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

Ludwig
Maximilians–
Universität
München

Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux

François Bry and Adnan Yahya

appeared in Proc. of *5th Wokshop on Theorem Proving with Analytic Tableaux and Related Methods,* Springer-Verlag LNAI 1071, May 1996
http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1995-5, April 1995

Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux

François Bry[1] and Adnan Yahya[1,2]
bry@informatik.uni-muenchen.de yahya@informatik.uni-muenchen.de

[1] Institut für Informatik, Ludwig-Maximilians-Universität München, Germany
[2] Electrical Engineering Department, Birzeit University, Birzeit, Palestine

Abstract. Herbrand models for clausal theories are useful in several areas of computer science. In most cases, however, the conventional model generation algorithms are inappropriate because they generate nonminimal Herbrand models and can be inefficient. This article describes a novel approach for generating minimal Herbrand models of clausal theories. The approach builds upon *positive unit hyper-resolution (PUHR) tableaux,* that are in general smaller than conventional tableaux. To generate only minimal Herbrand models, a *complement splitting* expansion rule and a specific search strategy are applied. The proposed procedure is optimal in the sense that each minimal model is generated only once, and nonminimal models are rejected before their complete construction. First measurements on an implementation point to its efficiency.

1 Introduction:

Generating Herbrand models of clausal theories is useful in several areas of computer science. In automated theorem proving, models can assist in making conjectures, that can be later checked for provability with conventional provers. In automated theorem proving and program verification, model generation can also be applied to searching for counter-examples to conjectures. Model generation is useful in logic programming and deductive databases for specifying their declarative semantics, in some approaches to query answering [2, 6, 14], and in nonmonotonic reasoning [5].

The conventional tableaux methods [3] are however inappropriate as model generation procedures because they often return redundant models [5, 10, 13]. The a posteriori detection of redundant models is tedious and might be time consuming. Moreover, redundant models are a source of inefficiency because they blow up the search space. This article describes a method for generating minimal Herbrand models of clausal theories. The proposed procedure is optimal in the sense that each minimal model is generated only once, and nonminimal models are rejected as soon as possible, in general before their complete construction. First measurements on an implementation point to the efficiency of the procedure.

The method is based on *positive unit hyper-resolution tableaux* (short *PUHR tableaux*), a (novel) formalization of an approach first introduced with the theorem prover SATCHMO [8]. PUHR tableaux are ground and positive, more precisely their nodes consist of sets of ground atoms and disjunctions of ground atoms. They are expanded by means of only two rules, the *positive unit hyper-resolution* and the *splitting* (a simple version of β expansion) rules, from *range-restricted* clauses. Range-restriction is a syntactical property required in deductive database languages which is comparable to Skolemization: although requiring an extension of the language, it preserves models in a certain sense. Both the branching factor and the size of the nodes of PUHR tableaux can be significantly smaller than those of conventional tableaux. Positive unit hyper-resolution makes it possible not to blindly instantiate universally quantified variables. Instead, it combines in one step instantiations (or γ expansions [3]) and splittings (or β expansion [3]), thus reducing the depth of PUHR tableaux.

In order to restrict the generation of nonminimal models, a *complement splitting* (or reduction in [11], folding-down in [7]) rule is applied in lieu of the classical splitting rule. Because PUHR tableaux are ground, complement splitting can be nicely and efficiently built into the method. While discarding many nonminimal models, and preventing the generation of duplicate models, complement splitting is not always sufficient to reject all nonminimal models. In order to prune redundant models as soon as possible, a special depth first search strategy is applied. The resulting procedure is sound in the sense that it generates only minimal models, and complete in the sense that it returns all minimal models of the input theory. A variation, we call MM-SATCHMO, of the SATCHMO program is given, which implements the minimal model generation procedure.

The plan of the rest of this paper is as follows. Section 2 introduces terminology and notations, defines range-restriction and PUHR tableaux, and recalls the SATCHMO implementation of PUHR tableaux. Section 3 is devoted to model generation using PUHR tableaux. Section 4 defines the minimal model generation procedure as a modified PUHR tableaux method and gives the implementation of MM-SATCHMO. The last chapter compares the proposed procedure with other approaches discussed in the literature, draws some conclusions, and points to possible directions for future research.

For space reasons, this article contains no proofs. They can be found in its complete version, which is remotely accessible via WWW.[3]

2 Preliminaries

2.1 Terminology and Notation

Throughout the paper usual terminology and notations are used, as in e.g. [3]. When not explicitly otherwise stated, a first-order language $\mathcal{L}$ is implicitly assumed. It is also assumed that two special atoms $\top$ and $\perp$ are available, expressing respectively truth and falsity, i.e. $\top$ is satisfied in every interpretation, no interpretations satisfy $\perp$.

[3] http://www.informatik.uni-muenchen.de/pms/publikationen/berichte/minimal-models.ps.gz

Every clause $C = L_1 \vee ... \vee L_k$ with negative literals $\{\neg A_1, ..., \neg A_n\}$ and positive literals $\{B_1, ..., B_m\}$ can be represented by a *clause in implication form*: $C' = A_1 \wedge .... \wedge A_n \rightarrow B_1 \vee ... \vee B_m$. $A_1 \wedge .... \wedge A_n$ is called the body of C', $B_1 \vee ... \vee B_m$ its head. If C contains no negative literals, $C' = \top \rightarrow B_1 \vee ... \vee B_m$. If C contains no positive literals, $C' = A_1 \wedge .... \wedge A_n \rightarrow \bot$.

A unifier (resp. most general unifier) σ of a conjunction of atoms $(A_1 \wedge .... \wedge A_n)$ and a sequence of atoms $(B_1, ..., B_n)$ is a unifier such that $A_i\sigma = B_i\sigma$, for all $i = 1, ..., n$. If $(A_1 \wedge .... \wedge A_n)$ and $(B_1, ..., B_n)$ have a unifier, they are unifiable.

An interpretation of $\mathcal{L}$ will be denoted as a pair $(\mathcal{D}, m)$ where the nonempty set $\mathcal{D}$ is the universe (or domain) and m is the mapping interpreting the symbols and expressions of the language.

