A Tableau Calculus for Minimal Model Reasoning

Ilkka Niemelä

University of Koblenz
Dept. of Computer Science
Rheinau 1, D-56075 Koblenz, Germany
ini@informatik.uni-koblenz.de

Abstract. The paper studies the automation of minimal model inference, i.e., determining whether a formula is true in every minimal model of the premises. A novel tableau calculus for propositional minimal model reasoning is presented in two steps. First an analytic clausal tableau calculus employing a restricted cut rule is introduced. Then the calculus is extended to handle minimal model inference by employing a groundedness property of minimal models. A decision procedure based on the basic calculus is devised and then it is extended to minimal model inference. The basic decision procedure and its extension enjoy some interesting properties. When deciding logical consequence, the basic procedure explores the search space of counter-models with a preference to minimal models and each counter-model is not generated more than once. The procedures can be implemented to run in polynomial space, and they provide polynomial time decision procedures for Horn clauses. The extended decision procedure can also be used to finding all minimal models of a set of clauses.

1 Introduction

Minimal model inference has been a focus of a great deal of attention because it is in the heart of the circumscriptive approach to formalizing common sense reasoning. Circumscription is one of the major non-monotonic logics [7] and it has been applied to formalizing inheritance reasoning, actions, plans, and the semantics of logic programs. For example, some of the most prominent semantics for disjunctive logic programs are built on top of minimal model reasoning [14].

In this paper we consider the problem of automating propositional minimal model reasoning. The goal is a method (i) that would have a good performance when the premises are Horn clauses or there is only a few non-Horn clauses (the premises are "near-Horn") and (ii) that would run in polynomial space so that really large examples with a high number of minimal models could be handled, e.g., the minimal models of large disjunctive logic programs could be computed.

There is some earlier work on implementing minimal model (circumscriptive) reasoning [8, 13, 12, 9, 11, 16] but the approaches do not seem to be able to handle larger examples because of exponential worst-case space requirements.

Analytic tableaux appear to offer a promising basis for automating minimal model reasoning for a couple of reasons. (i) Decision procedures for propositional logic with polynomial space complexity are obtained quite easily from analytic tableaux. (ii) Minimal model inference is stronger than classical inference and an interesting approach to strengthening a classical proof method to handle minimal model inference is to exploit information about the source of the failure of classical consequence. This information is readily available in tableaux in terms of open branches which provide "counter-models" for classical consequence where the premises are true but the query false. We use this information to extend a tableau calculus for propositional logic to handle minimal model reasoning.

Some problems related to the computational nature of classical cut-free analytic tableau [15] have been pointed out [3]. We present a tableau calculus for propositional clauses which seems to be better-suited to minimal model reasoning than original cut-free analytic tableaux. The new calculus employs a restricted cut-type rule but is still analytic [15] in the sense that it obeys the subformula property. It explores the search space of "counter-models" with a preference to minimal models and each "counter-model" is generated at most once. Furthermore, it has good performance on Horn and "near-Horn" clauses and has polynomial space complexity.

The outline of the paper is as follows. First we introduce the tableau calculus for propositional clauses that serves as the basis of our calculus for minimal model inference. Then we present a characterization of minimal models that is the key to the extension of the basic calculus to handle minimal model reasoning. In Section 4 we extend the basic calculus to minimal model inference. Section 5 presents decision procedures based on the calculi that have polynomial space complexity and good performance on Horn and "near-Horn" clauses. Section 6 discusses an implementation of the decision procedures and some experimental results. Section 7 compares our proposal with other related approaches presented in the literature. We finish with some concluding remarks.

2 A Tableau Calculus

In this section we introduce a tableau calculus for propositional logic that serves as the basis of our calculus for minimal model inference. The calculus is for formulae in a clausal form (conjunctive normal form). We write a clause (a disjunction) with positive literals $a_1, \dots a_m$ and negative literals $\neg b_1, \dots \neg b_n$ as

$$a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n \quad . \tag{1}$$

Often $a_1 \vee \cdots \vee a_m$ is referred to as the *head* of the clause and $b_1 \wedge \cdots \wedge b_n$ as the *body* of the clause. Sometimes we use a set notation for clauses, i.e., a clause of the form (1) is denoted by

$$A \leftarrow B$$

where $A = \{a_1, \dots, a_m\}$ and $B = \{b_1, \dots, b_n\}$.

To construct a tableau for a (finite) set of clauses Σ we first build a branch consisting of the clauses in Σ and then use two rules for extending branches:

$$
\frac{\begin{array}{l} a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n, \\ b_1, \ldots, b_n, \\ \neg a_1, \ldots, \neg a_{j-1}, \neg a_{j+1}, \ldots, \neg a_m \end{array}}{a_j} \tag{2}
$$

$$
\frac{\begin{array}{l} a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n, \\ b_1, \ldots, b_n \end{array}}{\neg a_j \mid a_j} \tag{3}
$$

The first rule is a restricted hyper-type rule which says that if a branch contains the body literals $(b_1, \ldots, b_n)$ of a clause (1) and the negations of the head literals except possibly not that of a_j $(\neg a_1, \ldots, \neg a_{j-1}, \neg a_{j+1}, \ldots, \neg a_m)$, then the branch can be extended by a_j.

The second rule is a limited cut-type rule which says that if a branch contains the body literals $(b_1, \ldots, b_n)$ of a clause (1), then for one of the head literals $a_j \in \{a_1, \ldots, a_m\}$, the branch can be extended into two branches containing $\neg a_j$ and a_j, respectively.

The application of the rules can be further restricted so that literals are not introduced repeatedly, i.e., rule (2) is applicable only when a_j does not appear on the branch and rule (3) only when neither a_j nor $\neg a_j$ appears on the branch.

Definition 1. A branch is said to be *closed* if it contains a complementary pair of literals or there is a clause $\leftarrow b_1 \wedge \cdots \wedge b_n$ on the branch such that also $b_1, \ldots, b_n$ appear on the branch. If a branch is not closed, it is said to be *open*. A tableau is open if it has an open branch otherwise it is closed.

