A Deduction Method Complete for Refutation and Finite Satisfiability

François Bry and Sunna Torge

Computer science Institute, University of Munich, Germany
http://www.pms.informatik.uni-muenchen.de/
{bry,torge}@informatik.uni-muenchen.de

Abstract. Database and Artificial Intelligence applications are briefly discussed and it is argued that they need deduction methods that are not only refutation complete but also complete for finite satisfiability. A novel deduction method is introduced for such applications. Instead of relying on Skolemization, as most refutation methods do, the proposed method processes existential quantifiers in a special manner which makes it complete not only for refutation, but also for finite satisfiability. A main contribution of this paper is the proof of these results.
Keywords: Artificial Intelligence, Expert Systems, Databases, Automated Reasoning, Finite Satisfiability.

1 Introduction

For many applications of automated reasoning, the tableaux methods [32, 16, 34, 18] 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. However, for most applications these 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 ones exist.

In this paper, a novel approach is formally introduced, which aims at overcoming these drawbacks. Like the approach [25, 11] it refines and extends, this method relies on resolution and "range restriction" for avoiding the "blind instantiation" performed by the γ rule [32, 16]. Thanks to range restriction, the proposed method can represent interpretations as sets 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, the proposed method uses the extended δ rule, also called δ^* rule, proposed in [10, 20, 23]. This rule makes the method complete not only for refutation, but also for finite satisfiability. A prototype written in Prolog implements the proposed method and a refinement of it is used in a database application [8, 9]. There, the method described in the present paper is informally recalled, but neither formally stated, nor proved to be sound and complete. A main contribution of the present paper is the proof of these results.

2 Applications

In several application areas, specific techniques have been developed that can be expressed as a systematic search for models of first-order logic specifications.

Diagnosis. The approach to diagnosis described in [28] 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. Tableaux methods are very convenient for this purpose like described e.g. in [4]. Diagnosis in fact requires to seek for models that are as small as possible [5], 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 [7, 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 [19]. 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: populating the database will be possible if and only if its schema has a nonempty and finite model. The system described in [8, 9] gives rise to verify this.

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.

Natural Language Understanding. Interpreting natural language sentences is often performed by generating the possible models of a logic representation of the considered sentences. Consider e.g. the sentences "*Anna sees a dog. So does Barbara.*" that can be expressed by $\exists x\ sees(Anna, x) \wedge dog(x)$ and $\exists y\ sees(Barbara, y) \wedge dog(y)$. Skolemization as well as the δ rule of standard tableaux methods would only generate one model with two distinct dogs. In contrast, thanks to the extended δ or δ^* rule, the method described in the present paper would more conveniently generate both, a model with a single dog and a model with two distinct dogs [14].

In the above mentioned applications, the models seeked for must be finite.

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. This is possible, since finite satisfiability is semidecidable [36]. For as well detecting unsatisfiability, one can rely on a refutation prover coupled with a finite-model finder such as FINDER [31] and SEM [37], as described e.g. in [30]. The present paper proposes instead a *single* deduction method complete for both, unsatisfiability and finite satisfiability. Arguably, this method is more convenient for many applications. In particular, a single method is better amenable to user interaction and explanation as provided by the system [8, 9]. For some applications - e.g. natural language understanding - a single method is necessary [14] and the coupling approach of [30] is not applicable.

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" in the sense of e.g. [11]. 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.) [16]. The domain of a term interpretation $\mathcal{I}$ consists in all ground terms, here constants, occurring in the ground atoms satisfied by $\mathcal{I}$ augmented with an additional, arbitrary constant c_0 occurring in no ground atoms satisfied by

$\mathcal{I}$. In contrast, the domain of an Herbrand interpretation consists in all possible ground terms constructable by constants and function symbols in the considered language. The mapping of an interpretation that assigns to every constant (resp. n-ary relation symbol) an element of the domain (resp. a n-ary relation over the domain) will be called *assignment*.

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 the special constant c_0 which without loss of generality can further be assumed not to occur in $\mathcal{S}$. 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 $\perp$ and $\top$ that respectively evaluate to false and true in all interpretations. A negated formula $\neg F$ will always be treated as the implication $F \rightarrow \perp$. 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 $\{c_1/x_1, \dots, c_n/x_n\}$.