The *universal closure* of a clause C is $\forall x_1 ... \forall x_n C$, where $x_1, ..., x_n$ are the variables occurring in C. A clause (resp. a set of clauses) is said to be *satisfied* by an interpretation when the universal closure of the clause (resp. the set of the universal closures of the clauses) is satisfied by this interpretation. A clause (resp. a set of clauses) is said to be *satisfiable* if it has at least one interpretation in which it is satisfied. A clause (resp. a set of clauses) is said to be *finitely satisfiable* if it is satisfied by an interpretation with a finite domain.

A term or formula in which no variable occurs is said to be *ground.* If $\mathcal{A}$ is a set of ground atoms, $H(\mathcal{A})$ denotes the Herbrand interpretation which satisfies a ground atom B if and only if $B \in \mathcal{A}$. A Herbrand interpretation $H(\mathcal{A})$ is said to be *finitely representable* if $\mathcal{A}$ is finite. Since confusions can be avoided from the context, a set of formulas having a finitely representable Herbrand model will be said to be *finitely representable.* Note that finite representability (of sets of formulas) and finite satisfiability are two distinct properties.

The subset relationship $\subseteq$ over sets of ground atoms induces an order $\leq$ on Herbrand interpretations: given two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ of ground atoms, $H(\mathcal{A}_1) \leq H(\mathcal{A}_2)$ if and only if $\mathcal{A}_1 \subseteq \mathcal{A}_2$. If $\mathcal{S}$ is a set of clauses, $\leq$ induces an order on Herbrand models of $\mathcal{S}$. A Herbrand model $H(\mathcal{A})$ of $\mathcal{S}$ is said to be a minimal Herbrand model of $\mathcal{S}$ if it is minimal for $\leq$, i.e. for every Herbrand model $H(\mathcal{A}')$ of $\mathcal{S}$, if $H(\mathcal{A}') \leq H(\mathcal{A})$, then $\mathcal{A}' = \mathcal{A}$.

If $\mathcal{E}$ is a set of formulas, $Units(\mathcal{E})$ denotes the set of unit clauses that are elements of $\mathcal{E}$. Variables are denoted by x with or without subscripts, constants by a, b, c or d, predicate symbols by P, Q, and R, and function symbols by f.

2.2 Range Restriction

Definition 1. (Range restricted clause) A clause (resp. a clause in implication form) is said to be range restricted if every variable occurring in a positive (resp. head) literal also appears in a negative (resp. body) literal.

Clearly, a range restricted clause in implication form is ground if its body is ground, e.g. if it is $\top$. A transformation is first defined, which associates a set $RR(\mathcal{S})$ of range restricted clauses in implication form with every set $\mathcal{S}$ of clauses in implication form.

Definition 2. (Range restriction transformation) Let $\mathcal{L}'$ be an extension of the language $\mathcal{L}$ with a unary predicate D (not belonging to $\mathcal{L}$).

For every $\mathcal{L}$-clause $C = A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m$, let $RR(C)$ be the following $\mathcal{L}'$-clause:

$$RR(C) := \begin{cases} C \text{ if } C \text{ is range restricted;} \\ D(x_1) \wedge \ldots \wedge D(x_k) \wedge A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m \text{ otherwise,} \\ \text{where } x_1, \ldots, x_k \text{ are the variables occurring in the } B_i\text{s and in none} \\ \text{of the } A_j\text{s.} \end{cases}$$

Let $\mathcal{S}$ be a set of $\mathcal{L}$-clauses. For a term t distinct from a variable occurring in $\mathcal{S}$, let C_t be the $\mathcal{L}'$-clause:

$$C_t := \begin{cases} D(x_1) \wedge \ldots \wedge D(x_k) \rightarrow D(t) \text{ if the variables } x_1, \ldots, x_k \text{ occur in } t; \\ \top \rightarrow D(t) \text{ if no variables occur in } t. \end{cases}$$

Let τ be the set of nonvariable terms occurring in $\mathcal{S}$. Let $\mathcal{S}'$ be the following set of $\mathcal{L}'$-clauses:

$$\mathcal{S}' := \begin{cases} \{C_t \mid t \in \tau\} \text{ if } \tau \neq \emptyset; \\ \{C_a\} \text{ otherwise, for some constant } a. \end{cases}$$

$RR(\mathcal{S}) := \{RR(C) \mid C \in \mathcal{S}\} \cup \mathcal{S}'$ is the range restriction of $\mathcal{S}$.

Note that by construction the clauses in $RR(\mathcal{S})$ are range restricted and that $RR(\mathcal{S})$ is finite if $\mathcal{S}$ is finite. Strictly speaking, the range restriction transformation does not preserve models because it extends the language. The following theorem, however, shows that it preserves models in a certain sense, similar to the way Skolemization does.

Theorem 3. *Let $\mathcal{S}$ be a set of clauses in a language $\mathcal{L}$ and $RR(\mathcal{S})$ be the range restriction of $\mathcal{S}$ (in an extension $\mathcal{L}'$ of $\mathcal{L}$ with a unary predicate D).*

1. *If $(\mathcal{D}, m)$ is a model of $\mathcal{S}$ and if m' is the mapping over $\mathcal{L}'$ defined as follows:*
$$m'(s) := \begin{cases} m(s) & \text{if } s \neq D, \\ \mathcal{D} & \text{if } s = D. \end{cases}$$
then $(\mathcal{D}, m')$ is a model of $RR(\mathcal{S})$.
2. *If $(\mathcal{D}, m')$ is a model of $RR(\mathcal{S})$, then $(\mathcal{D}, m'|_{\mathcal{L}})$ is a model of $\mathcal{S}$, where $m'|_{\mathcal{L}}$ denotes the restriction of m' to $\mathcal{L}$.*

In the context of PUHR tableaux, the range restriction transformation induces an enumeration of the ground terms, making the γ expansion rule of conventional tableaux [3] superfluous. Thus, if the procedures presented in this paper are applied to a set $RR(\mathcal{S})$ obtained from $\mathcal{S}$ by the transformation above, then the newly introduced atoms with predicate D have basically the same effect as an instantiation rule for the clauses of the original set $\mathcal{S}$.