A branch is said to be *finished* when for every $a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n$ on the branch, if $b_1, \ldots, b_n$ are on the branch, a_j is on the branch for some $j \in \{1, \ldots, m\}$. A tableau is finished if all of its branches are finished or closed.

Example 1. Consider a set $\Sigma = \{a \vee b \vee c, \leftarrow b, a \leftarrow c\}$. Below is a tableau for Σ built using the two rules.

$$
\begin{array}{c}
a \vee b \vee c \\
\leftarrow b \\
a \leftarrow c \\
\begin{array}{c|c}
\begin{array}{c} \neg a \\ \begin{array}{c|c} \begin{array}{c} \neg b \\ c \\ a \\ \times \end{array} & \begin{array}{c} b \\ \times \\ \\ \\ \end{array} \end{array} \end{array} & \begin{array}{c} a \\ \\ \\ \\ \\ \end{array}
\end{array}
\end{array}
$$

First rule (3) is applied to the first clause with respect to a. Notice that now the right most branch is already finished and we do not extend the branch anymore. Then rule (3) is applied to the first clause with respect to b. The resulting branch with b is closed as $\leftarrow b$ is on the branch and we do not extend a closed branch either. Now rule (2) is applied on the leftmost branch two times: first for $a \vee b \vee c$ and then for $a \leftarrow c$. This leads to a closed branch. Now all the branches in the tableau are either finished or closed. ■

It is quite straightforward to prove that this calculus is sound and complete for propositional clauses. The proof is based on the following observation that for every model of the initial set of premises there is an open branch with which the model agrees. A model M is said to *agree* with a branch B if for any atom a, (i) if a appears on the branch B, then $M \models a$ and (ii) if $\neg a$ appears on the branch B, then then $M \not\models a$.

Proposition 2. *In a tableau for Σ, for every model M of Σ, there is an open branch B with which M agrees.*

Proof. We can prove this by observing that the proposition holds initially and showing that it holds after each extension step of the tableau if it holds before the step. So let there be an open branch B with which the model agrees and assume that we are extending this branch.

(i) If the branch is extended by rule (2), then there is a clause $a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n \in \Sigma$ such that $\{\neg a_1, \ldots, \neg a_{j-1}, \neg a_{j+1}, \ldots, \neg a_m\} \subseteq B$ and $\{b_1, \ldots, b_n\} \subseteq B$ and the branch is extended by a_j. Now $M \models a_j$ because otherwise $M \models b_1 \wedge \cdots \wedge b_n$ but for all $i \in \{1, \ldots, m\}, M \not\models a_i$ implying that M is not a model of Σ.

(ii) If rule (3) is employed, then the branch is extended to two branches B_1 and B_2 containing $\neg a_j$ and a_j, respectively. If $M \models a_j$, then M agrees with B_2 and otherwise with B_1.

Hence, in a tableau for Σ there is an branch B with which M agrees. This branch cannot contain a complementary pair of literals a and $\neg a$ because then $M \models a$ and $M \not\models a$. There cannot be a clause $\leftarrow b_1 \wedge \cdots \wedge b_n \in \Sigma$ on the branch such that $b_1, \ldots, b_n$ appear on the branch because then M would not be a model of Σ. Hence, the branch is open. $\square$

Theorem 3 (Soundness and completeness). *Let Σ be a set of propositional clauses.*

(i) If there exists a closed tableau for Σ, then Σ is unsatisfiable.

(ii) If Σ is unsatisfiable, then any finished tableau for Σ is closed.

Proof. (i) (Soundness) Assume that Σ has a model. By Proposition 2 there is an open branch in any tableau for Σ.

(ii) (Completeness) Assume that there is a finished tableau for Σ that has an open branch B. We show that the set of atoms $\hat{B}$ on the branch B is a model of Σ. Let $a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n \in \Sigma$ and $\hat{B} \models b_1 \wedge \cdots \wedge b_n$, then for some $j \in \{1, \ldots, m\}$, $\hat{B} \models a_j$ as the branch is finished. Let $\leftarrow b_1 \wedge \cdots \wedge b_n \in \Sigma$, then $\hat{B} \not\models b_1 \wedge \cdots \wedge b_n$ because otherwise the branch would be closed. Hence, $\hat{B}$ is a model of Σ and Σ is satisfiable. $\square$

The calculus has the interesting feature that it provides an efficient decision method for Horn clauses using a simple application order of the two rules as the following proposition shows.

Proposition 4. *Let Σ be a set of propositional Horn clauses. Then any tableau for Σ closed under rule (2) is finished.*

Thus, if we first apply the rule (2) as far as possible, then for a set of Horn clauses we have a finished tableau with a single branch and rule (3) is not needed at all. Closing the initial branch under rule (2) can be implemented efficiently in polynomial time. Hence, this calculus, with a rule ordering such that rule (2) is always applied first, has the advantage that performance degrades gracefully for "near-Horn" set of clauses: for Horn clauses the run time is polynomial and the branching of the tableau depends on the amount of non-Horn clauses.

We aim to extend the calculus to handle minimal model reasoning where only minimal counter-models are of interests, i.e., minimal models of the premises where the query is false. The calculus seems to be better-suited to this than classical cut-free tableau calculi [15] where the β expansion rule is used for branching the tableau, i.e., for example, for $a \vee b$ a branch is extended into two branches containing a and b, respectively. When using the cut rule instead of the β expansion rule for branching, each counter-model is generated at most once and it is straightforward to implement the calculus in such a way that preference is given to minimal counter-models. Consider, e.g., a set of clauses $\{a \vee b_1, a \vee b_2, \ldots, a \vee b_n\}$ for which the usual β expansion rule would produce a tableau with 2^n branches whereas a tableau based on the new calculus would have at most $n+1$ branches if rule (2) is applied whenever possible. When using a heuristic favoring a cut for an atom a appearing in several clauses, a tableau with only two branches is obtained corresponding to the two minimal models $\{a\}$ and $\{b_1, \ldots, b_n\}$.

3 Properties of Minimal Models

In this section we recall the definition of minimal model inference and devise a characterization of minimal models which provides the basis for the computational treatment of minimal model inference to be developed.

Definition 5. An interpretation M (a set of atoms) is a minimal model of a set of formulae Σ iff M is a model of Σ and for no model M' of Σ, $M' \subset M$ holds.