In the following familiarity with tableaux methods as introduced in e.g. [32, 16, 18] is assumed.

4 Positive Formulas with Restricted Quantifications

In this section, a fragment of first-order logic, that of "positive formulas with restricted quantifications" (short PRQ formulas), is introduced. Arguably, this fragment is convenient for applications. It is shown to have the same expressive power as full first-order logic. The intuition of PRQ formulas is that of so-called "restricted quantification" in natural language. The first time an object is referred to in a formula, i.e., when a variable is quantified, a positive expression called "range" specifies which kind of object is meant, like e.g. in $\forall x\ (\underline{employee(x)} \rightarrow \exists y\ (\underline{boss(y)} \wedge works\_for(x,y)))$. The underlined expressions are ranges for x and y, respectively. Note also the use of an implication (resp. conjunction) for introducing the range of a universally (resp. existentially) quantified variable. Ranges and PRQ formulas are defined relying on auxiliary notions that are first introduced.

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

1. *Atoms except $\perp$ 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* $\perp$ *and* $\top$*) are PRQ formulas.*
2. *Conjunctions and disjunctions of PRQ formulas are PRQ formulas.*
3. *A formula of the form* $P \rightarrow F$ *is a PRQ formula if* P *is a positive condition and* F *a PRQ formula.*
4. *A formula of the form* $\forall x_1 \dots x_n\ (R \rightarrow F)$ $(n \geq 1)$ *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.*

Example 1. The formula $F = \forall xy\ (p(x) \wedge q(y) \vee r(x,y) \rightarrow \exists z\ (s(z) \wedge t(z)))$ is a PRQ formula, because $p(x) \wedge q(y) \vee r(x,y)$ is a range for x and y and $s(z)$ is a range for z. The formula $G = \forall xy\ (p(x) \vee q(y) \rightarrow s(y))$ in contrast is not a PRQ formula, since the premise of the implication is not a range for x and y.

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. ∎

For most applications the restriction to PRQ formulas is not a severe restriction since in general quantifications in natural languages are "restricted" through – implicit or explicit – sorts. Furthermore 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 $m' \mid_\Sigma$ is 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 literals 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, $C = \perp$ 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 $\mathcal{R}$ to the formulas in $\mathcal{F}''$: $\mathcal{R}(\forall x F) := \forall x(D(x) \rightarrow \mathcal{R}(F))$, $\mathcal{R}(\exists x F) := \exists x(D(x) \wedge \mathcal{R}(F))$, and $\mathcal{R}(F) := F$ if F is not a quantified formula. One easily verifies that PRQ($\mathcal{F}$) $:= \mathcal{F}' \cup \mathcal{C}(\mathcal{F})$ fulfills the condition of Theorem 1, where $\mathcal{C}(\mathcal{F}) := \{D(c) \mid c$ constant occurring in $\mathcal{F}\}$ if some constants occur in $\mathcal{F}$, $\mathcal{C}(\mathcal{F}) := \{D(c_0)\}$ for some arbitrary constant c_0, otherwise. ■

The predicate D of Theorem 1 generalizes the domain predicate *dom* of [25]. Note that other, more sophisticated transformations of first-order formulas into PRQ formulas than that used in the previous proof are possible which, for efficiency reasons, are more convenient in practice. For space reasons, they are not discussed here.

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 finite 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 [11] as a formalization of the SATCHMO theorem prover [25]. Other related formalizations are given in [13, 3]. The refinement of EP tableaux consists of the processing of PRQ formulas instead of (Skolemized) clauses, and in a tableaux expansion rule for existentially quantified subformulas which, as opposed to the standard δ rule [32, 16], performs no "run time Skolemization".

Example 2. Consider $\mathcal{S} = \{p(a), \forall x(p(x) \rightarrow \exists y p(y))\}$[1] and a Skolemized version $\mathrm{Sk}(\mathcal{S})$ of $\mathcal{S}$. Applied to $\mathrm{Sk}(\mathcal{S})$, the PUHR tableaux 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 tableaux method.