When applied in a refutation procedure, instantiation is often a source of inefficiency. Note, however, that this is not the case for model generation. In contrast to refutation, model generation requires instantiation anyway, indeed, for Herbrand models are characterized as sets of *ground* atoms.

Definition 4. (Positive unit hyperresolution) Let $C = A_1 \wedge ... \wedge A_n \rightarrow E_1 \vee ... \vee E_m$ be a clause in implication form, $B_1, ..., B_n$ be n atoms such that $(A_1 \wedge ... \wedge A_n)$ unifies with $(B_1, ..., B_n)$. If σ is a most general unifier of $(A_1 \wedge ... \wedge A_n)$ and $(B_1, ..., B_n)$, then $(E_1 \vee ... \vee E_m)\sigma$ is a positive unit hyper-resolvent of C and $B_1, ..., B_n$.

Lemma 5. *The positive unit hyper-resolvent of a range restricted clause in implication form and ground atoms is a ground atom or a disjunction of ground atoms.*

Note that no occur-checks are needed for computing the positive unit hyper-resolvent of a range restricted clause in implication form and ground atoms.

2.3 Positive Unit Hyper-Resolution Tableaux

Starting from the set $\{\top\}$, the PUHR tableaux method expands a tree – or positive unit hyper-resolution (PUHR) tableau – for a set $\mathcal{S}$ of range restricted clauses in implication form by applying the following expansion rules that are defined with respect to $\mathcal{S}$. The nodes of a PUHR tableau are sets of ground atoms or disjunctions of ground atoms.

Definition 6. (PUHR tableaux expansion rules) Let $\mathcal{S}$ be a set of clauses in implication form.

Positive unit hyper-resolution (PUHR) rule:

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

where σ is a most general unifier of the body of a clause $(A_1 \wedge ... \wedge A_n \rightarrow E) \in \mathcal{S}$ with $(B_1, ..., B_n)$.

Splitting rule:

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

In the following definition, the splitting rule is applied to *ground* disjunctions.

Definition 7. (PUHR tableaux) Positive unit hyper-resolution (PUHR) tableaux for a set $\mathcal{S}$ of clauses in implication form are trees whose nodes are sets of ground atoms and disjunctions of ground atoms. They are inductively defined as follows:

1. $\{\top\}$ is a positive unit hyper-resolution tableau for $\mathcal{S}$.
2. If T is a positive unit hyper-resolution tableau for $\mathcal{S}$, if L is a leaf of T such that an application of the PUHR rule (resp. splitting rule) to formulas in L yields a formula E (resp. two formulas E_1 and E_2), then the tree T' obtained from T by adding the node $L \cup \{E\}$ (resp. the two nodes $L \cup \{E_1\}$ and $L \cup \{E_2\}$) as successor(s) to L is a positive unit hyper-resolution tableau for $\mathcal{S}$.

A branch of a positive unit hyper-resolution tableau is said to be *closed,* if it includes a node containing the atom $\perp$. A positive unit hyper-resolution tableau is said to be closed if all its branches are closed. A branch (resp. tableau) which is not closed is said to be *open.* A positive unit hyper-resolution tableau T for $\mathcal{S}$ is said to be *satisfiable* if the union of $\mathcal{S}$ with the nodes of a branch of T is satisfiable.

Note that if $\mathcal{P}$ is a path from the root to a node N of a PUHR tableaux, then by Definition 7, $N = \cup\mathcal{P}$.

Example 1. Figure 1 gives a PUHR tableau for the following set of clauses:

$$\begin{array}{ll} \top \rightarrow P(a) \vee Q(b) & \qquad P(b) \rightarrow \perp \\ P(x) \rightarrow P(f(x)) \vee Q(f(x)) & \qquad P(f(x)) \rightarrow \perp \\ Q(x) \rightarrow P(x) \vee R(x) & \qquad P(x) \wedge Q(f(x)) \rightarrow \perp \end{array}$$

For the sake of readability, the nodes of the tree of Figure 1 are labeled with the ground atoms or disjunctions of ground atoms added at these nodes. We recall that by Definition 7 the nodes of a PUHR tableau are sets of ground atoms and disjunctions of ground atoms.

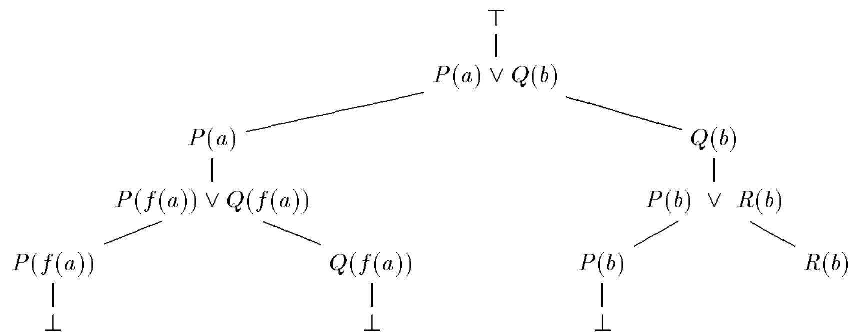

Fig. 1. A PUHR tableau.

Sets of clauses for which PUHR tableaux are defined may be infinite. According to Definition 6 clauses whose heads are $\perp$ only contribute to close branches. Since negative formulas do not explicitly occur in PUHR tableaux, closure is simply detected by the presence of $\perp$, which is simpler than checking for atomic closure [3].