A set of formulae Σ minimally entails a formula ϕ ($\Sigma \mid\!\sim_{\min} \phi$) iff ϕ is true in every minimal model of Σ.

For example, (i) $\{a\} \mid\!\sim_{\min} \neg b$ and (ii) $\{a \vee b, b \leftarrow a\} \mid\!\sim_{\min} \neg a \wedge b$ hold for atoms a and b. For instance, (ii) holds since $\{b\}$ is the only minimal model of $\{a \vee b, b \leftarrow a\}$.

As the basis for the computational treatment of minimal models we present a characterization of minimal models in terms of atoms not true in the models.

Proposition 6. *Let Σ be a set of formulae. An interpretation M is a minimal model of Σ iff M is a model of Σ and for every atom a, $M \models a$ implies $\Sigma \cup \mathrm{N}(M) \models a$ where $\mathrm{N}(M) = \{\neg a \mid a \text{ is an atom and } M \not\models a\}$.*

Proof. ($\Rightarrow$) Let M be a minimal model of Σ. Let a be an atom for which $M \models a$ holds. Assume $\Sigma \cup \mathrm{N}(M) \not\models a$. Hence, there is a model M' such that $M' \models \Sigma \cup \mathrm{N}(M)$ but $M' \not\models a$. Let $b \in M'$. Then $\neg b \notin \mathrm{N}(M)$ which implies $b \in M$. Thus $M' \subset M$, a contradiction and, hence, $\Sigma \cup \mathrm{N}(M) \models a$ holds.

($\Leftarrow$) Let M be a model of Σ and let for every atom a, $M \models a$ imply $\Sigma \cup \mathrm{N}(M) \models a$. Assume that there is a model M' of Σ such that $M' \subset M$. Let an atom $a \in M$. Then $\Sigma \cup \mathrm{N}(M) \models a$ holds. If $\neg b \in \mathrm{N}(M)$, then $b \notin M'$. Hence, $M' \models \mathrm{N}(M)$. But as $M' \models \Sigma$ and $\Sigma \cup \mathrm{N}(M) \models a$, then $M' \models a$. Thus, $M \subseteq M'$, a contradiction. So M is a minimal model of Σ. □

For a set of clauses, we can show that not all atoms false in the model are needed but only those appearing as positive literals in the premises.

Proposition 7. *Let Σ be a set of clauses. An interpretation M is a minimal model of Σ iff M is a model of Σ and for every atom a, $M \models a$ implies $\Sigma \cup \mathrm{N}_\Sigma(M) \models a$ where $\mathrm{N}_\Sigma(M) = \{\neg a \mid a$ is an atom appearing in the head of a clause in Σ and $M \not\models a\}$.*

Proof. By Proposition 6 it is sufficient to show that for all atoms a,

$$\Sigma \cup \mathrm{N}_\Sigma(M) \models a \text{ iff } \Sigma \cup \mathrm{N}(M) \models a \ . \tag{4}$$

($\Rightarrow$) As $\mathrm{N}_\Sigma(M) \subseteq \mathrm{N}(M)$, $\Sigma \cup \mathrm{N}_\Sigma(M) \models a$ implies $\Sigma \cup \mathrm{N}(M) \models a$. ($\Leftarrow$) Assume $\Sigma \cup \mathrm{N}_\Sigma(M) \not\models a$. Hence, there is a model M' such that $M' \models \Sigma \cup \mathrm{N}_\Sigma(M)$ but $M' \not\models a$. Let $M'' = \{b \in M' \mid b$ appears in the head of a clause in $\Sigma\}$. Clearly, $M'' \not\models a$. We show that M'' is a model of $\Sigma \cup \mathrm{N}(M)$ which implies that (4) holds. Let $a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n \in \Sigma$. If $M'' \models b_1 \wedge \cdots \wedge b_n$, then $M' \models b_1 \wedge \cdots \wedge b_n$ and thus $M' \models a_1 \vee \cdots \vee a_m$. Hence, for some $j \in \{1, \ldots, m\}, M' \models a_j$. But then $M'' \models a_j$ and $M'' \models a_1 \vee \cdots \vee a_m$. Hence, M'' is a model of Σ. If $b \in M''$, then $\neg b \notin \mathrm{N}_\Sigma(M)$ and $b \in M$. Hence, $\neg b \in \mathrm{N}(M)$ implies $b \notin M''$. Thus $M'' \models \Sigma \cup \mathrm{N}(M)$ and (4) holds. □

We say that a model M is *grounded* in Σ if for every atom a, $M \models a$ implies $\Sigma \cup \mathrm{N}_\Sigma(M) \models a$ and otherwise we say that it is ungrounded. Hence, a model of Σ is a minimal model of Σ iff it is grounded in Σ.

Example 2. Let $\Sigma = \{a \vee b, b \leftarrow a\}$. Now $M = \{a, b\}$ is not a minimal model of Σ since it is not grounded in Σ. This is because $\mathrm{N}_\Sigma(M) = \{\}$ and $\Sigma \cup \mathrm{N}_\Sigma(M) \not\models a$. But $M = \{b\}$ is a minimal model of Σ because it is grounded in Σ: $\mathrm{N}_\Sigma(M) = \{\neg a\}$ and $\Sigma \cup \mathrm{N}_\Sigma(M) \models b$ holds. ■

4 A Tableau Calculus for Minimal Model Inference

In this section we present an extension of the tableau calculus introduced in Section 2 that is capable of performing minimal model inference, i.e., capable of determining whether $\Sigma \mid\!\sim_{\min} \phi$ holds. The idea is to achieve this using a tableau approach where a tableau for $\Sigma \cup \{\neg\phi\}$ is constructed and if the tableau closes, a proof is found. As we are considering a clausal setting we assume that Σ and $\neg\phi$ are in clausal form (or are transformed into the clausal form using standard techniques [2]). In minimal model reasoning the set of premises plays a special role and to emphasize this we usually denote the original set of premises by Σ_{O}.

