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

Ludwig
Maximilians
Universität
München
LMU

Model Generation for Applications – A Tableaux Method Complete for Finite Satisfiability

François Bry and Sunna Torge

http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1997-15, Dezember 1997

Model Generation for Applications – A Tableaux Method Complete for Finite Satisfiability

François Bry and Sunna Torge

Institut für Informatik, Ludwig-Maximilians-Universität München, Germany**

Abstract. For many applications of automated reasoning, tableaux methods have several advantages over theorem provers that do not generate models. A couple of such applications are briefly discussed and it is argued that they need methods that are not only refutation complete but also complete for finite satisfiability. The novel approach *Extended Positive Tableaux* is introduced which was developed for such applications. Extended Positive Tableaux rely on positive unit hyper-resolution and "range restriction" for avoiding the "blind instantiation" performed by the γ rule of standard tableaux. Instead of relying on Skolemization, as most refutation methods do, Extended Positive Tableaux use an extended δ rule. This rule makes the Extended Positive Tableaux method complete not only for refutation, like standard tableaux methods, but also for finite satisfiability. A prototype written in Prolog implements the Extended Positive Tableaux method.

1 Introduction

For many practical applications of automated reasoning, tableaux methods [20, 10, 22, 23] have the following advantages: They not only detect unsatisfiability but also generate models; they are close to common sense reasoning, hence easy to enhance with an explanation tool; and they are quite easy to adapt to the special syntax used in some applications. In the next sections, a few applications are briefly recalled. For these applications, the above mentioned particularities of tableaux methods, especially their "model generation character", are beneficial.

However, for most applications the standard tableaux methods suffer from the following drawbacks: They are often significantly less efficient than resolution based methods and they sometimes initiate the construction of infinite models, even if finite models exist.

In this paper, a novel approach called *Extended Positive Tableaux* is introduced, which aims at overcoming these drawbacks. Like the Positive Unit Hyper-Resolution (short: PUHR) Tableaux [15, 6, 7] they refine and extend, Extended Positive Tableaux rely on positive unit hyper-resolution and "range restriction" [11] for avoiding the "blind instantiation" performed by the γ rule of standard tableaux [20, 10]. Thanks to range restriction, Extended Positive Tableaux can represent interpretations as sets

** Oettingenstraße 67, D – 80538 München, torge@informatik.uni-muenchen.de

of ground positive literals. This is beneficial for two reasons. First, it often considerably reduces the search space. Second, it is well suited in application areas such as Artificial Intelligence, Databases, and Logic Programming, where this representation of interpretations and models is usual.

Instead of relying on Skolemization, as most refutation methods do, Extended Positive Tableaux use the extended δ rule of [5, 12, 14]. This rule makes the Extended Positive Tableaux method complete not only for refutation, like standard tableaux methods, but also for finite satisfiability. Arguably, its completeness for both, unsatisfiability and finite satisfiability, makes the Extended Positive Tableaux method particularly convenient for applications.

A prototype written in Prolog implements the Extended Positive Tableaux method.

2 Applications of Model Generation

In several application areas, specific techniques have been developed that can be expressed as a systematic search for models of first-order logic specifications. In the following, a few such areas are briefly described.

Diagnosis. The approach to diagnosis described in [17] relies on rules of the form $P_1 \wedge ... \wedge P_n \rightarrow C_1 \vee ... \vee C_m$ interpreted as follows: the premisses $P_1, ..., P_n$ are causes for symptoms C_1, ..., or C_m. Generating a diagnostic thus consists in building up models of both the set of rules and in selecting those models that satisfy the observed symptoms. Diagnosis in fact requires to seek for models that are as small as possible, for simpler explanations are to be preferred to redundant ones: This principle is known as "Occam's razor".

Database View Updates. A database view can be defined as the universal closure of a rule of the form $P_1 \wedge ... \wedge P_n \rightarrow C$. Such a view gives rise to compute instances of C from instances of the P_i, thus making it possible not to blow up the database with "C data". If the view, i.e. the set of derived "C data", is to be updated, changes to the P_i corresponding to the desired view update have to be determined. This is conveniently expressed as a model generation problem [2]. Meaningful solutions to a view update problem obviously have to be finite. Thus, view updates can only be computed by model generators that are complete for finite satisfiability.

Database Schema Design. In general, a database is "populated" from an initial database consisting of empty relations and views, and integrity constraints [11]. It is, however, possible that ill-defined integrity constraints prevent the insertion of any data. A model generator can be applied to detect such cases [3]: populating the database will be possible if and only if its schema has a nonempty and finite model. The system described in [4] for assisting in the design of database integrity constraints relies on the tableau method presented in the next sections.

Planning and Design. Solving planning and design problems can as well be seen as model generation. The specifications might describe an environment, the possible movements of a robot, a starting position, and a goal to reach. They can also describe how a complex object can be built from atomic components. In both cases, each finite model describes a solution while infinite models are meaningless.

In the above mentioned four applications, the models seeked for must be finite. Finitely representable models – as generated e.g. by the method described in [9] – would not provide with acceptable solutions in case of the above-mentioned applications.

Program Verification. In program verification, one tries to prove properties from programs, e.g. loop invariants. Often enough, program drafts do not fulfill their specifications. Model generators can be applied to (a logic representation of) the programs to generate "samples", or "cases" in which a requirement is violated. These samples can then be used for correcting the programs under development. Clearly Occam's razor applies: The simplest samples are preferable over larger ones, that would be interpreted as "redundant" by programmers.

Theorem Proving. Refutation theorem proving can benefit from model generation in a similar manner. If a conjecture C is not a consequence of a set $\mathcal{S}$ of formulas, then, applying a model generator to $\mathcal{S} \cup \{\neg C\}$ will construct counterexamples to the conjectured theorem, i.e. models of $\mathcal{S} \cup \{\neg C\}$. These models can be used for correcting the conjecture C. Here again Occam's razor applies: if counterexamples can be found, the "smallest" ones will better help in understanding the flaw in the conjecture than redundant counterexamples.

Counterexamples to program specifications and conjectures do not have to be finite. For these applications, counterexamples that can be found in finite time, i.e. that are finitely representable, are sufficient. However, in case finite counterexamples exist, it is desirable to detect them. For this purpose, a model generator complete for finite satisfiability is needed.

Note that the applications mentioned here in general do not give hints for the size of the finite models seeked for. Note also that most applications require that the model generator constructs only "minimal models" [6]. This related issue is beyond the scope of the present paper.