Definition 8. Let $\mathcal{S}$ be a set of range-restricted clauses in implication form and $\mathcal{A}$ a set of ground atoms and disjunctions of ground atoms. $\mathcal{A}$ is said to be *saturated* with respect to $\mathcal{S}$ for the positive unit hyper-resolution and splitting expansion rules when the following properties hold:

1. if $(A_1 \wedge ... \wedge A_n \rightarrow E) \in \mathcal{S}$, $B_1 \in \mathcal{A}$, ..., and $B_n \in \mathcal{A}$, and $(A_1 \wedge ... \wedge A_n)$ and $(B_1, ..., B_n)$ are unifiable, then $E\sigma \in \mathcal{A}$ for a most general unifier σ of $(A_1 \wedge ... \wedge A_n)$ and $(B_1, ..., B_n)$.
2. If $(E_1 \vee E_2) \in \mathcal{A}$, then $E_1 \in \mathcal{A}$ *or* $E_2 \in \mathcal{A}$.

Note that if $\mathcal{B}$ is an open branch of a PUHR tableau, then $\cup\mathcal{B}$ is not necessarily saturated. As well, if $\cup\mathcal{B}$ is saturated, then $\mathcal{B}$ is not necessarily open.

Theorem 9. (Refutation soundness) *Let $\mathcal{S}$ be a set of range-restricted clauses in implication form. If there exists a closed PUHR tableau for $\mathcal{S}$, then $\mathcal{S}$ is unsatisfiable.*

Definition 10. A PUHR tableau is said to be *fair,* if the union of the nodes of each of its open branches is saturated for the expansion rules.

Informally, a PUHR tableaux is fair if along each of its open branches, each possible application of an expansion rule is performed at least once. If $\mathcal{B}$ is a branch of a tableau, then $Units(\cup\mathcal{B})$ denotes the set of unit clauses that are elements of some nodes in $\mathcal{B}$.

Theorem 11. (Refutation completeness) *Let $\mathcal{S}$ be a set of range-restricted clauses in implication form. If $\mathcal{S}$ is unsatisfiable, then every fair positive unit hyper-resolution tableau for $\mathcal{S}$ is closed.*

PUHR tableaux are defined for sets of range restricted clauses. Combined with the PUHR expansion rule of Definition 6, the range restriction transformation induces an enumeration of the ground terms.

2.4 Implementation in Prolog

The Prolog program of Figure 2 expands fair PUHR tableaux for sets of range-restricted clauses in implication form under a depth-first search strategy. The tableaux expanded by this program are strict [3] and subsumption-free. Strictness means that no application of an expansion rule is performed more than once to given clauses, atoms, or disjunctions. Subsumption-freeness means that only ground disjunctions that are not subsumed by previously generated atoms or disjunctions can be split.

Backtracking over `satisfiable` returns Herbrand models $H(\mathcal{M})$. The ground atoms of $\mathcal{M}$ are inserted into the Prolog database by the predicate `assume`. On backtracking, they are removed. A clause $A_1 \wedge ... \wedge A_n \rightarrow B_1 \vee ... \vee B_m$ is represented in the Prolog database as `A1, ..., An ---> B1 ; ... ; Bm`, where `--->` is declared as an infix binary predicate. $\perp$ is represented as `false`, $\top$ as the built-in predicate `true`, which is always satisfied. Fairness is ensured by the call to the all-solutions built-in predicate `findall`. The predicate `component` on backtracking successively returns the atoms of a disjunction. The program of Figure 2, called basic SATCHMO, as well as refinements of it, were published first in [8], where the programs are explained in more detail and performance on bench mark examples is reported. The PUHR tableaux introduced in Section 2.3 are a formalization of the principle of the SATCHMO programs.

```
satisfiable :-
   findall(Clause, violated_instance(Clause), Set),
   not (Set = []),
   !,
   satisfy_all(Set),
   satisfiable.
satisfiable.

satisfy_all([]).
satisfy_all([_B ---> H | Tail]) :-
   satisfy(H),
   satisfy_all(Tail).

component(Atom, (Atom ; _Rest)).
component(Atom, (_ ; Rest)) :-
   !,
   component(Atom, Rest).
component(Atom, Atom).

violated_instance(B ---> H) :-
   (B ---> H),
   B,
   not H.

satisfy(E) :-
   component(Atom, E),
   not (Atom = false),
   assume(Atom).

assume(Atom) :-
   asserta(Atom).
assume(Atom) :-
   once(retract(Atom)),
   fail.
```

Fig. 2. The basic SATCHMO program.

3 Model Generation with PUHR Tableaux

In the previous section, PUHR tableaux were considered from the angle of refutation. In this section, their properties with respect to model generation are investigated.

Theorem 12. (Model soundness) *Let $\mathcal{S}$ be a satisfiable set of range-restricted clauses in implication form and T a fair PUHR tableau for $\mathcal{S}$. If $\mathcal{B}$ is an open branch of T, then $H(Units(\cup\mathcal{B}))$ is a model of $\mathcal{S}$.*

Theorem 13. (Minimal model completeness) *Let $\mathcal{S}$ be a satisfiable set of range-restricted clauses in implication form, T be a fair positive unit hyper-resolution tableau for $\mathcal{S}$, and $\mathcal{M}$ a set of ground atoms. If $H(\mathcal{M})$ is a minimal model of $\mathcal{S}$, then there is a branch $\mathcal{B}$ of T such that $Units(\cup\mathcal{B}) = \mathcal{M}$.*

The following example demonstrates that a plain PUHR tableau can generate both, nonminimal and duplicate models.

Example 2. Let $\mathcal{S}$ be the following set of clauses:

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

Figure 3 is a PUHR tableau for $\mathcal{S}$. The minimal model $H(\{P(a), P(b)\})$ of $\mathcal{S}$ is generated twice, at the leftmost branch and at the third branch from the left of the PUHR tableau. The fourth branch from the left of the PUHR tableau generates the nonminimal model $H(\{P(a), P(b), P(c)\})$. Note that the PUHR tableau returns among others all minimal models of $\mathcal{S}$, i.e. $H(\{P(a), P(b)\})$, $H(\{P(a), P(d)\})$, and $H(\{P(b), P(c), P(d)\})$.