The tableau method for minimal model inference starts by putting Σ_O and $\neg\phi$ in clausal form into the initial branch of the tableau to be constructed. Let us call this set of clauses Υ. Then the two rules (2) and (3) are used to construct a classical tableau for Υ. If the tableau closes, then ϕ follows classically from Σ_O and, of course, also $\Sigma_\mathrm{O} \mid\!\sim_{\min} \phi$ holds. However, if the tableau has some open branches, an extension of the calculus is clearly needed because minimal model inference is stronger than classical: there are formulae ϕ for which $\Sigma_\mathrm{O} \mid\!\sim_{\min} \phi$ holds although ϕ is not a classical consequence of Σ_O. For example, $\{p\} \mid\!\sim_{\min} \neg q$ holds for atoms p and q.

Proposition 7 provides a way to strengthen the classical tableau calculus in an appropriate manner by extending the closing condition for a branch. The idea is the following. An open branch in the tableau for Υ provides a model of Υ, i.e., a counter-model for classical consequence where Σ_O is true but ϕ is false. However, it does not automatically offer a counter-model for $\Sigma_\mathrm{O} \mid\!\sim_{\min} \phi$ because the model might not be a minimal model of Σ_O. Hence, for minimal model reasoning we can strengthen the closing condition and consider an open branch not corresponding to a minimal model closed. The remaining problem is how to determine whether a model provided by an open branch is minimal. Proposition 7 offers a solution: a model is minimal iff it is grounded. The corresponding notion of (un)groundedness of a branch can be defined as follows. The model M provided by an open branch B is the set of atoms on the branch. Groundedness of a model M is formulated using the set of negative literals true in M, $\mathrm{N}_{\Sigma_\mathrm{O}}(M)$. This set can be defined directly for a branch B:

$$\mathrm{N}_{\Sigma_\mathrm{O}}(B) = \{\neg a \mid a \text{ is an atom in the head of a clause in } \Sigma_\mathrm{O} \text{ but is not in } B\} \ .$$

Definition 8. A branch B is said to be *ungrounded* in Σ_O when for some atom a on the branch, $\Sigma_\mathrm{O} \cup \mathrm{N}_{\Sigma_\mathrm{O}}(B) \models a$ does not hold.

A branch is *MM-closed* w.r.t. Σ_O if it is (classically) closed or ungrounded in Σ_O.

Hence, a tableau proof for $\Sigma_\mathrm{O} \mid\!\sim_{\min} \phi$ is an MM-closed tableau for $\Sigma_\mathrm{O} \cup \{\neg\phi\}$.

Example 3. Consider a set of clauses $\Sigma_\mathrm{O} = \{a \vee b \vee c, a \vee c \leftarrow b\}$ and the problem of determining whether $\Sigma_\mathrm{O} \mid\!\sim_{\min} \neg b$ holds. We first build a classical tableau for $\Sigma_\mathrm{O} \cup \{b\}$.

$$\begin{array}{c}
a \vee b \vee c \\
a \vee c \leftarrow b \\
b \\
\begin{array}{c|c}
\neg a & a \\
c &
\end{array}
\end{array}$$

Both branches in the tableau are finished and (classically) open. The branch B_1 on the left is ungrounded because $\mathrm{N}_{\Sigma_\mathrm{O}}(B_1) = \{\neg a\}$ and b appears on the branch but $\Sigma_\mathrm{O} \cup \mathrm{N}_{\Sigma_\mathrm{O}}(B_1) \models b$ does not hold. The branch B_2 on the right is also ungrounded because $\mathrm{N}_{\Sigma_\mathrm{O}}(B_2) = \{\neg c\}$ and b appears on the branch but $\Sigma_\mathrm{O} \cup \mathrm{N}_{\Sigma_\mathrm{O}}(B_2) \models b$ does not hold. Consequently, all the branches are MM-closed w.r.t. Σ_O and hence we have found a tableau proof for $\Sigma_\mathrm{O} \mid\!\sim_{\min} \neg b$. ■

Notice that the resulting calculus is somewhat extraordinary as the closing condition for a branch involves a test for classical consequence. However, this seems to be in harmony with the complexity results which say that propositional minimal model reasoning is a complete problem on the second level of the polynomial hierarchy [4]. If we assume that the hierarchy does not collapse, minimal model reasoning is strictly harder than classical reasoning and it cannot be reduced to classical reasoning using a mapping computable in polynomial time.

The calculus can be show to be sound and complete for propositional minimal model inference.

Theorem 9 (Soundness and completeness). *Let Σ_{O} be a set of propositional formulae, ϕ a propositional formula and Υ a set of clauses equivalent to $\Sigma_{\mathrm{O}} \cup \{\neg\phi\}$.*

(i) If a tableau for Υ is MM-closed w.r.t. Σ_{O}, then $\Sigma_{\mathrm{O}} \mid\!\sim_{\min} \phi$ holds.

(ii) If $\Sigma_{\mathrm{O}} \mid\!\sim_{\min} \phi$, then any finished tableau for Υ is MM-closed w.r.t. Σ_{O}.

Proof. (i) (Soundness) Assume that $\Sigma \mid\!\sim_{\min} \phi$ does not hold. Hence, there is a minimal model M of Σ_{O} where ϕ is not true. Then M is a model of Υ. By Proposition 2 there is an open branch B such that $\hat{B} \subseteq M$ where $\hat{B}$ is the set of atoms on the branch B. Because M is a minimal model, for every atom a, $M \models a$ implies $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(M) \models a$ by Proposition 7. If an atom a appears on the branch, then $M \models a$ and, hence, $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(M) \models a$. Now $\mathrm{N}_{\Sigma_{\mathrm{O}}}(M) \subseteq \mathrm{N}_{\Sigma_{\mathrm{O}}}(B)$ because if $\neg b \in \mathrm{N}_{\Sigma_{\mathrm{O}}}(M)$, then $b \notin M$ implying that b is not on the branch and, hence, that $\neg b \in \mathrm{N}_{\Sigma_{\mathrm{O}}}(B)$ holds. But then $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(B) \models a$ holds. Thus for every atom a appearing on the branch, then $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(B) \models a$ holds. Hence, B is grounded and open and the tableau is not MM-closed w.r.t. Σ_{O}.