Definition 2. (EP Tableaux expansion rules)

$\exists$ (or δ^*) 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 expanded node, and where c_{new} is a constant distinct from all c_i for $i = 1, \dots, k$.

PUHR rule: $\qquad$ *$\vee$ rule:* $\qquad$ *$\wedge$ rule:*

$$\frac{\forall \bar{x}(R(\bar{x}) \rightarrow F)}{F[\bar{c}/\bar{x}]} \qquad \frac{E_1 \vee E_2}{E_1 \mid E_2} \qquad \frac{E_1 \wedge E_2}{\begin{array}{c} E_1 \\ E_2 \end{array}}$$

where $R[\bar{c}/\bar{x}]$ is satisfied by the interpretation specified by the expanded node, i.e., by the set of ground atoms occurring in that node.

In the PUHR rule, $\bar{c}$ is a tuple of constants occurring in the expanded node. These constants are determined by evaluating R against the already constructed interpretation, i.e., the term interpretation determined by the set of ground atoms occurring in the node. 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 $\mathcal{S}$ is a set of formulas, $Atoms(\mathcal{S})$ will denote the set of ground atoms in $\mathcal{S}$. 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 of the single node $\mathcal{S}$ is an EP Tableau for $\mathcal{S}$.*
2. *Let T be an EP tableau for $\mathcal{S}$, L a leaf of T, and φ a formula in L that is not satisfied in the term interpretation $\mathcal{T}(Atoms(L))$. Then the tree obtained from T by appending one or more children to L according to the expansion rule applicable to φ is an EP tableau for $\mathcal{S}$. Each child of L consists of L and one more formula or two more formulas in the case of φ being a conjunction.*

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

A branch of an EP Tableau is closed *if it contains $\perp$. Otherwise, it is* open. *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 the same formula along a branch of an EP tableau. Indeed the condition "that is not satisfied in the term interpretation $\mathcal{T}(Atoms(L))$" prevents repeated applications of rules other than the PUHR rule. Note also that, although only *finite* EP tableaux can be constructed in *finite* time, Definition 3 does not preclude infinite EP tableaux.

Example 3. Let $\mathcal{S}_1 = \{p(a), \forall x\ (p(x) \rightarrow r(x) \vee \exists y\ q(x,y))\}$.[2] The following table denotes a EP tableau for $\mathcal{S}_1$ in the manner of [6] (to which the denomination "tableaux method" goes back): Successor nodes are right from their parent nodes, branching is represented vertically, and the nodes are not labelled with sets of formulas but with the single formula added at the corresponding node.

$$\begin{array}{llll}
\mathcal{S}_1 & r(a) \vee \exists y q(a,y) & r(a) & \\
 & & \exists y q(a,y) & q(a,a) \\
 & & & q(a, c_{\mathrm{new}})
\end{array}$$

Example 4. Let $\mathcal{S}_2 = \{empl(c_0), \forall x\ (empl(x) \rightarrow \exists y\ works\text{-}for(x,y)), \forall xy\ (works\text{-}\mathrm{for}(x,y) \rightarrow empl(x) \wedge empl(y))\}$[3] The following denotes an infinite EP tableau for $\mathcal{S}_2$ (the predicates are abbreviated to their first letters):

$$\begin{array}{lllllll}
\mathcal{S}_2 & \exists y w(c_0,y) & w(c_0,c_0) & & & & \\
 & & w(c_0,c_1) & e(c_1) & \exists y w(c_1,y) & w(c_1,c_0) & \\
 & & & & & w(c_1,c_1) & \\
 & & & & & w(c_1,c_2) & e(c_2)\ \ldots
\end{array}$$

6 Refutation Soundness and Completeness

The results of this section are established using standard techniques (cf. e.g. [16, 11, 18]). In the following, $\mathcal{S}$ denotes a set of PRQ formulas.