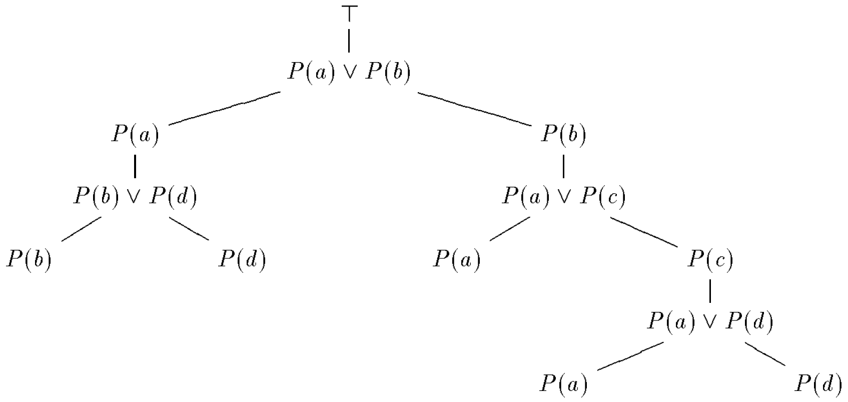

Fig. 3. A PUHR tableau with nonminimal and duplicate models.

Theorem 13 is established, though in a different context, in [1] and mentioned without proof in [6]. As the following counter-example shows, fairness is necessary in Theorem 13.

Example 3. With the theory $\mathcal{S} = \{\top \rightarrow P(a), P(x) \rightarrow P(f(x)) \vee P(b), P(a) \rightarrow P(b)\}$ consistently expanding on the second clause will not allow the generation of the (only) minimal model $H(\{P(a), P(b)\})$ of $\mathcal{S}$.

4 Minimal Model Generation

By Thorem 13 fair PUHR tableaux generate all minimal models. However, they often also generate duplicate and/or nonminimal models. A naive approach to minimal model generation consists in first expanding fair PUHR tableaux, and later pruning them from redundant branches. In this section a more efficient approach is described which consists in pruning PUHR tableaux from redundant branches as soon as possible. The pruning involves a refinement of the splitting rule, and a specific search strategy based on depth-first search. Under certain finiteness conditions, the proposed procedure is complete.

4.1 Finiteness Properties

Theorem 14. *Let $\mathcal{S}$ be a set of formulas. If $\mathcal{S}$ has a finitely representable Herbrand model it also has a finite model.*

The following result relates the finiteness of the set of minimal models to the finite representability of the minimal models. Let us call *finitary* a set of clauses, whose minimal Herbrand models are all finitely representable.

Theorem 15. *Let $\mathcal{S}$ be a set of clauses. If $\mathcal{S}$ is finitary, then $\mathcal{S}$ has finitely many minimal Herbrand models.*

Although finite representability (of a set of formulas) is a stronger property than finite satisfiability, we conjecture that it is semi-decidable like finite satisfiability. We also conjecture that the finitary property is semi-decidable.

Let $\mathcal{S}$ be a set of clauses whose minimal Herbrand models are all finitely representable. By Theorem 15 a PUHR tableau for $\mathcal{S}$ pruned from those branches corresponding to nonminimal models is finite. In many applications, the finite representability of the minimal Herbrand models is often assumed. This is the case in particular of disjunctive databases and of some forms of nonmonotonic reasoning [5].

4.2 Complement Splitting

If $C = A_1 \vee ... \vee A_n$ is a disjunction of atoms, let $Neg(C)$ denote the finite set of clauses in implication form $Neg(C) := \{A_1 \rightarrow \perp, ..., A_n \rightarrow \perp\}$.

Definition 16. (Complement splitting rule)

$$\frac{E_1 \vee E_2}{\begin{array}{l|r} E_1 & E_2 \\ Neg(E_2) & \end{array}}$$

Like the splitting rule, the complement splitting rule (already mentioned in [8], called reduction in [11] and folding-down in [7]) is applied in the following definitions to *ground* disjunctions. Tableaux expanded with the positive unit hyper-resolution and the complement splitting rules are defined inductively, similarly as in Definition 7. Let us call such tableaux *PUHR complement tableaux.*

Definition 17. (PUHR complement tableaux) Positive unit hyper-resolution (PUHR) complement tableaux for a set $\mathcal{S}$ of clauses in implication form are trees whose nodes are sets of ground atoms, disjunctions of ground atoms, and ground implications of the form $A \rightarrow \perp$. They are inductively defined as follows:

1. $\{\top\}$ is a positive unit hyper-resolution complement tableau for $\mathcal{S}$.
2. If T is a PUHR complement tableau for $\mathcal{S}$, if L is a leaf of T such that an application of the PUHR rule (resp. complement splitting rule) to formulas in L yields a formula E (resp. two sets of formulas $\{E_1, Neg(E_2)\}$ and $\{E_2\}$), then the tree T' obtained from T by adding the node $L \cup \{E\}$ (resp. the two nodes $L \cup \{E_1, Neg(E_2)\}$ and $L \cup \{E_2\}$) as successor(s) to L is a PUHR complement tableau for $\mathcal{S}$.

For PUHR complement tableaux, closedness and openness of branches and tableaux are defined like in Definition 7.

Definition 18. Let $\mathcal{S}$ be a set of range-restricted clauses in implication form and $\mathcal{A}$ a set of ground atoms, disjunctions, and clauses in implication form. $\mathcal{A}$ is said to be saturated with respect to $\mathcal{S}$ for the positive unit hyper-resolution and the complement splitting expansion rules when the following properties hold:

- if $(A_1 \wedge ... \wedge A_n \rightarrow E) \in \mathcal{S}$, $B_1 \in \mathcal{A}, ..., B_n \in \mathcal{A}$, and $(A_1 \wedge ... \wedge A_n)$ and $(B_1, ..., B_n)$ are unifiable, then $E\sigma \in \mathcal{A}$ for some most general unifier σ of $(A_1 \wedge ... \wedge A_n)$ and $(B_1, ..., B_n)$.
- If $(E_1 \vee E_2) \in \mathcal{A}$, then $\{E_1\} \cup Neg(E_2) \subseteq \mathcal{A}$, or $E_2 \in \mathcal{A}$.