(ii) (Completeness) Assume that a finished tableau for Υ has an open and grounded branch B. Now the set of atoms $\hat{B}$ on the branch B is a model of Υ because B is open and finished. Hence, $\hat{B} \models \Sigma_{\mathrm{O}} \cup \{\neg\phi\}$ holds. As B is grounded, $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(B) \models a$ holds for every $a \in \hat{B}$. Since $\mathrm{N}_{\Sigma_{\mathrm{O}}}(B) = \mathrm{N}_{\Sigma_{\mathrm{O}}}(\hat{B})$, $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(\hat{B}) \models a$ holds for every $a \in \hat{B}$, as well. By Proposition 7 this implies that $\hat{B}$ is a minimal model of Σ_{O} and hence $\Sigma_{\mathrm{O}} \mid\!\sim_{\min} \phi$ does not hold. $\square$

5 Decision Procedures

In this section we devise first a decision procedure for propositional clauses based on the calculus of Section 2. The procedure runs in polynomial space and handles efficiently Horn clauses and "near-Horn" clauses. This is achieved by

- adopting an application ordering of the rules where rule (2) is always applied first and only when it is not applicable, rule (3) is employed and
- constructing the tableau one branch at a time and covering the rest of the branches by backtracking search.

Then we present a decision procedure for minimal model inference which is an extension of the basic propositional decision procedure.

Figure 1 presents a decision procedure for propositional logic based on the calculus of Section 2. In the algorithm a branch is represented by a set literals T that contains the literals added to the branch initially or by the two rules. The procedure employs two functions:

doHrules(Υ, T) :
: This function applies rule (2) as far as possible on the branch T and detects if the resulting branch is closed. This can be stated more precisely as follows: Let H be the set of atoms derivable by rule (2) from $\Upsilon \cup T$. If $T \cup H$ contains a complementary pair of literals or there exists a clause $\leftarrow b_1 \wedge \cdots \wedge b_n \in \Upsilon$ such that $\{b_1, \ldots, b_n\} \subseteq T \cup H$, then **doHrules**$(\Upsilon, T)$ returns '`closed`' and otherwise it returns the set $T \cup H$.

nexthead(Υ, T) :
: This function checks whether the branch T is already finished and if not, it returns an atom from a head of a clause whose body is satisfied in T but the head is not. More precisely this can be formulated as follows: If there exists a clause $a_1 \vee \cdots \vee a_m \leftarrow b_1 \wedge \cdots \wedge b_n \in \Upsilon$ such that $\{b_1, \ldots, b_n\} \subseteq T$ but $\{a_1, \ldots, a_m\} \cap T = \emptyset$, then **nexthead**$(\Upsilon, T)$ returns an atom $a_j, 1 \leq j \leq m$ such that $\neg a_j \notin T$[1] and otherwise it returns '`nonexisting`'.

The procedure **tab** searches for an open finished branch in the following way. Given Υ and T, rule (2) is first propagated as far as possible by the function **doHrules**(Υ, T). If the resulting branch T_{n} is not closed, then an atom for the cut-type rule (3) is chosen by the function **nexthead**$(\Upsilon, T_{\mathrm{n}})$. If **nexthead**$(\Upsilon, T_{\mathrm{n}})$ returns `nonexisting`, an open finished branch has been found. If **nexthead**$(\Upsilon, T_{\mathrm{n}})$ returns an atom, the branch is not yet finished and rule (3) is employed where the two alternative branches are explored using backtracking. If no open finished branch is found, **tab** returns '`closed`'.

The procedure finishes after finitely many steps because during every recursive call the number of *undetermined heads*, i.e., atoms b which appear in the head of a clause in Υ but for which neither b nor $\neg b$ is on the branch T, diminishes. When there are no undetermined heads, the branch is either closed or finished and **nexthead**(Υ, T) returns '`nonexisting`'.

The procedure can be proved sound and complete in a strong sense where the procedure examines only models M that agree with the set T, i.e., (i) $a \in T$ implies $M \models a$ and (ii) $\neg a \in T$ implies $M \not\models a$.

Theorem 10 (Soundness and completeness). *Let Υ be a set of clauses and T a set of literals. Then* **tab**(Υ, T) *returns* `closed` *iff there is no model M of Υ agreeing with T.*

The usual soundness and completeness result is obtained by starting with an empty initial branch.

Corollary 11. *Let Υ be a set of clauses. Then* **tab**$(\Upsilon, \emptyset)$ *returns* `closed` *iff Υ is unsatisfiable.*

---

[1] Note that such a_j with $\neg a_j \notin T$ always exists because when the function is called, T contains no complementary pair of literals and is closed under rule (2).

```
function tab(Υ, T)
T_n := doHrules(Υ, T)
if T_n = closed then return closed
else
    a := nexthead(Υ, T_n);
    if a = nonexisting then return open
    else
        if tab(Υ, T_n ∪ {¬a}) returns open then return open
        else return tab(Υ, T_n ∪ {a})
        endif
    endif
endif
```

Fig. 1. A decision procedure for propositional logic

Notice that as discussed in Section 2 this procedure runs in polynomial time when Υ is a set of Horn clauses: after the first call of **doHrules**(Υ, T) the branch is either closed or finished.

Next we show how to modify the decision procedure **tab** to get a decision procedure for minimal model inference with the same properties: polynomial space complexity and good performance on Horn and "near-Horn" clauses. The modification is quite straightforward. We just incorporate the groundedness test to a stage in the procedure **tab** where an open finished branch is found. Here the idea is simply that `open` is returned only if the branch is grounded in Σ_{O}. The resulting procedure **mm** is presented in Fig. 2.

The procedure **mm** can be proved sound and complete in a strong sense where the procedure examines only models that agree with the set T. This means that **mm** is capable of performing *focused minimal model search*, i.e., restricting its attention to a class of minimal models where a given set of literals is true.

Theorem 12 (Soundness and completeness). *Let T be a set of literals, Σ_{O} a set of propositional formulae and ϕ a propositional formula and let Υ be a set of clauses equivalent to $\Sigma_{\mathrm{O}} \cup \{\neg\phi\}$. Then* **mm**$(\Sigma_{\mathrm{O}}, \Upsilon, T)$ *returns* `closed` *iff ϕ is true in all minimal models M of Σ_{O} agreeing with T.*

The usual soundness and completeness result follows straightforwardly.