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. ∎

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

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

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

The following definition formalises the standard concept of fairness cf. e.g. [16]. 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 $\ldots$, 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 the node that is expanded by the $\exists$-rule, and c_{new} is a constant, not occurring in this node.*
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. ∎

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

Proof. Immediate consequence of Lemma 3. ∎

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

The proof of the completeness for finite satisfiability of EP tableaux is more sophisticated than that of other theorems given in the previous sections. It makes use of nonstandard 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* $\mathcal{T}(\mathcal{G}) \models \varphi_i$.[4]
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* $\mathcal{T}(\mathcal{G}) \models \varphi_1[c/x]$.[5]
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.* $\mathcal{T}(\mathcal{G}) \models R[\bar{x}/\bar{c}]\}$.[6]

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 tableaux expansion rule can be applied to φ. 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 φ. ∎

Definition 6. (Rank)
• *Let* φ *be a (nonnecessarily 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* $\mathrm{Atoms}(\mathcal{S}) \subset \mathcal{E}$ *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 $\mathcal{S}$.

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

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 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 the same formula along the 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 the 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 $\mathcal{B}$ s. t. 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 of 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 $\mathcal{B}$ s.t. 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 ($\star\star$) $Atoms(\mathcal{B}_i^{n+1}) \subseteq \mathcal{E}$.
Otherwise, since by induction hypothesis ($\star\star$) $Atoms(\mathcal{B}_i^n) \subseteq \mathcal{E}$, 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 $Atoms(\mathcal{B}_i^{n+1}) = Atoms({\mathcal{B}'}_i^{n+1})$. 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, $Atoms(\cup\mathcal{B}) = \mathcal{E}$. ∎

It follows from Theorem 4 that a breadth-first expansion of EP tableaux is complete for finite satisfiability. A depth-first expansion of EP tableaux is not complete for finite satisfiability, as Example 5 shows. However, by theorem 3, a depth-first expansion of EP tableaux is complete for unsatisfiability.

Example 5. Consider the following PRQ formulas: $F_1 = s(a,b)$, $F_2 = \forall xy\ (s(x,y) \rightarrow \exists z\ s(y,z))$, $F_3 = \forall xyz\ (s(x,y) \land s(y,z) \rightarrow s(x,z))$, $F_4 = \forall xy\ (s(x,y) \land s(y,x) \rightarrow \bot)$. Let $\mathcal{S}_3 = \{F_1, F_2, F_3, F_4\}$. The models of $\mathcal{S}_3$ are infinite, as the following EP tableau shows:

$$
\begin{array}{llllllll}
\mathcal{S}_3 & \exists z\ s(b,z)) & s(b,a) & \bot & & & & \\
 & & s(b,b) & \bot & & & & \\
 & & s(b,c_1) & s(a,c_1) & \exists z\ s(c_1,z)) & s(c_1,a) & \bot & \\
 & & & & & s(c_1,b) & \bot & \\
 & & & & & s(c_1,c_1) & \bot & \\
 & & & & & s(c_1,c_2) & \exists z\ s(c_2,z)) & \ldots
\end{array}
$$

Consider now $G = (F_1 \land F_2 \land F_3 \land F_4) \lor p$ and $\mathcal{S}_4 = \{G\}$. A depth-first, leftmost expansion of an EP tableau for $\mathcal{S}_4$ first starts the construction of an EP tableau for the subformula $(F_1 \land F_2 \land F_3 \land F_4)$, i.e. of an infinite EP tableau similar to an EP tableau for $\mathcal{S}_3$, and thus never expands the finite EP tableau for the subformula p.

8 Implementation

A concise Prolog program, called FINFIMO (FINd all FInite MOdels), in the style of [25] implements a depth-first expansion of EP tableaux. For space reasons, this implementation is not commented here. It is given in [9] and available at: http://www.pms.informatik.uni-muenchen.de/software/finfimo/ . First experiments as well as the database application presented in [8, 9], which is based on an implementation of a breadth-first expansion of EP tableaux [8, 9], point to an efficiency that is fully acceptable for the applications mentioned in Section 2.

9 Related Work

EP tableaux are related to approaches of different kinds: **1** Generators of models of (or up to) a given cardinality, **2** coupling of generators of models of (or up to) a given cardinality with a refutation prover, **3** tableaux methods that rely on the extended δ or δ^* rule first introduced in [10], **4** tableaux methods that make use of the γ rule instead of a PUHR rule, and **5** generators of finitely representable models.