3 Preliminaries

Throughout this paper, a language with a denumerable number of constants, but without function symbols other than constants, is assumed.

The interpretations (and models) considered are *term interpretations* (term models, resp.) that, except for their domains, are defined like Herbrand interpretations (models, resp.) [10]. The domain of a term interpretation $\mathcal{I}$ consists in all ground terms, here constants, occurring in the ground atoms satisfied by $\mathcal{I}$, if this set is nonempty. Otherwise, it is assumed to be $\{c_0\}$, where c_0 is a given constant. This definition is natural for artificial intelligence and database applications.

A term interpretation is uniquely characterized by the set $\mathcal{G}$ of ground atoms it satisfies, and will therefore be denoted by $\mathcal{T}(\mathcal{G})$. If $\mathcal{S}$ is a set (finite set, resp.) of formulas and $\mathcal{T}(\mathcal{G})$ a term model of $\mathcal{S}$, there might be constants (finitely many constants, resp.) in $\mathcal{S}$ which do not occur in $\mathcal{G}$. These constants are assumed to be interpreted over a special constant $\mathbf{c}$ which neither occur in $\mathcal{S}$ nor in $\mathcal{G}$.

The subset relation $\subseteq$ induces an order $\leq$ on term interpretations: $\mathcal{T}(\mathcal{G}_1) \leq \mathcal{T}(\mathcal{G}_2)$ iff $\mathcal{G}_1 \subseteq \mathcal{G}_2$. A term model of a set of formulas is said to be *minimal*, if it is minimal for $\leq$.

The first-order language considered is assumed to include two atoms $\bot$ and $\top$ that respectively evaluate to false and true in all interpretations. A negated formula $\neg F$ will always be treated as the implication $F \to \bot$. The multiple quantification $\forall x_1 x_2 \dots x_n F$, also noted $\forall \bar{x} F$ if $\bar{x}$ is the tuple of variables $x_1 x_2 \dots x_n$, is a shorthand notation for $\forall x_1 \forall x_2 \dots \forall x_n F$. The notation $\forall \epsilon F$, where ϵ denotes the empty tuple, is allowed and stands for the formula F. Except when otherwise stated, "formula" is used in lieu of "closed formula".

If $\bar{x}$ is a tuple of variables $x_1 \dots x_n$ and if $\bar{c}$ is a tuple of constants $c_1 \dots c_n$, then $[\bar{c}/\bar{x}]$ will denote the substitution $\{c_1/x_1, \dots, c_n/x_n\}$.

4 Positive Formulas with Restricted Quantifications

In this section, a fragment of first-order logic, that of "positive formulas with restricted quantifications", is introduced. Arguably, this fragment is convenient for applications. It is shown to have the same expressive power as full first-order logic.

Positive formulas with restricted quantifications are defined relying on auxiliary notions that are first introduced.

Definition 1.
• Positive conditions *are inductively defined as follows:*

1. *Atoms except* $\bot$ *are positive conditions.*
2. *Conjunctions and disjunctions of positive conditions are positive conditions.*
3. $\exists y F$ *is a positive condition if* F *is a positive condition.*

• Ranges for variables $x_1, \dots, x_n$ *are inductively defined as follows:*

1. *An atom in which all of* $x_1, \dots,$ *and* x_n *occur is a range for* $x_1, \dots, x_n$.
2. $A_1 \vee A_2$ *is a range for* $x_1, \dots, x_n$ *if both* A_1 *and* A_2 *are ranges for* $x_1, \dots, x_n$.
3. $A_1 \wedge A_2$ *is a range for* $x_1, \dots, x_n$ *if* A_1 *is a range for* $x_1, \dots, x_n$ *and* A_2 *is a positive condition.*
4. $\exists y R$ *is a range for* $x_1, ..., x_n$ *if* R *is a range for* $y, x_1, ..., x_n$ *and if* $x_i \neq y$ *for all* $i = 1, \dots, n$.

• Positive Formulas with Restricted Quantifications *(short PRQ formulas) are inductively defined as follows:*

1. *Atoms (in particular* $\bot$ *and* $\top$*) are PRQ formulas.*
2. *Conjunctions and disjunctions of PRQ formulas are PRQ formulas.*
3. *A formula of the form* $P \to F$ *is a PRQ formula if* P *is a positive condition and* F *a PRQ formula.*
4. *A formula of the form* $\forall x_1 \dots x_n \ (R \to F)$ *(*$n \geq 1$*) is a PRQ formula if* R *is a range for* $x_1, \dots, x_n$ *and if* F *is a PRQ formula.*
5. *A formula of the form* $\exists x (R \wedge F)$ *is a PRQ formula if* R *is a range for* x *and if* F *is a PRQ formula.*

Note, that ranges are positive conditions. The following Lemma will be used in proving Theorem 4.

Lemma 1. *Let $\mathcal{M}$ and $\mathcal{N}$ be sets of ground atoms such that $\mathcal{M} \subseteq \mathcal{N}$ and R a positive condition. If $\mathcal{T}(\mathcal{M}) \models R$, then $\mathcal{T}(\mathcal{N}) \models R$.*

Proof. (sketched) By induction on the structure of R. ∎

The restriction to PRQ formulas is not a severe restriction for most applications since (1) quantifications in natural languages are restricted, and (2) for every finite set $\mathcal{F}$ of first-order formulas there exists a finite set PRQ($\mathcal{F}$) of PRQ formulas with the "same" models as $\mathcal{F}$ in the following sense:

Theorem 1. (Expressive Power of PRQ Formulas) *Let Σ be the signature of the first-order language under consideration, D a unary predicate such that $D \notin \Sigma$, $\Sigma' = \Sigma \cup \{D\}$. Then for every finite set $\mathcal{F}$ of first-order formulas over Σ there exists a finite set $PRQ(\mathcal{F})$ of PRQ formulas over Σ' such that:*

1. *If $(\mathcal{D}, m)$ is a model of $\mathcal{F}$ with domain $\mathcal{D}$ and assignment function m and if m' is the mapping over Σ' 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 $PRQ(\mathcal{F})$.
2. *If $(\mathcal{D}', m')$ is a model of $PRQ(\mathcal{F})$, then there exists $\mathcal{D} \subseteq \mathcal{D}'$ such that $(\mathcal{D}, m' \mid_\Sigma)$ is a model of $\mathcal{F}$, where $m' \mid_\Sigma$ denotes the restriction of m' to Σ.*