Corollary 13. *Let Σ_{O} be a set of propositional formulae and ϕ a propositional formula and let Υ be a set of clauses equivalent to $\Sigma_{\mathrm{O}} \cup \{\neg\phi\}$. Then* **mm**$(\Sigma_{\mathrm{O}}, \Upsilon, \emptyset)$ *returns* `closed` *iff $\Sigma_{\mathrm{O}} \mid\!\sim_{\min} \phi$.*

This procedure can be modified easily to generate all minimal models of a set of formulae. Let us call **mm**$'$ a procedure which is obtained from **mm** by making the following change. When an open finished branch passing the groundedness test is found, **mm**$'$ outputs the set of atoms on the branch and returns '`closed`', instead of just returning '`open`' as **mm** does.

```
function mm(Σ_O, Υ, T)
T_n := doHrules(Υ, T)
if T_n = closed then return closed
else
    a := nexthead(Υ, T_n);
    if a = nonexisting then
        if for all atoms b ∈ T_n, Σ_O ∪ N_{Σ_O}(T_n) ⊨ b then return open
        else return closed
        endif
    else
        if mm(Σ_O, Υ, T_n ∪ {¬a}) returns open then return open
        else return mm(Σ_O, Υ, T_n ∪ {a})
        endif
    endif
endif
```

Fig. 2. A decision procedure for minimal model inference

Corollary 14. *Let Σ_{O} be a set of propositional formulae and Υ a set of clauses equivalent to Σ_{O}. Then $\mathbf{mm}'(\Sigma_{\mathrm{O}}, \Upsilon, \emptyset)$ outputs all minimal models of Σ_{O}.*

5.1 Optimizations

The decision procedure in Fig. 2 is readily implementable provided that we have a sound and complete propositional decision procedure for the class of formulae we are considering. The external decision procedure is needed for the groundedness tests, i.e. for determining whether $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(T) \models b$ holds for the atoms b on an open finished branch T. It should be noted that groundedness tests are classical consequence tests where minimal model reasoning plays no role. Hence, any propositional decision procedure can be employed. In our implementation we use the decision procedure **tab** for the groundedness tests. Notice that the external decision procedure is needed in a rather limited way.

(i) A groundedness test is actually needed only for atoms that were added to the branch T initially or by rule (3) because if these are grounded, then the other atoms are, too. This implies that for Horn clauses groundedness tests (and, consequently, calls to the external decision procedure) are not needed.
(ii) When the decision procedure is invoked, the theorem-proving task might be limited in a nice way by the negative literals $\mathrm{N}_{\Sigma_{\mathrm{O}}}(T)$ given as premises.
(iii) The decision procedure is used only when we already have a model of Σ_{O} and we have to verify that it is minimal.
(iv) Actually only one call to the external decision procedure is needed for each open finished branch: a branch T is grounded iff $\Sigma_{\mathrm{O}} \cup \mathrm{N}_{\Sigma_{\mathrm{O}}}(T) \models b_1 \wedge \cdots \wedge b_n$ holds where $\{b_1, \ldots, b_n\}$ is the set of atoms that were initially on the branch T or have been introduced by rule (3).

6 An Implementation

In this section we develop an implementation method for the procedures **tab** and **mm** where we reduce the set of premises on the branch in question always after adding a new literal to the branch. Figure 3 presents the procedure **mm** in a modified form where (i) optimizations related to the groundedness tests described in the previous section have been added and (ii) an extra argument New, containing the new literals to be added to the branch, has been included to the functions **mm** and **doHrules**. The new argument is only for efficiency and it enables incremental reductions of the set of premises Υ each time the branch is extended by new literals. To implement the groundedness test we use the decision procedure **tab** which can be implemented using the same technique.

```
function mm(Σ_O, Υ, New, T)
Mark the positive atoms in New;
⟨Υ_n, T_n⟩ := doHrules(Υ, New, T)
if T_n = closed then return closed
else
    a := nexthead(Υ_n, T_n);
    if a = nonexisting then
        if tab(Σ_O ∪ {← b_1 ∧ ··· ∧ b_m}, N_{Σ_O}(T_n)) returns closed
                where {b_1, ..., b_m} is the set of all marked atoms in T_n then
            return open
        else
            return closed
        endif
    else
        if mm(Σ_O, Υ_n, {¬a}, T_n) returns open then return open
        else return mm(Σ_O, Υ_n, {a}, T_n)
        endif
    endif
endif
```

Fig. 3. An implementation of **mm**

The idea is to maintain the set of clauses Υ in a reduced form

$$\Upsilon_{\mathrm{O}}^{T} = \{A' \leftarrow B' \mid A \leftarrow B \in \Upsilon_{\mathrm{O}}, B' = B - T, A' = A - \{b \mid \neg b \in T\}, T \cap A = \emptyset\}$$

with respect to the branch T in question where Υ_{O} is the original set of clauses. If Υ is in the reduced form, i.e., $\Upsilon = \Upsilon_{\mathrm{O}}^{T}$, then the function **nexthead**(Υ, T) can be implemented by choosing an atom a from the head of a clause with an empty body in Υ if such a clause exists and otherwise returning '`nonexisting`'.

So the only thing remaining to be done is to devise an implementation of the function **doHrules**$(\Upsilon, \mathrm{New}, T)$ that computes the new atoms H derivable from $\Upsilon \cup T \cup \text{New}$ using rule (2) and performs the necessary reductions. More precisely, **doHrules**$(\Upsilon, \text{New}, T)$ should return a pair $\langle \Upsilon_\text{n}, T_\text{n} \rangle$ such that

$$\Upsilon = \Upsilon_\text{O}^T \text{ implies } \Upsilon_\text{n} = \Upsilon_\text{O}^{T \cup \text{New} \cup H}$$
$$T_\text{n} = T \cup \text{New} \cup H$$

if $T \cup \text{New} \cup H$ contains no pair of complementary literals nor a clause $\leftarrow b_1 \wedge \cdots \wedge b_n \in \Upsilon$ such that $\{b_1, \ldots, b_n\} \subseteq T \cup \text{New} \cup H$, otherwise $\langle \Upsilon_\text{n}, T_\text{n} \rangle = \langle \Upsilon, \texttt{closed} \rangle$.