Note that if $\mathcal{A}$ is saturated with respect to $\mathcal{S}$ for the positive unit hyper-resolution and the complement splitting expansion rules, then it is also saturated for the positive unit hyper-resolution and the splitting expansion rules. Model soundness for PUHR complement tableaux follows from Theorem 12.

Theorem 19. (Minimal model completeness for complement tableaux) *Let $\mathcal{S}$ be a satisfiable set of range-restricted clauses in implication form, T be a fair PUHR complement tableau for $\mathcal{S}$, and $\mathcal{M}$ a set of ground atoms. If $H(\mathcal{M})$ is a minimal model of $\mathcal{S}$, then there is a branch $\mathcal{B}$ of T such that $Units(\cup\mathcal{B}) = \mathcal{M}$.*

The following example shows that complement splitting is not always sufficient to prune all nonminimal models.

Example 4. Let $\mathcal{S} = \{\top \rightarrow P(a), P(x) \rightarrow P(b) \vee P(f(x)), P(a) \rightarrow P(b)\}$. Let T be the PUHR complement tableau for $\mathcal{S}$ by applying first the PUHR rule on $\top \rightarrow P(a)$ and $P(a) \rightarrow P(b)$, and then alternatively the PUHR and splitting rule on $P(x) \rightarrow P(b) \vee P(f(x))$. Although $H(\{P(a), P(b)\})$ is the unique minimal model of $\mathcal{S}$, T also has branches corresponding to the models $H(\{P(a), P(b), P(f(a)), ..., P(f^n(a))\})$ for all $n \in \mathbf{N}$.

Although possibly having branches corresponding to nonminimal models, PUHR complement tableaux never have two distinct branches defining the same model, as established next.

Lemma 20. *Let $\mathcal{S}$ be a satisfiable set of range-restricted clauses in implication form, T be a fair PUHR complement tableau for $\mathcal{S}$, and $\mathcal{B}_L$ and $\mathcal{B}_R$ be two open branches of T. If $\mathcal{B}_L$ appears to the left of $\mathcal{B}_R$ in T, then $Units(\cup\mathcal{B}_R) \not\subseteq Units(\cup\mathcal{B}_L)$.*

Theorem 21. *Let $\mathcal{S}$ be a satisfiable set of range-restricted clauses in implication form, T be a fair PUHR complement tableau for $\mathcal{S}$ and $\mathcal{B}_0, ..., \mathcal{B}_i, ...$ a left-to-right enumeration of the open branches of T.*

1. *$H(Units(\cup\mathcal{B}_0))$ is a minimal model of $\mathcal{S}$.*
2. *If $i \neq j$, then $Units(\cup\mathcal{B}_i) \neq Units(\cup\mathcal{B}_j)$*

The following example demonstrates that a PUHR complement tableau can generate nonminimal models. Note, however, that no models are returned twice.

Example 5. Let $\mathcal{S}$ be the set of clauses of Example 2. Figure 4 gives a PUHR complement tableau for $\mathcal{S}$. The models generated by this PUHR complement tableau are $H(\{P(a), P(d)\})$, $H(\{P(b), P(c), P(a)\})$, $H(\{P(b), P(a)\})$, and $H(\{P(b), P(c), P(d)\})$. Note that although some of them are not minimal, the PUHR complement tableau returns no duplicates.

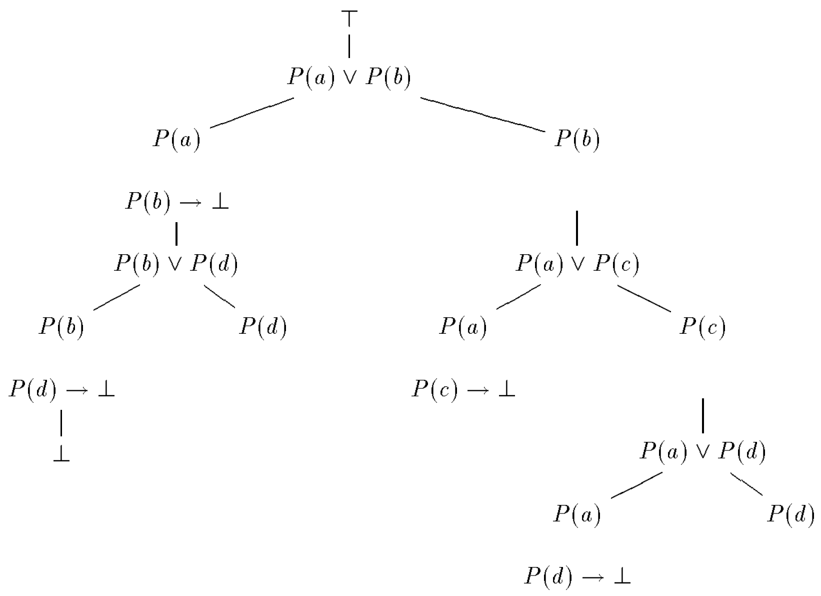

Fig. 4. A PUHR complement tableau.

4.3 Implementation of Complement Splitting

Complement splitting can be built into SATCHMO by replacing the procedure `satisfy` by the following procedure `cs_satisfy`, as shown by Figure 5. `cs_component` returns not only the atoms of a disjunction, like `component` does, but also the rest of the disjunction on the right hand side of the returned atom (`false` if this right hand side is empty). This implementation departs slightly from Definition 16 since it represents $Neg(A_1 \vee ... \vee A_n)$ as $A_1 \vee ... \vee A_n \rightarrow \bot$ instead of $\{A_1 \rightarrow \bot, ..., A_n \rightarrow \bot\}$. Since the A_i are ground, the two representations are equivalent.

```
cs_satisfy(E) :-                          cs_component(A, P, (A ; P)).
   cs_component(Atom, Suffix, E),         cs_component(A, P, (_ ; Rest)) :-
   not (Atom = false),                        !,
   assume(Atom),                              cs_component(A, P, Rest).
   assume_neg(Suffix).                    cs_component(A, false, A).

                     assume_neg(false) :-
                        !.
                     assume_neg(E) :-
                        assume(E ---> false).
```

Fig. 5. Complement splitting for basic SATCHMO

4.4 Constrained Search

By Theorem 21 the first model returned from a depth-first-left-first traversal of a PUHR complement tableau is minimal, and by Lemma 20 no models are $\leq$-larger than subsequently returned models. In order to prune PUHR complement tableaux from nonminimal models, it therefore suffices to constrain any model under construction not to be $\leq$-larger than any previously returned model. This is easily achieved by adding to the set of clauses a constraint $Neg(\{A_1, ..., A_n\}) = \{A_1 \wedge ... \wedge A_n \rightarrow \bot\}$ once a (finite) model $H(\{A_1, ..., A_n\})$ has been constructed.