Proof. (sketched) Let $\mathcal{F}$ be a finite set of formulas. Recall that there exists a finite set $\mathcal{G}$ of formulas in prenex conjunctive normal form such that $\mathcal{F}$ and $\mathcal{G}$ are logically equivalent. Recall also that a disjunction $D = D_1 \vee \cdots \vee D_n$ of atoms is equivalent to the implication $P \rightarrow C$ with (1) $P = P_1 \wedge \ldots \wedge P_k$ if the set $\{\neg P_i \mid i = 1, \ldots, k\}$ of negative literals in D is nonempty, $P = \top$ otherwise, and (2) $C = C_1 \vee \ldots \vee C_m$ if the set $\{C_i \mid i = 1, \ldots, m\}$ of positive literals in D is nonempty, $C = \bot$ otherwise. Call "in implication form" the formula obtained from a formula in prenex conjunctive normal form by transforming each of its conjuncts into the above-mentioned, logically equivalent implication form. Hence, there exists a finite set $\mathcal{F}''$ of formulas in implication form which is logically equivalent to $\mathcal{F}$. Let $\mathcal{F}'$ be the finite set of PRQ formulas obtained by applying the following transformation $\mathcal{T}$ to the formulas in $\mathcal{F}''$: $\mathcal{T}(\forall x F) := \forall x(D(x) \rightarrow \mathcal{T}(F))$, $\mathcal{T}(\exists x F) := \exists x(D(x) \wedge \mathcal{T}(F))$, and $\mathcal{T}(F) := F$ if F is not a quantified formula.

One easily verifies that $\mathrm{PRQ}(\mathcal{F}) := \mathcal{F}' \cup \mathcal{R}(\mathcal{F}) \cup \mathcal{C}(\mathcal{F})$ fulfills the condition of Theorem 1, where $\mathcal{R}(\mathcal{F}) := \{\forall x_1 \ldots x_n(R(x_1, \ldots x_n) \rightarrow D(x_1) \wedge \ldots \wedge D(x_n)) \mid R \text{ } n\text{-ary}$ predicate occurring in $\mathcal{F}\}$ and $\mathcal{C}(\mathcal{F}) := \{D(c) \mid c$ constant occurring in $\mathcal{F}\}$ if some constants occur in $\mathcal{F}$, $\mathcal{C}(\mathcal{F}) := \{c_0\}$ for some arbitrary constant c_0, otherwise. ∎

Corollary 1. *For every finite set $\mathcal{F}$ of first-order formulas there exists a finite set $PRQ(\mathcal{F})$ of PRQ formulas such that $\mathcal{F}$ is finitely satisfiable if and only if $PRQ(\mathcal{F})$ has a finite term model.*

Proof. From Theorem 1 follows that for every finite set $\mathcal{F}$ of first-order formulas there exists a finite set PRQ($\mathcal{F}$) of PRQ formulas such that $\mathcal{F}$ is finitely satisfiable if and only if PRQ($\mathcal{F}$) is finitely satisfiable. If PRQ($\mathcal{F}$) has a finite model $\mathcal{M}$, then a term model of PRQ($\mathcal{F}$) is obtained by a renaming of the elements of the universe of $\mathcal{M}$. ∎

5 Extended Positive Tableaux

Extended Positive tableaux, short EP tableaux, are a refinement of the PUHR tableaux defined in [6] as a formalization of the SATCHMO theorem prover [15]. The refinement consists in the processing of PRQ formulas instead of (Skolemized) clauses, and in a tableau expansion rule for existentially quantified subformulas which, as opposed to the standard δ rule [20, 10], performs no "run time Skolemization".

Example 1. Consider $\mathcal{S} = \{p(a), \forall x(p(x) \rightarrow \exists y p(y))\}$ and a Skolemized version $\mathrm{Sk}(\mathcal{S})$ of $\mathcal{S}$. Applied to $\mathrm{Sk}(\mathcal{S})$, the PUHR tableau method initiates the construction of the infinite model $\{p(a), p(f(a)), p(f(f(a))), \dots\}$. A similar problem arises if the standard δ rule is applied to $\mathcal{S}$: The finite model $\{p(a)\}$ of $\mathcal{S}$ is not detected by the PUHR tableau method.

The expansion rule for existentially quantified subformulas considered below ensure the completeness with respect to finite satisfiability of the EP tableaux method.

Definition 2. (EP Tableaux expansion rules)

$\exists$ *rule:*

$$\frac{\exists x E(x)}{E[c_1/x] \mid \dots \mid E[c_k/x] \mid E[c_{new}/x]}$$

where $\{c_1 \dots c_k\}$ is the set of all constants occurring in the root to node branch, and where c_{new} is a constant distinct from all c_i for $i = 1, \dots, k$.

PUHR rule:	$\vee$ *rule:*	$\wedge$ *rule:*
$\dfrac{\forall \bar{x}(R(\bar{x}) \rightarrow F)}{F[\bar{c}/\bar{x}]}$	$\dfrac{E_1 \vee E_2}{E_1 \mid E_2}$	$\dfrac{E_1 \wedge E_2}{\begin{array}{c} E_1 \\ E_2 \end{array}}$

where $R[\bar{c}/\bar{x}]$ is satisfied by the interpretation specified by the branch.

In the PUHR rule, $\bar{c}$ is a tuple of constants occurring in the root to node branch. These constants are determined by evaluating R against the already constructed interpretation. This evaluation corresponds to an extension of positive unit hyperresolution. It coincides with (standard) positive unit hyperresolution if $R \rightarrow F$ has the form $P_1 \wedge \dots \wedge P_n \rightarrow C_1 \vee \dots C_m$ where the P_i $(i = 1, \dots, n)$ and C_j $(j = 1, \dots, m)$ are atoms. Recall that the notation $\forall \epsilon(R \rightarrow F)$, where ϵ denotes the empty tuple, is allowed and stands for the formula $R \rightarrow F$. Thus, the PUHR rule handles both, universally quantified and implicative formulas.