1 Possibly, one of the first generator of models of a given cardinality has been described in [21]. Nowadays, among the best known generator of finite models of (or up to) a given cardinality are FINDER [31] and SEM [37]. Their 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 require an upper bound for the cardinality of the models seeked for. For the applications mentioned in Section 2, e.g. the application described in [8], this might be too strong a requirement. Moreover, model generators for given cardinalities such as FINDER and SEM cannot detect unsatisfiability.

2 Model generators for given cardinalities such as FINDER [31] and SEM [37] can be coupled with a refutation prover as described e.g. in [30], resulting in a system complete for both finite satisfiability and unsatisfiability. Arguably, this approach is less convenient for many applications than the approach described here. In particular, for natural language understanding [14] and for an interactive system as described in [8, 9] a single deduction system as proposed in the present paper seems preferable.
3 The extended δ (or δ^*) rule used in EP tableaux has been proposed in former studies. It has been mentioned several times, to the best of our knowledge first in [10, 20, 23]. It is by no means always the case that this rule results in a loss of efficiency for refutation reasoning, for it sometimes gives rise to sooner detection of (finite) satisfiability. This is e.g. the case with the formula $p(a) \wedge \forall x\ (p(x) \rightarrow \exists y p(y))$. Although it has a finite model with only one satisfied p fact, most refutation provers as well as tableaux methods using the standard δ rule expand an infinite model. Such an example is by no means unlikely in applications. An "unprovability proof system" is proposed in [35] which is complete for finite falsifiability.[7] Although it relies on two rules that remind of the extended δ or δ^* rule, it is not complete for both, unsatisfiability and finite falsifiability. It is suggested in [35] to couple it with a refutation method to achieve completeness for both properties.
4 The approach described in the present paper differs from most tableaux methods in the use of the PUHR (positive unit hyperresolution) rule, whose introduction in a tableau method has been first proposed in [25]. Formalizations of this approach [25] have been given in e.g. [11, 3]. The PUHR rule avoids the "blind instantiation" of the γ rule in those - frequent - cases, where the D predicate of theorem 1 is not needed. In such cases, the gain in efficiency compared with tableaux methods relying on the γ rule can be considerable [25]. In the implementation described in [23] the blind instantiation of the γ rule is controlled by giving a limit on the number of γ expansions for each γ formula. In practice, conveniently setting such upper bounds might be difficult. A further interest of the approach presented here is its short and easily adaptable implementation given in [9].
5 Other extensions and refinements of tableau methods generate finite representation for (possibly infinite) models [12, 33, 15, 27]. In [27] a method for extracting models of (possibly infinite) branches by means of equational constraints is described. The approaches [33, 15] 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 [33] only applies to the monadic and Ackermann class. The method of [15] which, like the PUHR [25, 11] and EP tableaux, is based on positive hyperresolution, avoids splitting. In some cases, this results in gains in efficiency. This also makes the method capable of building (finite representations of) *infinite* models for formulas that are not finitely satisfiable. The goal of EP tableaux being completeness for both, refutation and finite satisfiability, the capability of the

[7] Rather unappropriately called "finite unprovability" in [35].

method of [15] to build finite representation of infinite models cannot be taken as comparison criterium: Adding it to the EP tableaux method would make it incorrect with respect to finite satisfiability, indeed. Recall that for the applications outlined in Section 2, that motivated the present paper, finite models are needed and *finitely representable* models are not convenient.

It is difficult to compare the EP tableau method with refutation methods for *clausal* axiom systems, for it does not make much sense to check Skolemized axiom systems for finite satisfiability. Except in trivial cases, Skolemization expands finite Herbrand as well as term models into infinite ones, indeed.