Definition 22. (Minimal model generation procedure) Let $\mathcal{S}$ be a set of range restricted clauses in implication form. Applying the minimal model generation procedure to $\mathcal{S}$ consists in a depth-first-left-first construction of a fair PUHR complement tableau for $\mathcal{S}$ such that $\mathcal{S}$ is augmented with $Neg(\mathcal{M})$ after each computation of a model $H(\mathcal{M})$ of $\mathcal{S}$.

Note that, by Definitions 7 and 16, if $\mathcal{S}_1$ and $\mathcal{S}_2$ are sets of range-restricted clauses in implication form such that $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and all clauses in $\mathcal{S}_2 \setminus \mathcal{S}_1$ are of the form $A_1 \wedge ... \wedge A_n \rightarrow \bot$, then every PUHR (complement) tableau for $\mathcal{S}_2$ can be obtained from a PUHR (complement) tableau for $\mathcal{S}_1$ by discarding $\bot$ from some nodes. Conversely, every PUHR (complement) tableau for $\mathcal{S}_1$ can be obtained from a PUHR (complement) tableau for $\mathcal{S}_2$ by adding $\bot$ to some nodes. Recall that a set of clauses is finitary if its minimal Herbrand models are all finitely representable.

Theorem 23. (Soundness and completeness of the minimal model generation procedure) *Let $\mathcal{S}$ be a finite set of range-restricted clauses in implication form. If $\mathcal{S}$ is finitary, then applied on $\mathcal{S}$, the minimal model generation procedure terminates, returns all minimal models of $\mathcal{S}$ (i.e. it is complete), does not return any nonminimal model of $\mathcal{S}$ (i.e. it is sound), and does not return any minimal model more than once.*

The following example shows how the minimal model generation procedure generates only minimal models and does not return duplicates.

Example 6. Figure 6 gives the search space of the minimal model generation procedure for the set of clauses of Examples 2 and 5. Note that all models returned by the procedure are minimal.

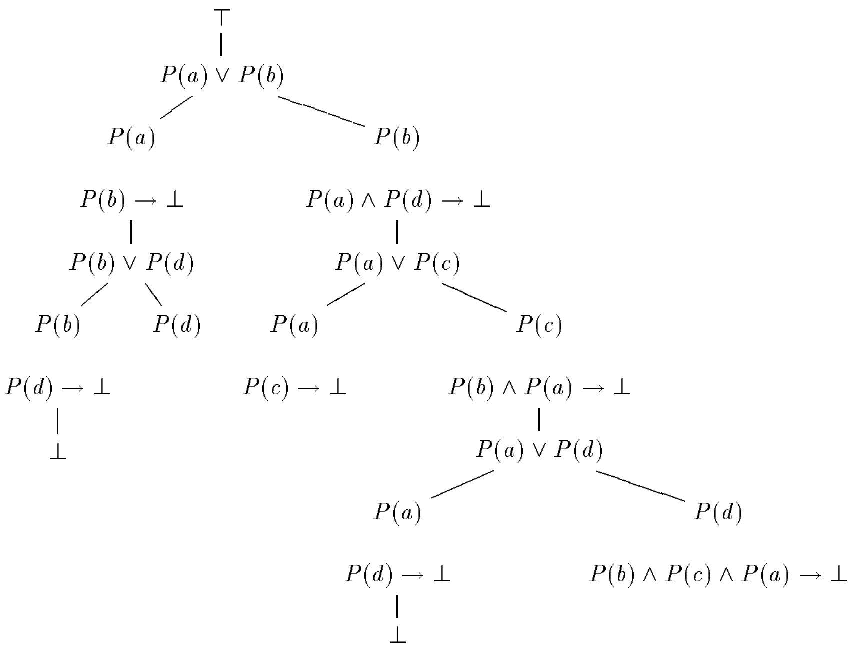

Fig. 6. A run of the minimal model generation procedure.

It is worth noting that fairness is necessary for the minimal model generation procedure, as the following counter-example shows.

Example 7. Let $\mathcal{S} = \{\top \rightarrow P(a), P(x) \rightarrow P(f(x)) \vee P(b), P(a) \rightarrow P(b)\}$. An unfair PUHR complement tableau for $\mathcal{S}$ with leftmost branch $\{P(a), P(f(a)), ..., P(f^n(a)), ...\}$ not containing $P(b)$ does not return the minimal model $H(\{P(a), P(b)\})$ and does not give rise to applying the constraint $P(a) \wedge P(b) \rightarrow \bot$ for pruning redundant branches.

4.5 MM-SATCHMO

Figure 7 gives a program, we call MM-SATCHMO, which implements the minimal model generation procedure. It builds upon the implementation of complement splitting described in Section 4.2. A slight modification of `satisfiable` suffices to construct the constraints induced by a (minimal) model. (**vi** stands for **violated_instance**, **mm_s_a** for **mm_satisfy_all**).

