eds.).* Proceedings of the AID'94 workshop W9 on Constraint Processing in Computer-Aided Design.

17/94 *W. Hower.* Constraint satisfaction — algorithms and complexity analysis.

16/94 *S. Brass, J. Dix.* A Disjunctive Semantics Based on Unfolding and Bottom-Up Evaluation.

15/94 *S. Brass, J. Dix.* A Characterization of the Stable Semantics by Partial Evaluation.

14/94 *Michael Möhring.* Grundlagen der Prozeßmodellierung.

13/94 *D. Zöbel.* Program Transformations for Distributed Control Systems.

12/94 *Martin Volk, Michael Jung, Dirk Richarz, Arne Fitschen, Johannes Hubrich, Christian Lieske, Stefan Pieper, Hanno Ridder, Andreas Wagner.* GTU – A workbench for the development of natural language grammars.

11/94 *S. Brass, J. Dix.* A General Approach to Bottom-Up Computation of Disjunctive Semantics.

10/94 *P. Baumgartner, F. Stolzenburg.* Constraint Model Elimination and a PTTP Implementation.

9/94 *K.-E. Großpietsch, R. Hofestädt, C. Steigner (Hrsg.).* Workshop Parallele Datenverabeitung im Verbund von Hochleistungs-Workstations.

8/94 *Baumgartner, Bürckert, Comon, Frisch, Furbach, Murray, Petermann, Stickel (Hrsg.).* Theory Reasoning in Automated Deduction.

7/94 *E. Ntienjem.* A descriptive mode inference for normal logic programs.

6/94 *A. Winter, J. Ebert.* Ein Referenz-Schema zur Organisationsbeschreibung.

5/94 *F. Stolzenburg.* Membership-Constraints and Some Applications.

4/94 *J. Ebert (Hrsg.).* Alternative Konzepte für Sprachen und Rechner – Bad Honnef 1993.

3/94 *J. Ebert, A. Franzke.* A Declarative Approach to Graph Based Modeling.

2/94 *M. Dahr, K. Lautenbach, T. Marx, H. Ridder.* NET CASE: Towards a Petri Net Based Technique for the Development of Expert/Database Systems.

1/94 *U. Furbach.* Theory Reasoning in First Order Calculi.