Definition 3. (EP Tableaux) *If L is a set of formulas, $Atoms(L)$ will denote the set of ground atoms in L. EP Tableaux for a set $\mathcal{S}$ of PRQ formulas are trees whose nodes are sets of closed formulas. They are inductively defined as follows:*

1. *The tree consisting in the single node $\mathcal{S}$ is an EP Tableau for $\mathcal{S}$.*

2. *If T is an EP Tableau for $\mathcal{S}$, L a leaf of T, and φ a formula in L which is not satisfied in the term interpretation $\mathcal{T}(Atoms(L))$, then the tree obtained from T by applying the relevant expansion rule to L with respect to φ is an EP Tableau for $\mathcal{S}$.*

A branch of an EP Tableau is open if it does not contain $\bot$. Otherwise, it is closed. An EP tableau is open if at least one of its branches is open; otherwise, it is closed. If $\mathcal{B}$ is a branch in an EP tableau, then $\cup\mathcal{B}$ denotes the union of the nodes in $\mathcal{B}$. An EP tableau is satisfiable if it has a branch $\mathcal{B}$ such that $\cup\mathcal{B}$ is satisfiable.

Note that the PUHR rule is the only expansion rule which can be applied more than once to a same formula along a branch of an EP tableau. Indeed the condition "which is not satisfied in the term interpretation $\mathcal{T}(Atoms(L))$" prevents repeated applications of rules other than the PUHR rule.

Example 2. An EP Tableau for $\mathcal{S}_1 = \{p(a), \forall x(p(x) \rightarrow r(x) \vee \exists y q(x,y))\}$ is given by Fig. 1.[1] For the sake of readability, the nodes in the figures are not labelled with sets of formulas but with the single formula added at the corresponding node of the represented EP tableau.

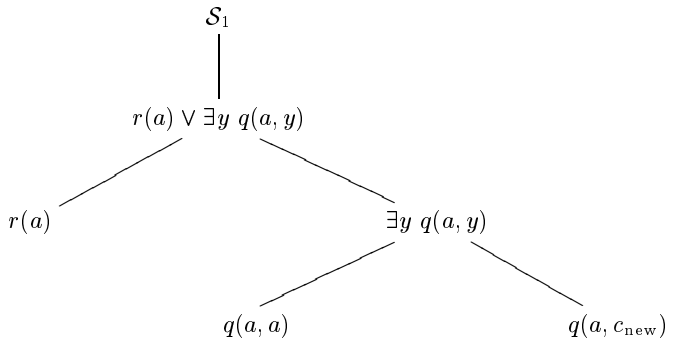

Fig. 1. An EP tableau for Example 2

Example 3. An infinite EP Tableau for $\mathcal{S}_2 = \{empl(c_0),\ \forall x(empl(x) \rightarrow \exists y\ works\text{-}for(x, y)),\ \forall x \forall y(works\text{-}for(x,y) \rightarrow empl(x) \wedge empl(y))\}$ is given by Fig. 2 (the predicates are abbreviated to their first letters).

[1] Strictly, the syntax of Definition 1 would require $\forall x(p(x) \rightarrow r(x) \vee \exists y(q(x,y)\ \underline{\wedge \top}))$.

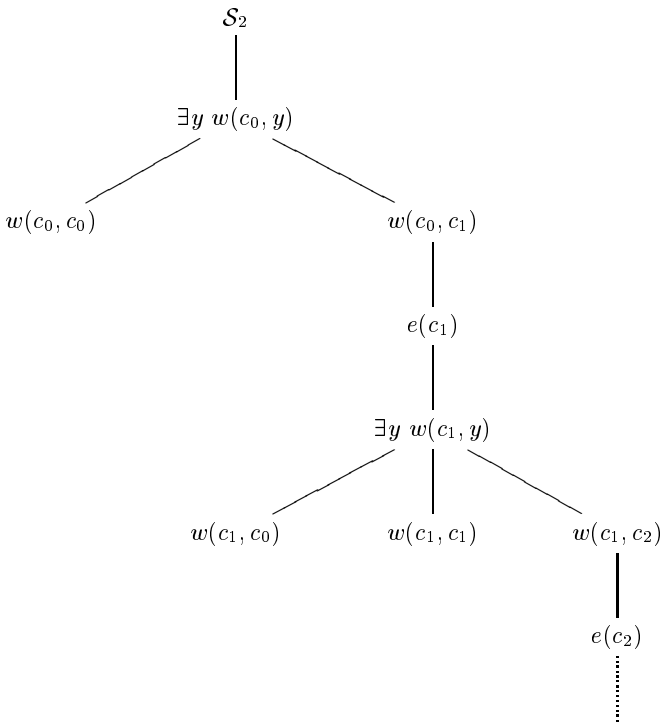

Fig. 2. An EP tableau for Example 3

6 Refutation Soundness and Completeness

The results of this section are standard. They can be established in the usual manner (cf. e.g. [10]).

Lemma 2. *The application of an expansion rule to a satisfiable EP tableau results in a satisfiable EP tableau.*

Proof. (sketched) For every expansion rule, one easily shows that if a node N of an EP tableau is satisfiable, then there is at least one successor of N which is satisfiable. ■

In the following, $\mathcal{S}$ denotes a set of PRQ formulas.

Theorem 2. (Refutation Soundness) *If there exists a closed EP tableau for $\mathcal{S}$, then $\mathcal{S}$ is unsatisfiable.*

Proof. Assume $\mathcal{S}$ is satisfiable. By Lemma 2 there exists no closed EP tableaux for $\mathcal{S}$. ■

The following is a formalization of the standard concept of fairness [10]. Recall that the nodes of an EP tableau are sets of PRQ formulas.

Definition 4.
• *Let T be an EP tableau for $\mathcal{S}$, and $\mathcal{B}$ a branch in T. Then $\cup\mathcal{B}$ is said to be* saturated *if the following holds:*

1. *If $E_1 \vee E_2 \in \cup\mathcal{B}$ then $E_1 \in \cup\mathcal{B}$ or $E_2 \in \cup\mathcal{B}$.*
2. *If $E_1 \wedge E_2 \in \cup\mathcal{B}$ then $E_1 \in \cup\mathcal{B}$ and $E_2 \in \cup\mathcal{B}$.*
3. *If $\exists x E(x) \in \cup\mathcal{B}$ then there is $E[x/c_1] \in \cup\mathcal{B}$, or ..., or $E[x/c_n] \in \cup\mathcal{B}$, or $E[x/c_{new}] \in \cup\mathcal{B}$. $c_1, \ldots, c_n$ are all constants occurring in $\mathcal{B}$ above the node which is expanded by the $\exists$-rule, and c_{new} is a constant, not occurring in the branch.*
4. *If $\forall\bar{x}(R(\bar{x}) \rightarrow F) \in \cup\mathcal{B}$, then for all substitutions σ, such that $\mathcal{T}(Atoms \cup \mathcal{B}) \models R\sigma$, $F\sigma \in \cup\mathcal{B}$.*