The argument of the procedure **mm** is the body of the constraint under construction. This data structure is redundant, for the model under construction is also represented in the Prolog database. This redundancy can be easily removed, at the cost of a less readable program. A more serious source of inefficiency lies in the way how violated clauses are detected: the last inserted atoms are not used for an incremental detection. Although quite simple, an incremental evaluation requires longer and more complicated programs. An incremental clause evaluation turns out to be especially beneficial for the constrained search.

```
minimal_model :-
   mm(true).

mm(C1) :-
    findall(Clause, vi(Clause), Set),
    not (Set = []),
    !,
    mm_s_a(Set, C1, C2),
    mm(C2).
mm(C) :-
    asserta(C ---> false).

mm_s_a([], C, C).
mm_s_a([_B ---> H | Rest], C1, C3) :-
   mm_satisfy(H, A),
   and_merge(A, C1, C2),
   mm_s_a(Rest, C2, C3).

mm_satisfy(E, Atom) :-
   cs_component(Atom, Suffix, E),
   not (Atom = false),
   assume(Atom),
   assume_neg(Suffix).

and_merge(Atom, true, Atom) :-
   !.
and_merge(Atom, Conj, (Atom, Conj)).
```

Fig. 7. The MM-SATCHMO program.

5 Conclusions and Future Work

This paper presented a positive unit hyper-resolution tableau method for computing the minimal Herbrand models of sets of range restricted clauses. The method is complete and sound in the sense that it generates all and only minimal models of its input set. A compact implementation of the procedure in the form of a short Prolog program called MM-SATCHMO was also presented.

As a tableau procedure our approach enjoys a good degree of efficiency stemming from its restricted search space, from limiting the applications of expansion rules and the use of matching without occur-check rather than full unification. This is possible because, as a side-effect of a special range-restricted syntactical form, the generated tableaux are ground. Since it makes instantiation necessary, groundness of tableaux might be considered as a source of inefficiency in a refutation procedure. However, since Herbrand models are characterized as sets of ground atoms, this objection does not apply to a model generation procedure.

As a model generation procedure, ours compares well with those reported in the literature, many of which are not sound in the sense that they generate nonminimal models [8, 6]. Compared with approaches based on model generation then testing for minimality [2, 9] our approach avoids nonminimal model generation altogether. The generation of nonminimal models is aborted as soon as possible, in general before they are fully developed. Also, the method we propose is applicable to first-order clauses and not confined to propositional or ground theories as the algorithms reported in [2, 14, 9]. While the applicability of the approach proposed in this article to first-order theories is a major advantage, most of the techniques increasing the efficiency for propositional or ground theories proposed in [14, 9] can be incorporated into a version of our algorithm tailored for that case. In [5] an approach is described for testing for a "mini-consequence" property and for avoiding models that may be made nonminimal by the existential instantiation rule (or δ expansion [3]). The concept is useful when existential quantifiers are allowed, which is not the case we consider in the present article.

Among the limitations of the procedure are its applicability only to range restricted and so called finitary theories. However, range restriction is not much of a constraint, because a model preserving transformation of general clauses into range restricted ones was given. We believe that much of real-life tasks enjoy the finiteness properties needed for the applicability of our procedure. One of the shortcomings of the procedure as reported here is its lack of incrementality. Further improvements, not discussed in this paper, can also be incorporated.

Initial testing of a prototype of this procedure points to its efficiency both as a model generator, and as a refutation system [4]. Indeed, the restriction to minimal models often dramatically reduces the search space, thus speeding up the closing of a tableaux. The prototype was able to deal with theories with a large number of minimal models with performances comparable to the best reported in the literature [9]. Further testing is needed to better evaluate the gains in performance and compare the minimal model generation procedure with existing systems.

Acknowledgments

We thank Norbert Eisinger, Heribert Schütz and Tim Geisler for the many fruitful discussions on the topic of this paper. This research was done while the second author was visiting at Ludwig-Maximilians-Universität München on an Alexander von Humboldt Research Fellowship. The support of Alexander-von-Humboldt-Stiftung is appreciated. We also thank the three anonymous referees for useful comments.

References

1. M. Denecker and D. Schreye. A framework for indeterministic model generation with equality. In *Proc. Conf. on Fifth Generation Computer Systems.*, 1992.
2. J.A. Fernández and J. Minker. Bottom-up evaluation of Hierarchical Disjunctive Deductive Databases. In *Proc.* 8^{th} *Int. Conf. on Logic Programming*, 660–675. MIT Press, 1991.
3. M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, 1987.
4. H. Schüz and T. Geisler. Efficient model generation through compilation. Tech. Rep. PMS-96-2, Inst. für Informatik, Munich University, 1996. Submitted for publication.
5. J. Hintikka. Model minimization – an alternative to circumscription. *J. Automated Reasoning*, Vol. 4, 1–13, 1988.
6. K. Inoue, M. Koshimura, and R. Hasegawa. Embedding negation as failure into a model generation theorem prover. In *Proc.* 11^{th} *Int. Conf. on Automated Deduction*, 1992.
7. R. Letz, K. Mayr, and C. Goller. Controlled integration of the cut rule into connection tableau calculi. *J. Automated Reasoning*, Vol. 13 No. 3, 297–338, 1994.
8. R. Manthey and F. Bry. Satchmo: a theorem prover implemented in Prolog. In *Proc.* 9^{th} *Conf. on Automated Deduction*, 456–459, 1988.
9. I. Niemelä. A tableau calculus for minimal model reasoning. In *Proc.* 5^{th} *Workshop on Theorem Proving with Analytic Tableaux and Related Methods,* 1996. Springer-Verlag.
10. N. Olivetti. Tableaux and sequent calculus for minimal entailment. *J. Automated Reasoning*, Vol. 9, 99–139, 1992.
11. D. Prawitz. A new improved proof procedure *Theoria*, Vol. 26, 102–139, 1960.
12. J.A. Robinson. Automatic deduction with hyper-resoultion. *Int. J. Computational Mathematics*, Vol. 1, 227–234, 1965.
13. M. Suchenek. First-order syntactic characterizations of minimal entailment, domain minimal entailment and herbrand entailment. *J. Automated Reasoning*, Vol. 10, 237–236, 1993.
14. A. Yahya, J.A. Fernandez, and J. Minker. Ordered model trees: A normal form for disjunctive deductive databases. *J. Automated Reasoning*, Vol. 13 No. 1, 117–144, 1994.

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