With the reduced form the application of rule (2) is quite easy to implement: an atom is derivable from the clauses Υ_O on the branch T using rule (2) iff $\{a\} \leftarrow \emptyset \in \Upsilon_\text{O}^T$. We implement **doHrules**$(\Upsilon, \text{New}, T)$ with a function **applyhyperrule**(Υ, New) that returns $\langle \Upsilon', \text{New}' \rangle$ where

$$\Upsilon' = \{A' \leftarrow B' \mid A \leftarrow B \in \Upsilon, B' = B - \text{New}, A' = A - \{b \mid \neg b \in \text{New}\}, \text{New} \cap A = \emptyset\}$$

and $\text{New}' = \{a \mid \{a\} \leftarrow \emptyset \in \Upsilon'\}$ if $\emptyset \leftarrow \emptyset \notin \Upsilon'$ and otherwise it returns $\langle \Upsilon, \texttt{closed} \rangle$. Hence, if $\Upsilon = \Upsilon_\text{O}^T$, then New' is the set of atoms derivable by one application of rule (2) from $\Upsilon_\text{O} \cup T \cup \text{New}$ but which are not yet in $T \cup \text{New}$. To satisfy the specification for **doHrules**$(\Upsilon, \text{New}, T)$ it is enough to iterate **applyhyperrule**(Υ, New) until no new atom is derivable by rule (2) ($\text{New}' = \emptyset$) or the branch closes. Figure 4 describes an implementation of the two functions.

6.1 Some Experimental Results

We have made a prototype implementation of the decision procedure for propositional clauses and its extension to minimal model inference in ECLiPSe Prolog.

The initial test results are quite encouraging. Because of the polynomial space complexity, the decision procedure has no difficulties in handling examples with even hundreds of thousands of minimal models and minimal models can be generated quite efficiently. For example, the procedure finds all the 341 minimal models of $\Upsilon \cup S_a$ in less than 2 seconds (on a SUN Sparc 4) where

$$\Upsilon = \{a_i \vee b_i \vee c_i \vee d_i \vee e_i \mid i \in \{1, \ldots, 4\}\}$$
$$S_a = \{a_2 \leftarrow a_3, a_3 \leftarrow a_4, a_4 \leftarrow a_1\}$$

needing roughly 6 ms per minimal model. Notice that there are 20 atoms in the premises for which there is over a million (2^{20}) truth assignments.

We tried to compare our implementation with an implementation of the method in [11]. For small examples our Prolog implementation compares favorably to the available C implementation [5] of this method. The method [11] with its current implementation was not able to handle very large examples. It could not even cope with examples where there are relatively few minimal models such as the following set of clauses

$$\Upsilon \cup S_a \cup S_b \cup S_c \cup S_d$$

which has only 17 minimal models where S_b, S_c, and S_d are defined similarly to S_a, e.g., $S_b = \{b_2 \leftarrow b_3, b_3 \leftarrow b_4, b_4 \leftarrow b_1\}$. Our implementation was able to generate all the minimal models in less than 0.5 seconds.

```
function doHrules(Υ, New, T)
Υ' := Υ
repeat
    if T ∪ New contains a complementary pair of literals then New := closed
    else
        T := T ∪ New;
        ⟨Υ', New⟩ := applyhyperrule(Υ', New)
    endif
until New = ∅ or New = closed
if New = closed then return ⟨Υ, closed⟩
else return ⟨Υ', T⟩
endif

function applyhyperrule(Υ, New)
Υ' := ∅; New' := ∅;
for each A ← B ∈ Υ do
    if New ∩ A = ∅ then
        B' := B − New; A' := A − {b | ¬b ∈ New};
        if B' = ∅ then
            if A' = ∅ then return ⟨Υ, closed⟩
            else if A' = {a} then New' := New' ∪ {a}
            endif
        endif
        Υ' := Υ' ∪ {A' ← B'}
    endif
endfor;
return ⟨Υ', New'⟩.
```

Fig. 4. An implementation of **doHrules**.

7 Related Work

Our basic tableau calculus is similar to the tableau calculus of D'Agostino and Mondadori [3] which also employs a cut rule as the only branching rule. The differences are that their calculus is for arbitrary formulae whereas we have a clausal calculus and our cut rule is more restricted that theirs. Actually they propose a series of restrictions on the use of cut but our cut rule (3) is more restricted than the strictest (canonical) version of the cut rule that they consider.

Our basic tableau calculus is closely related to the Davis-Putnam decision procedure for propositional logic (see, e.g. [6]). When considering the implementation method proposed in Section 6, the close correspondence is clearly visible. Rules (2) and (3) play the roles of the one-literal rule and the splitting rule in the Davis-Putnam procedure, respectively. However, rule (2) does not correspond exactly to the one-literal rule because it cannot extend the branch with a negative literal. Rule (2) could be modified to match the one-literal rule without compromising the soundness and completeness of the calculus. There are other differences, as well. The affirmative-negative rule in the Davis-Putnam procedure says that if an atom does not appear in the body of any of the clauses, all clauses having the atom in the head can be removed (by implicitly assuming the atom to be true). This rule cannot be used for generating minimal models because some minimal models might then be lost. For example, for $\Sigma = \{a \vee b\}$ the Davis-Putnam procedure with the affirmative-negative rule returns that Σ is satisfiable with (implicitly) assuming both a and b true and, thus, misses the two minimal models $\{a\}$ and $\{b\}$ of Σ. Another difference is that when applying the splitting rule in the basic Davis-Putnam procedure there is no counter-part for the hyper-type condition of our cut rule (3). So, e.g, for $\Sigma = \{a \leftarrow b, b \leftarrow a\}$, the basic Davis-Putnam procedure applies the splitting rule and returns two models $\emptyset$ and $\{a, b\}$ whereas our restricted cut rule is not applicable and only the unique minimal model $\emptyset$ is generated.