• *An EP tableau T is called* fair *if $\cup\mathcal{B}$ is saturated for each open branch $\mathcal{B}$ of T.*

Lemma 3. (Model Soundness) *Let T be an EP tableau for $\mathcal{S}$ and $\mathcal{B}$ an open branch of T. If T is fair, then $\mathcal{T}(Atoms(\cup\mathcal{B})) \models \mathcal{S}$.*

Proof. (sketched) By induction on the structure of PRQ formulas. ∎

From Lemma 3 follows immediately:

Theorem 3. (Refutation Completeness) *If $\mathcal{S}$ is unsatisfiable, then every fair EP tableau for $\mathcal{S}$ is closed.*

Corollary 2. *If $\mathcal{S}$ is not finitely satisfiable, then every open branch of a fair EP tableau for $\mathcal{S}$ is infinite.*

Proof. Assume that $\mathcal{S}$ is not finitely satisfiable. Assume there is a fair EP tableau T with a finite open branch $\mathcal{B}$. By Lemma 3 $\mathcal{T}(Atoms(\cup\mathcal{B}))$ is a finite model of $\mathcal{S}$, a contradiction. ∎

7 Finite Satisfiability Completeness

In the following, it is shown that EP tableaux are complete for finite satisfiability in the sense that they give rise to constructing up to a constant renaming all the finite term models of satisfiable sets of PRQ formulas. The proof of this result is more complex than that of other theorems given in this paper. It makes use of non standard notions, that are first introduced.

Definition 5. (Simple Expansion) *Let $\mathcal{S}$ be a satisfiable set of PRQ formulas, φ an element of $\mathcal{S}$, and $\mathcal{T}(\mathcal{G})$ a term model of $\mathcal{S}$.* Simple expansions $\mathcal{S}'$ of $\mathcal{S}$ with respect to φ and $\mathcal{T}(\mathcal{G})$ *are defined as follows:*

1. *If φ is a ground atom, then $\mathcal{S}' := \mathcal{S}$.*
2. *If $\varphi = \varphi_1 \wedge \varphi_2$, then $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{\varphi_1, \varphi_2\}$.*
3. *If $\varphi = \varphi_1 \vee \varphi_2$, then $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{\varphi_i\}$ for one $i \in \{1, 2\}$ such that $\mathcal{T}(\mathcal{G}) \models \varphi_i$.*[2]
4. *If $\varphi = \exists x \varphi_1$, then $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{\varphi_1[c/x]\}$, where c is a constant, such that $\mathcal{T}(\mathcal{G}) \models \varphi_1[c/x]$.*[3]
5. *If $\varphi = \forall\bar{x}(R(\bar{x}) \rightarrow F)$, then $\mathcal{S}' := (\mathcal{S} \setminus \{\varphi\}) \cup \{F[\bar{x}/\bar{c}] \mid \bar{c}$ a tuple of constants s.t. $\mathcal{T}(\mathcal{G}) \models R[\bar{x}/\bar{c}]\}$.*[4]

[2] Since $\mathcal{T}(\mathcal{G}) \models \mathcal{S}$ and $\varphi \in \mathcal{S}$, $\mathcal{T}(\mathcal{G}) \models \varphi_i$ for at least one of $i = 1, 2$.
[3] Such a constant exists necessarily since $\mathcal{T}(\mathcal{G}) \models \mathcal{S}$ and $\varphi \in \mathcal{S}$.
[4] If there are no constants $\bar{c}$ such that $\mathcal{T}(\mathcal{G}) \models R[\bar{x}/\bar{c}]$, then $\mathcal{S}' := \mathcal{S} \setminus \{\varphi\}$.

Note that for every $\mathcal{S}$, every element φ of $\mathcal{S}$, and every model $\mathcal{T}(\mathcal{G})$ of $\mathcal{S}$, there exists at least one simple expansion of $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{G})$. A simple expansion $\mathcal{S}'$ of $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{G})$ differs from $\mathcal{S}$ whenever φ is nonatomic. The existence of a simple expansion $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{G})$ such that $\mathcal{S} \neq \mathcal{S}'$ does not necessarily mean that some EP tableau expansion rule can be applied to some formula in $\mathcal{S}$. Indeed, according to Definition 3 an expansion rule can only be applied if $\mathcal{T}(Atoms(\mathcal{S})) \not\models \varphi$. Every simple expansion of a finite set $\mathcal{S}$ of PRQ formulas w.r.t a formula and a *finite* model $\mathcal{T}(\mathcal{G})$ of $\mathcal{S}$ is finite. Because of 5. in Definition 5 this is not necessarily the case if $\mathcal{T}(\mathcal{G})$ is infinite.

Lemma 4. *Let $\mathcal{S}$ be a set of PRQ formulas, $\varphi \in \mathcal{S}$, $\mathcal{T}(\mathcal{E})$ a finite, minimal term model of $\mathcal{S}$, and $\mathcal{S}'$ a simple expansion of $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{E})$. $\mathcal{T}(\mathcal{E})$ is a minimal model of $\mathcal{S}'$.*

Proof. (sketched) By a case analysis based on the structure of the PRQ formula φ. ■

Definition 6. (Rank)
• *Let φ be a (non necessarily closed) PRQ formula and d a positive integer. The* d-rank *$rk(\varphi, d)$ of a PRQ formula is inductively defined as follows:*

1. *If φ is an atom, then $rk(\varphi, d) := 0$.*
2. *If $\varphi = \varphi_1 \wedge \varphi_2$, or $\varphi = \varphi_1 \vee \varphi_2$, or $\varphi = \varphi_1 \rightarrow \varphi_2$, then $rk(\varphi, d) := rk(\varphi_1, d) + rk(\varphi_2, d) + 1$.*
3. *If $\varphi = \exists x \psi$, then $rk(\varphi, d) := rk(\psi, d) + 1$.*
4. *If $\varphi = \forall \bar{x} \psi$, then $rk(\varphi, d) := rk(\psi, d) \times d^n$, where n is the size of the tuple $\bar{x}$.*

• *Let $\mathcal{S}$ be a set of PRQ formulas, $\mathcal{T}(\mathcal{E})$ a finite minimal model of $\mathcal{S}$, and d the cardinality of the domain of $\mathcal{T}(\mathcal{E})$. The* rank *$rk(\mathcal{S}, \mathcal{T}(\mathcal{E}))$ of $\mathcal{S}$ with respect to $\mathcal{T}(\mathcal{E})$ is defined by $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) := \sum_{\psi \in \mathcal{S}} rk(\psi, d)$ if $Atoms(\mathcal{S}) \subset \mathcal{E}$ and $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) := 0$ if $Atoms(\mathcal{S}) = \mathcal{E}$.*

Note that $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) = 0$ if and only if $\mathcal{T}(Atoms(\mathcal{S}))$ is a model of $\mathcal{S}$. In other words $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) > 0$ if and only if some EP tableau expansion rule can be applied to some formula in $\mathcal{S}$.