10 Conclusion and Perspectives

Some applications of theorem proving have been discussed that can benefit from a model generator complete for both, refutation and finite satisfiability and furthermore not imposing an upper bound on the size of the models searched for. The approach *Extended Positive (EP) Tableaux*, developed for such applications, has presented. Like the PUHR Tableaux [25, 11] they extend, EP Tableaux rely on positive unit hyperresolution and "range restriction" for avoiding the "blind instantiation" performed by the γ rule of standard tableaux [16, 34, 18]. Instead of relying on Skolemization, as most refutation methods do, Extended Positive Tableaux use the extended δ (or δ^*) rule of [10, 20, 23]. 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 given in [9] in the style of SATCHMO [25] implements the EP Tableaux method with a depth-first strategy. The system described in [8, 9] is based on a breadth-first expansion of EP tableaux. In these papers, neither are EP tableaux formally introduced, nor are soundness and completeness properties established.

The following issues deserve further investigations.

1 An analysis of EP tableaux and FINFIMO's efficiency is needed. This issue is however a difficult one because there are not much systems that are complete for both, finite satisfiability and unsatisfiability, and no benchmarks are available that are fully relevant for finite satisfiability verification. Note that the extended δ (or $\delta*$) rule sometimes cuts down infinite search spaces and thus sometimes results in a gain in efficiency for refutation reasoning. In other cases, however, it expands a larger search space than the standard δ rule [32, 16]. In some cases, the EP tableaux method expands isomorphic interpretations. It would be interesting to investigate, whether techniques for avoiding this, such as the least number heuristic mentioned in [37] can be integrated in the EP tableaux method.

2 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 applications such as described in [8].

3 The method could be extended to languages with function symbols. Due to the existential quantifiers it is possible with EP formulas to express functions by relations. No theoretical extensions would be necessary for such an extension and

constraint reasoning techniques as in [1] seem applicable. Nevertheless, explicit function symbols might be more convenient for some applications. On the other hand, there are applications like e.g., that described in [8] that do not need function symbols at all.
4 For applications such as diagnosis [4], natural language understanding [14], and the database issues mentioned in Section 2 [7, 2, 8, 9], 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 [11] or in [26].

Acknowledgement

The authors are thanksfull to Norbert Eisinger and the referees for usefull remarks and for pointing out some relevant references.

References

1. S. Abdennadher and H. Schütz. Model Generation with Existentially Quantified Variables and Constraints. *Proc. 6th Int. Conf. on Algebraic and Logic Programming*, Springer LNCS 1298, 1997.
2. C. Aravindan and P. Baumgartner. A Rational and Efficient Algorithm for View Deletion in Databases. *Proc. Int. Logic Programming Symposium*, MIT Press, 1997.
3. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper tableaux. *Proc. 5th Europ. Workshop on Logics in AI (JELIA)*, Springer LNCS 1126, 1996.
4. P. Baumgartner, P. Fröhlich, U. Furbach, and W. Nejdl. Tableaux for Diagnosis Applications. *Proc. 6th Workshop on Theorem Proving with Tableaux and Related Methods*, Springer LNAI , 76-90, 1997.
5. P. Baumgartner, P. Fröhlich, U. Furbach, and W. Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. *Proc. 15th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 460-465, 1997.
6. E. W. Beth. The Foundations of Mathematics. North Holland, 1959.
7. F. Bry. Intensional Updates: Abduction via Deduction. *Proc. 7th Int. Conf. on Logic Programming*, MIT Press, 561-575, 1990.
8. F. Bry, N. Eisinger, H. Schütz, and S. Torge. SIC: An Interactive Tool for the Design of Integrity Constraints (System Description). *EDBT'98 Demo Session Proceedings*, 45–46, 1998.
9. F. Bry, N. Eisinger, H. Schütz, and S. Torge. SIC: Satisfiability Checking for Integrity Constraints. *Proc. Deductive Databases and Logic Programming (DDLP), Workshop at JICSLP*, 1998.
10. F. Bry and R. Manthey. Proving Finite Satisfiability of Deductive Databases. *Proc. 1st Workshop on Computer Science Logic*, Springer LNCS 329, 44-55, 1987.
11. F. Bry and A. Yahya. Minimal Model Generation with Positive Unit Hyperresolution Tableaux. *Proc. 5th Workshop on Theorem Proving with Tableaux and Related Methods*, Springer LNAI 1071, 1996.
12. R. Caferra and N. Zabel. Building Models by Using Tableaux Extended by Equational Problems. *J. of Logic and Computation*, 3, 3-25, 1993.
13. M. Denecker and D. de Schreye. On the Duality of Abduction and Model Generation in a Framework of Model Generation with Equality. *Theoretical Computer Science*, 122, 1994.