There are some other approaches where a tableau calculus has been extended to handle minimal model reasoning [12, 9, 1]. The main difference is that these approaches have exponential worst-case space complexity whereas ours works in polynomial space in the propositional case. Olivetti [12] presents a tableau calculus for arbitrary formulae. In this approach the exponential worst-case space complexity stems from the fact that a complete tableau needs to be build and comparisons between different branches are required.

Inoue et al. [9] devise an approach to generating minimal models for function-free clauses using a model generation theorem prover MGTP which is a parallel and refined version of SATCHMO [10]. The basic calculus in this model generation approach is similar to our propositional tableau calculus as in both calculi hyper-type rules are used where all the body literals of a clause have to be on a branch before the clause can be used for extending the branch. The difference is that we use a very restricted unit resulting (non-branching) hyper rule (2) and for branching a cut rule (3) whereas in the model generation approach branching is handled by the usual β expansion rule. The use of a cut rule leads to a calculus where it is easy to guarantee that the same (counter-)model is not generated more than once and that branches often correspond to minimal models. This is not easily achievable when the usual β expansion rule is employed as discussed in the end of Section 2. Inoue et al. [9] do not address the problem of efficiently testing the minimality of the generated models but seem to rely on a brute force method where all generated models are compared against each other to find the minimal ones. This is the source of exponential space complexity in this approach.

Bry and Yahya [1] propose a refinement of SATCHMO [10] to handle minimal model generation. In the resulting system, MM-SATCHMO, the usual β expansion rule is replaced by a more restricted *complement splitting* rule. This provides to a large degree the same advantages as the usage of the cut rule: each model is generated at most once and minimal models are preferred to a larger extent than when using β expansion. The model generation approach in MM-

SATCHMO is quite close to our propositional calculus. However, it does not seem to be quite straightforward to prefer branches corresponding to minimal models to the degree that is easily achievable in our calculus by applying rule (2) whenever possible and using a suitable heuristic to choose atoms for the cut.

Although MM-SATCHMO and our approach are quite similar with respect to classical reasoning, there are major differences when considering the extension to minimal model reasoning because of different techniques employed to test the minimality of models. We use the groundedness condition which can be tested independent of other minimal models. MM-SATCHMO has to generate model candidates in a certain order and compare each candidate with all previously generated minimal models. This is the source of exponential space complexity because a set of premises can have an exponential number of minimal models. Hence, MM-SATCHMO has difficulties when number of minimal models to be generated grows. In our approach no extra space is needed to generate the next minimal model and the computation time does not increase with the number of minimal models generated so far. Our approach to testing minimality seems to be more flexible, as well. By using the groundedness test we can handle both minimal model generation and query evaluation for all models as well as for a given restricted set of models (focused model search). MM-SATCHMO can only cope with minimal model generation for all models.

8 Conclusions

We have presented a tableau calculus for propositional clauses and based on it we have devised a decision procedure that possesses some attractive properties. (i) When deciding logical consequence, it explores the search space for counter-models with a preference to minimal models. (ii) Each counter-model is not generated more than once. (iii) It has a polynomial space complexity for all clauses and (iv) a polynomial time complexity for Horn clauses. Hence, the calculus offers an interesting basis for automating minimal model reasoning. We have extended the calculus to handle minimal model inferences and devised a decision procedure which enjoys the same attractive features as the underlying propositional decision procedure. We have outlined an implementation technique for the decision procedure and made a prototype implementation. The initial experimental results are encouraging and the implementation is able to handle test cases with hundreds of thousands of minimal models. This has not been possible using previous methods.

There are many interesting topics for further research. It seems that the method presented here can be extended to handle also more general forms of minimal model reasoning such as parallel circumscription with fixed and varying predicates. Also extension to the first-order case appears to be feasible. An interesting area is to study further optimizations such as good heuristics for applying the cut rule and to develop systematic testing and evaluation methods.

Acknowledgments

I would like to thank Chandrabose Aravindan, Peter Baumgartner, and Ulrich Furbach for stimulating discussions and comments on the topic of the paper.

References

1. F. Bry and A. Yahya. Minimal model generation with positive unit hyper-resolution tableaux. In *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 143–159, Terrasini, Italy, May 1996. Springer-Verlag. (This volume).
2. C. Chang and R. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.
3. M. D'Agostino and M. Mondadori. The taming of the cut. *Journal of Logic and Computation*, 4:285–319, 1994.
4. T. Eiter and G. Gottlob. Propositional circumscription and extended closed world reasoning are Π_2^p-complete. *Theoretical Computer Science*, 114:231–245, 1993.
5. R. Emery. Computing circumscriptive databases. Master's Thesis, University of Maryland, 1992.
6. M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, New York, 1990.
7. D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors. *Handbook of Logic and Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*. Oxford University Press, Oxford, 1994.
8. M.L. Ginsberg. A circumscriptive theorem prover. *Artificial Intelligence*, 39:209–230, 1989.
9. K. Inoue, M. Koshimura, and R. Hasegawa. Embedding negation as failure into a model generation theorem prover. In *The 11th International Conference on Automated Deduction*, pages 400–415, Saratoga Springs, NY, USA, June 1992. Springer-Verlag.
10. R. Manthey and F. Bry. SATCHMO: a theorem prover implemented in Prolog. In *Proceedings of the 9th International Conference on Automated Deduction*, pages 415–434, Argonne, USA, May 1988. Springer-Verlag.
11. A. Nerode, R.T. Ng, and V.S. Subrahmanian. Computing circumscriptive databases: I. theory and algorithms. *Information and Computation*, 116:58–80, 1995.
12. N. Olivetti. A tableaux and sequent calculus for minimal entailment. *Journal of Automated Reasoning*, 9:99–139, 1992.
13. T.C. Przymusinski. An algorithm to compute circumscription. *Artificial Intelligence*, 38:49–73, 1989.
14. T.C. Przymusinski. Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence, Special Issue on Disjunctive Programs*, 1995. To appear.
15. R.M. Smullyan. *First-Order Logic*. Springer-Verlag, Berlin, 1968.
16. A. Yahya, J.A. Fernandez, and J. Minker. Ordered model trees: A normal form for disjunctive deductive databases. *Journal of Automated Reasoning*, 13(1):117–144, 1994.