Lemma 5. *Let $\mathcal{S}$ be a finitely satisfiable set of PRQ formulas, $\varphi \in \mathcal{S}$, φ nonatomic, $\mathcal{T}(\mathcal{E})$ a finite minimal model of $\mathcal{S}$, and $\mathcal{S}'$ a simple expansion of $\mathcal{S}$ wrt φ and $\mathcal{T}(\mathcal{E})$. If $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) \neq 0$, then $rk(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rk(\mathcal{S}, \mathcal{T}(\mathcal{E}))$.*

Proof. (sketched) By a case analysis based on the structure of the PRQ formula φ. ■

Theorem 4. (Completeness for Finite Satisfiability) *Let $\mathcal{T}(\mathcal{E})$ be a finite term model of $\mathcal{S}$. If $\mathcal{T}(\mathcal{E})$ is a minimal model of $\mathcal{S}$, then every fair EP tableau for $\mathcal{S}$ has a finite, open branch $\mathcal{B}$ such that, up to a renaming of constants, $Atoms(\cup \mathcal{B}) = \mathcal{E}$.*

The proof is based on a double induction. This is needed since the PUHR rule can repeatedly be applied to a same formula along a same branch. As a consequence, a measure of the syntactical complexity of the set of formulas, which would be a natural induction parameter, does not decrease after an application of the PUHR rule. This is overcome by a second induction on the number of applications of the PUHR rule to a same formula.

Proof. Let $\mathcal{T}(\mathcal{E})$ be a finite, minimal term model of $\mathcal{S}$. The proof is by induction on $rk(\mathcal{S}, \mathcal{T}(\mathcal{E}))$. Induction hypothesis:

> ($\star$) If $\mathcal{M}$ is a set of PRQ formulas, if $\mathcal{T}(\mathcal{F})$ is a finite, minimal term model of $\mathcal{M}$, and if $rk(\mathcal{M}, \mathcal{T}(\mathcal{F})) < n$, then every fair EP tableau for $\mathcal{M}$ has a finite, open branch $\mathcal{B}$ such that, up to a renaming of constants, $Atoms(\cup\mathcal{B}) = \mathcal{F}$.

Assume that $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) = 0$. $\mathcal{S}$ has therefore a single minimal term model, namely $\mathcal{T}(Atoms(\mathcal{S}))$, and every fair EP tableau for $\mathcal{S}$ consists in one single node equal to $\mathcal{S}$. Clearly, the result holds.
Assume that $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) = n > 0$.
Let T be a fair EP tableau for $\mathcal{S}$. Since $\mathcal{S}$ is satisfiable, by Theorem 2 T is open. Since $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) > 0$ there exists at least one formula $\varphi \in \mathcal{S}$ on which an expansion rule can be applied. Since T is fair, its root necessarily has successor(s). Let $\varphi \in \mathcal{S}$ be the formula on which the application of an expansion rule yields the successor(s) of the root of T.
Case 1: $\varphi = \varphi_1 \wedge \varphi_2$, or $\varphi = \varphi_1 \vee \varphi_2$, or $\varphi = \exists x\psi$. By Definition 5 there is at least one successor N of the root such that $N = \{\varphi\} \cup \mathcal{S}'$ where, up to constant renaming in case $\varphi = \exists x\psi$, $\mathcal{S}'$ is a simple expansion of $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{E})$. Since an EP tableau expansion rule cannot be applied more than once to a formula like φ, the tableau rooted at N is an EP tableau T' for the simple expansion $\mathcal{S}'$. T' is fair because so is T. For every simple expansion $\mathcal{S}'$ of $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{E})$, by Lemma 4, $\mathcal{T}(\mathcal{E})$ is a minimal model of $\mathcal{S}'$. Since $rk(\mathcal{S}, \mathcal{T}(\mathcal{E})) > 0$ and φ is nonatomic, by Lemma 5 $rk(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rk(\mathcal{S}, \mathcal{T}(\mathcal{E}))$. Therefore, by induction hypothesis ($\star$) , the tableau rooted at N has a finite open branch $\mathcal{B}'$ such that, up to a renaming of constants, $Atoms(\cup\mathcal{B}') = \mathcal{E}$. Hence, the same holds of T.
Case 2: $\varphi = \forall\bar{x}(R(\bar{x}) \rightarrow F)$. Let $\mathcal{S}'$ be the (unique) simple expansion of $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{E})$. Along a branch of the fair EP tableau T for $\mathcal{S}$, the PUHR rule is possibly applied more than once to φ. Therefore, the tree rooted at the successor N of the root of T is not necessarily an EP tableau for $\mathcal{S}'$. In the following it is shown how parts of T can be regarded as parts of an EP tableau for $\mathcal{S}'$. For $n \in \mathbb{N}$ and a branch $\mathcal{B}$ of T, let $\mathcal{B}^n$ denote the prefix of $\mathcal{B}$ up till (and without) the $(n+1)$-th application of the PUHR rule on φ, if the PUHR rule is applied more than n times to φ in $\mathcal{B}$; otherwise, let $\mathcal{B}^n := \mathcal{B}$. The following is first established by induction on n: For all $n \in \mathbb{N} \setminus \{0\}$,

> ($\star\star$) T has a branch $\mathcal{B}$ such that, up to a renaming of constants, $Atoms(\mathcal{B}^n) \subseteq \mathcal{E}$.