14. N. Eisinger and T. Geisler. Problem Solving with Model-Generation Approaches based on PUHR Tableaux. *Proc. Workshop on Problem-solving Methodologies with Automated Deduction, Workshop at CADE-15*, 1998.
15. C. Fermüller and A. Leitsch. Hyperresolution and Automated Model Building. *J. of Logic and Computation*, 6:2,173-203, 1996.
16. M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990.
17. H. Fujita and R. Hasegawa. A Model Generation Theorem Prover in KL1 Using a Ramified -Stack Algorithm. *Proc. 8th Int. Conf. on Logic Programming*, MIT Press, 1991.
18. U. Furbach, ed. *Tableaux and Connection Calculi. Part I.* In *Automated Deduction – A Basis for Applications.* Kluwer Academic Publishers, 1998. To appear.
19. H. Gallaire, J. Minker, and J.-M. Nicolas. Logic and Databases: A Deductive Approach. *ACM Computing Surveys*, 16:2, 1984.
20. J. Hintikka. Model Minimization - An Alternative to Circumscription. *J. of Automated Reasoning*, 4, 1988.
21. K. M. Hörnig. Generating Small Models of First Order Axioms. *Proc. GWAI-81*, Informatik-Fachberichte 47, Springer-Verlag, 1981.
22. M. Kettner and N. Eisinger. The Tableau Browser SNARKS (System Description). *Proc. 14th Int. Conf. on Automated Deduction*, Springer LNAI 1249, 1997.
23. S. Lorenz. A Tableau Prover for Domain Minimization. *J. of Automated Reasoning*, 13, 1994.
24. D. W. Loveland, D. W. Reed, and D. S. Wilson. SATCHMORE: SATCHMO with RElevancy. *J. of Automated Reasoning*, 14, 325–351, 1995.
25. R. Manthey and F. Bry. SATCHMO: A Theorem Prover Implemented in Prolog. *Proc. 9th Int. Conf. on Automated Deduction*, Springer LNAI 310, 1988.
26. I. Niemelä. A Tableau Calculus for Minimal Model Reasoning. *Proc. 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, Springer LNAI 1071, 1996.
27. N. Peltier. Simplifying and Generalizing Formulae in Tableaux. Pruning the Search Space and Building Models. *Proc. 6th Workshop on Theorem Proving with Tableaux and Related Methods*, Springer LNAI 1227, 1997.
28. D. Poole. Normality and Faults in Logic-Based Diagnosis. *Proc. 11th Int. Joint Conf. on Artificial Intelligence*, 1304–1310, 1985.
29. R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32, 57–95, 1987.
30. M. Paramasivam and D. Plaisted. Automated Deduction Techniques for Classification in Description Logic Systems. *J. of Automated Reasoning*, 20(3), 1998.
31. J. Slaney. Finder (finite domain enumerator): Notes and Guides. Tech. rep., Australian National University Automated Reasoning Project, Canberra, 1992.
32. R. Smullyan. *First-Order Logic.* Springer, 1968.
33. T. Tammet. Using Resolution for Deciding Solvable Classes and Building Finite Models. *Baltic Computer Science*, Springer LNCS 502, 1991.
34. G. Wrightson. ed. Special Issue on Automated Reasoning with Analytic Tableaux, Parts I and II. *J. of Automated Reasoning*, 13:2 and 3, 173–421, 1994.
35. M. Tiomkin. Proving Unprovability. *Proc. 3rd Symp. Logic in Computer Science*, 22–26, 1988.
36. B. A. Trakhtenbrot. Impossibility of an Algorithm for the Decision Problem in Finite Classes. *Proc. Dokl. Acad. Nauk.*, SSSR 70, 1950.
37. Jian Zhang and Hantao Zhang. SEM: A System for Enumerating Models. *Proc. International Joint Conference on Artificial Intelligence*, 1995.