Case 2.1: $n = 1$: The successor N of the root of T results from an application of the PUHR rule to $\varphi = \forall\bar{x}(R(\bar{x}) \rightarrow F)$, i.e., by Definition 2 and 3, there is a set $\mathcal{G}$ of ground atoms and a substitution σ such that $\mathcal{G} \subset \mathcal{S}$, $\mathcal{T}(\mathcal{G}) \models R\sigma$, and $\mathcal{T}(\mathcal{G}) \not\models F\sigma$, and $N = \mathcal{S} \cup \{F\sigma\}$. Since by hypothesis $\mathcal{T}(\mathcal{E})$ is a model of $\mathcal{S}$, $\mathcal{G} \subset \mathcal{E}$ and by Lemma 1 $\mathcal{T}(\mathcal{E}) \models R\sigma$. Furthermore, since $\mathcal{T}(\mathcal{E}) \models \varphi$, $\mathcal{T}(\mathcal{E}) \models F\sigma$. Since $\mathcal{S}'$ is by hypothesis the (unique) simple expansion of $\mathcal{S}$ w.r.t. φ and $\mathcal{T}(\mathcal{E})$, $F\sigma \in \mathcal{S}'$. So, there is an EP-tableau T' for $\mathcal{S}'$, which coincides with T from N until the second application of the PUHR rule on φ in all branches. Since by Lemma 4 $\mathcal{T}(\mathcal{E})$ is a minimal model of $\mathcal{S}'$ and since by Lemma 5 $rk(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rk(\mathcal{S}, \mathcal{T}(\mathcal{E}))$, the induction hypothesis ($\star$) is applicable:

There is a branch $\mathcal{B}'$ in T' with, up to constant renaming, $Atoms(\cup\mathcal{B}') = \mathcal{E}$. So, for the corresponding branch $\mathcal{B}$ in T $Atoms(\mathcal{B}^1) \subseteq \mathcal{E}$.
Case 2.2: $n > 1$: Assume that $(\star\star)$ holds for all $m \leq n$. Let $\mathcal{B}_1, \ldots, \mathcal{B}_k$ be all such branches of T.
If for some $i = 1, \ldots, k$ $\mathcal{B}_i^n = \mathcal{B}_i^{n+1} = \mathcal{B}_i$, i.e. the PUHR rule is applied at most n times to φ along $\mathcal{B}_i$, then by induction hypothesis $(\star\star)$ $Atoms(\mathcal{B}^{n+1}) \subseteq \mathcal{E}$.
Otherwise, since by induction hypothesis $(\star\star)$ $Atoms(\mathcal{B}^n) \subseteq \mathcal{E}$, by Lemma 1 and by definition of $\mathcal{S}'$ each formula $F\sigma_i$ resulting from an $(n+1)$-th application of the PUHR rule to φ in the branch $\mathcal{B}_i$ is in $\mathcal{S}'$. Therefore, an EP tableau T' for $\mathcal{S}'$ can be constructed from the subtree of T rooted at N as follows: First, replace N by $\mathcal{S}'$. Second, keep from each branch $\mathcal{B}_i$ only the prefix $\mathcal{B}_i^{n+1}$. Third, remove from each $\mathcal{B}_i^{n+1}$ those nodes resulting from applications of the PUHR rule to φ. Fourth, cut all other branches immediately before the first application of the PUHR rule to φ. T' is a finite EP tableau for $\mathcal{S}'$, which is not necessarily fair. Since T' is finite, a fair EP tableau T'' for $\mathcal{S}'$ can be obtained by further expanding T'. By Lemma 5, $rk(\mathcal{S}', \mathcal{T}(\mathcal{E})) < rk(\mathcal{S}, \mathcal{T}(\mathcal{E}))$. By induction hypothesis $(\star)$, T'' has a branch $\mathcal{B}'$ with, up to constant renaming, $Atoms(\cup\mathcal{B}') = \mathcal{E}$. By definition of $\mathcal{B}_1, \ldots, \mathcal{B}_k$ and T'' there is a branch $\mathcal{B}_i$ in T such that $Atoms(\mathcal{B}_i^{n+1}) = Atoms(\mathcal{B}'^{n+1}_i)$. Hence, $Atoms(\mathcal{B}_i^{n+1}) \subseteq \mathcal{E}$.

Since by hypothesis $\mathcal{T}(\mathcal{E})$ is finite, T has a finite branch $\mathcal{B}$ for which $(\star\star)$ holds. Hence, this branch is open. Since T is fair, by Lemma 3 $\mathcal{T}(Atoms(\cup\mathcal{B})) \models \mathcal{S}$ and since $\mathcal{T}(\mathcal{E})$ is minimal, up to a renaming of constants, $Atoms(\cup\mathcal{B}) = \mathcal{E}$. ∎

8 Implementation

A Prolog program, called FINFIMO (FINd all FInite MOdels), implements a depth-first expansion of EP tableaux, cf.: http://www.pms.informatik.uni-muenchen.de/software

Since the extensions ensuring the completeness for finite satisfiability do not compromise range restriction and the "positive preference" of PUHR tableaux, a concise implementation in the style of [15] is possible. For space reasons, this implementation cannot be commented here. First experiments point to a reasonable efficiency. The system SIC [4] for assisting in the design of database integrity constraints relies on FINFIMO.

9 Related Work

The method described in the present paper is related to approaches of three kinds: (1) Generators of models of (or up to) a given cardinality, (2) tableaux methods complete for finite satisfiability, and (3) generators of finitely representable models.

One of the best known generator of finite models of (or up to) a given cardinality is FINDER [19]. Its strength lies in a sophisticated, very efficient implementation of the exhaustive search for models up to a given cardinality. Most generators of finite models up to a given cardinality can continue the search with a higher cardinality, if no models of the formerly given cardinality can be found. However they always require an upper bound for the cardinality of the models seeked for. For the applications mentioned in

Section 2, this might be too strong a requirement. In this respect, the EP tableaux method presented in the present paper is more flexible. This flexibility is exploited in the applicatiioon described in [4].

Tableaux methods complete for finite satisfiability that process existentially quantified formulas like described here have been proposed in [5, 12, 14]. They all replace the δ rule of classical tableaux [20, 10] by an expansion rule like the $\exists$ rule of Section 5. The approach described in the present paper differs in the use of the PUHR (positive unit hyper-resolution) rule which relies on resolution for avoiding the "blind instantiation" and the resulting inefficiency of the classical γ rule. In the implementation of the procedure proposed by [14] this problem is resolved by giving a limit on the number of γ expansion for each γ formula. In practice, it is however not always easy to set such an upper bound. A further interest of the approach presented here is its short and easily adaptable implementation. This is useful for practical applications such as that described in [4].

Other extensions and refinements of tableau methods generate finite representation for (possibly infinite) models [8, 21, 9, 16]. In [16] a method for extracting models of (possibly infinite) branches by means of equational constraints is described. The approaches [21, 9] are based on resolution and therefore are much more efficient than approaches based on the δ rule of classical tableaux methods. In contrast to the method described in the present paper, the method described in [21] only applies to the monadic and Ackermann class. The method of [9] which, like the PUHR and EP tableaux, is based on positive hyper-resolution, avoids splitting. In some cases, this results in gains in efficiency. For most applications mentioned in Section 2, in particular for the database problems, *finitely representable* models are not convenient. Instead *finite* models are needed. This motivated the research reported in the present paper.

10 Conclusion and Perspectives

Some applications of theorem proving have been discussed that can benefit from a model generator complete for finite satisfiability not imposing an upper bound on the size of the models searched for. A novel approach *Extended Positive Tableaux* has been developed for such applications. Like the PUHR Tableaux [6] they extend, Extended Positive Tableaux rely on positive unit hyper-resolution and "range restriction" for avoiding the "blind instantiation" performed by the γ rule of standard tableaux [20, 10, 22, 23]. Instead of relying on Skolemization, as most refutation methods do, Extended Positive Tableaux use the extended δ rule of [5, 12, 14]. It was shown that this rule makes the Extended Positive Tableaux method complete not only for refutation, like standard tableaux methods, but also for finite satisfiability. A prototype written in Prolog in the style of SATCHMO [15] implements the Extended Positive Tableaux method.

The following issues deserve further investigations.

First, for most applications it would be desirable to have typed variables. An extension based on a simple type system and many-sorted logic seem sufficient for the application described in [4].

Second, the method remains to be extended to languages with function symbols. Even though the restriction to languages without function symbols is not stringent since existential quantifiers are allowed, explicit function symbols would be more convenient for applications. To handle explicit function symbols a promising direction could be the use of constraint reasoning techniques in the manner of [1].

Third, for applications such as diagnosis and the database issues mentioned in Section 2, it would be preferable to have a method not only complete for finite satisfiability, but also which generates only minimal models, as investigated e.g. in [6].

References

[1] S. Abdennadher and H. Schütz. Model Generation with Existentially Quantified Variables and Constraints. In *Proc. Sixth International Conference on Algebraic and Logic Programming*, Springer LNCS 1298, 1997.

[2] F. Bry. Intensional Updates: Abduction via Deduction. In *Proc. 7th Int. Conf. on Logic Programming*, MIT Press, 561-575, 1990

[3] F. Bry, H. Decker, and R. Manthey. A Uniform Approach to Constraint Satisfaction and Constraint Satisfiability in Deductive Databases. In *Proc. 1st Int. Conf. Extending Data Base Technology*, Springer LNCS 303, 1988

[4] F. Bry, N. Eisinger, H. Schütz, and S. Torge. SIC: An Interactive Tool for the Design of Integrity Constraints (System Description). Res. rep. PMS-FB-1997-15, Institut für Informatik, Universität München, 1997, submitted for publication.

[5] F. Bry and R. Manthey. Proving Finite Satisfiability of Deductive Databases. In *Proc. 1st Workshop on Computer Science Logic*, Karlsruhe, Germany, Springer LNCS 329, 44-55, 1987

[6] F. Bry and A. Yahya. Minimal Model Generation with Positive Unit Hyperresolution Tableaux. In *Proc. 5th Workshop on Theorem Proving with Tableaux and Related Methods*, Springer LNAI 1071, 1996

[7] F. Bry and A. Yahya. Positive Unit Hyper-Resolution Tableaux for Minimal Model Generation. Res. rep. PMS-FB-1997-8, Institut für Informatik, Universität München, 1997, complete version of [6], submitted for publication

[8] R. Caferra and N. Zabel. Building Models by Using Tableaux Extended by Equational Problems. In *J. of Logic and Computation*, 3:3-25, 1993

[9] C. Fermüller and A. Leitsch. Hyperresolution and Automated Model Building. In *J. of Logic and Computation*, 6(2):173-203, 1996

[10] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990

[11] H. Gallaire, J. Minker, and J.-M. Nicolas. Logic and Databases: A Deductive Approach. In *ACM Computing Surveys*, Vol. 16, Nr. 2, 1984

[12] J. Hintikka. Model Minimization - An Alternative to Circumscription. *J. of Automated Reasoning*, Vol. 4, 1988

[13] M. Kettner and N. Eisinger. The Tableau Browser SNARKS (System Description). In *Proc. 14th Int. Conf. on Automated Deduction*, Springer LNAI 1249, 1997

[14] S. Lorenz. A Tableau Prover for Domain Minimization. In *J. of Automated Reasoning*, Vol. 13, 1994

[15] R. Manthey and F. Bry. SATCHMO: A Theorem Prover Implemented in Prolog. In *Proc. 9th Int. Conf. on Automated Deduction*, Springer LNAI 310, 1988

[16] N. Peltier. Simplifying and Generalizing Formulae in Tableaux. Pruning the Search Space and Building Models. In *Proc. 6th Workshop on Theorem Proving with Tableaux and Related Methods*, Springer LNAI 1227, 1997

[17] D. Poole. Normality and Faults in Logic-Based Diagnosis. In *Proc. 11th Int. Joint Conf. on Artificial Intelligence*, 1304–1310, 1985

[18] R. Reiter. A Theory of Diagnosis from First Principles. In *Artificial Intelligence*, Vol. 32, 57–95, 1987

[19] J. Slaney. Finder (finite domain enumerator): Notes and Guides. Tech. rep., Australian National University Automated Reasoning Project, Canberra, 1992

[20] R. Smullyan. *First-Order Logic*. Springer, 1968

[21] T. Tammet. Using Resolution for Deciding Solvable Classes and Building Finite Models. In *Baltic Computer Science*, Springer LNCS 502, 1991

[22] G. Wrightson. ed. Special Issue on Automated Reasoning with Analytic Tableaux, Part I. *J. of Automated Reasoning*, Vol. 13 No. 2, 173–281, 1994

[23] G. Wrightson, ed. Special Issue on Automated Reasoning with Analytic Tableaux, Part II. *J. of Automated Reasoning*, Vol. 13 No. 3, 283–421, 1994