A New Fast Tableau-Based Decision Procedure for an Unquantified Fragment of Set Theory*

Domenico Cantone[1] and Calogero G. Zarba[2]

[1] Università di Catania, Dipartimento di Matematica,
Viale A. Doria 6, I-95125 Catania, Italy,
e-mail: cantone@cs.unict.it
[2] Stanford University, Computer Science Department,
Gates Building, Stanford CA 94305, USA,
e-mail: zarba@theory.stanford.edu

Abstract. In this paper we present a new fast tableau-based decision procedure for the ground set-theoretic fragment Multi-Level Syllogistic with Singleton (in short **MLSS**) which avoids the interleaving of model checking steps.
The underlying tableau calculus is based upon the system **KE**.

1 Introduction

In the last few years many fragments of set theory have been proved decidable [6, 5]. However, the problem of finding *efficient* decision procedures for these fragments still remains largely unexplored.

In this paper we present a new fast tableau-based decision procedure for the ground set-theoretic fragment Multi-Level Syllogistic with Singleton (in short **MLSS**).

Tableaux have the appealing feature that it is easy to extract a counter-example from an open and saturated branch, but on the other hand they can be highly inefficient if the splitting rules are not designed properly, at least for certain classes of formulae. We address this anomaly by presenting a tableau calculus based on the system **KE** introduced in [7] which forces tableau branches to be mutually exclusive. This results in an exponential speed-up with respect to Smullyan tableau-based calculi.[1]

In addition, in the procedure we are going to describe useful cuts are recognized in constant time, without the interleaving model-checking steps approach used in [1, 4, 9]. Moreover, useless cuts that might be executed by an exhaustive search strategy are totally avoided. This will have the overall effect to considerably speed up the saturation process with respect to the previous approaches.

* Work partially supported by the C.N.R. of Italy, coordinated project SETA, by M.U.R.S.T. Project "Tecniche speciali per la specifica, l'analisi, la verifica, la sintesi e la trasformazione di programmi", and by project "Deduction in Set Theory: A Tool for Software Verification" under the 1998 Vigoni Program.

[1] See [7] for further details about the cited speed-up.

Our decision procedure has been implemented as part of the Stanford Temporal Prover, STeP [2], a system which supports the computer-aided verification of reactive, real time and hybrid systems based on temporal specification. The integration of our decision procedure with STeP First-Order Reasoning and STeP's other decision procedures is done using the method described in [3].

The paper is organized as follows. In Section 2 we introduce some preliminary concepts which will be useful in what follows. In Section 3 we present a decidable tableau calculus for **MLSS**, whose proof of correctness is given in Section 4. Finally, after giving in Section 5 some experimental results, in Section 6 we hint at some directions of future research.

2 Preliminaries

In this section we introduce the syntax and semantics of **MLSS**, as well as the concept of realization.

2.1 Syntax

The unquantified set-theoretic fragment **MLSS** contains

- a denumerable infinity of variables,
- the constant $\emptyset$ (empty set),
- the operator symbols $\sqcup$ (union), $\sqcap$ (intersection), $-$ (set difference) and $[\bullet]$ (singleton),
- the predicate symbols $\sqsubset$ (membership) and $\approx$ (equality), and
- the logical connectives $\neg$, $\wedge$ and $\vee$.[2]

Plainly, the predicate $\sqsubseteq$ and the finite enumeration operator $[\bullet, \bullet, \ldots, \bullet]$ can be expressed in **MLSS** by noticing that $s \sqsubseteq t$ is equivalent to $s \sqcup t \approx t$ and that $[t_1, t_2, ..., t_k]$ can be expressed by $[t_1] \sqcup \ldots \sqcup [t_k]$.

We denote by T_φ the collection of all terms occurring in the formula φ, and we use the abbreviations $s \not\sqsubset t$ and $s \not\approx t$ to denote $\neg(s \sqsubset t)$ and $\neg(s \approx t)$, respectively .

2.2 Semantics

The semantics of **MLSS** is based upon the von Neumann standard cumulative hierarchy $\mathcal{V}$ of sets defined by:

$$\begin{aligned} \mathcal{V}_0 &= \emptyset \\ \mathcal{V}_{\alpha+1} &= \mathcal{P}(\mathcal{V}_\alpha), \qquad \text{for each ordinal } \alpha \\ \mathcal{V}_\lambda &= \bigcup_{\mu<\lambda} \mathcal{V}_\mu, \quad \text{for each limit ordinal } \lambda \\ \mathcal{V} &= \bigcup_{\alpha \in On} \mathcal{V}_\alpha, \end{aligned}$$

[2] In our treatment, $\neg\neg p$ is considered to be a syntactic variation of p.

where $\mathcal{P}(S)$ is the power set of S and On denotes the class of all ordinals. It can easily be seen that there can be no membership cycle in $\mathcal{V}$, namely sets in $\mathcal{V}$ are well-founded with respect to membership.

An ASSIGNMENT M over a collection V of variables is any function $M : V \to \mathcal{V}$. Given an assignment M over the variables of a formula φ, we denote with $M\varphi$ the truth-value obtained by interpreting each variable v in φ with the set Mv and the set symbols and logical connectives according to their standard meaning (thus, for instance, $\sqcup$, $\sqcap$, $-$, $[\bullet]$, $\sqsubseteq$, and $\approx$ are interpreted as the set operators $\cup$, $\cap$, $\setminus$, $\{\bullet\}$, and as the set predicates $\in$ and $=$, respectively).

A SET MODEL for a formula φ is an assignment M over the collection of variables occurring in φ such that $M\varphi$ evaluates to true.

A formula φ is SATISFIABLE if it has a set model.

2.3 Realizations

Let $G = (N, \widehat{\in})$ be a directed acyclic graph, and let (P, T) be a bipartition of N. Also, let $\{u_x : x \in P\}$ be a family of sets.

Definition 1. *The* REALIZATION *of $G = (N, \widehat{\in})$ relative to $\{u_x : x \in P\}$ and to (P, T) is the assignment R over N recursively defined by:*

$$\begin{aligned} Rx &= \{u_x\}, && \text{for } x \text{ in } P \\ Rt &= \{Rs : s \mathrel{\widehat{\in}} t\}, && \text{for } t \text{ in } T. \end{aligned}$$

Observe that R is well-defined since G is acyclic.

Next we define the function $h : N \to \mathbb{N}$ (called the HEIGHT), by putting:

$$h(t) = \begin{cases} 0 & \text{if } t \in P \text{ or } s \mathrel{\widehat{\notin}} t, \text{ for all } s \in N \\ \max\{h(s) : s \mathrel{\widehat{\in}} t\} + 1 & \text{otherwise.} \end{cases}$$

The following lemma states the main properties of realizations.

Lemma 1. *Let $G = (P \cup T, \widehat{\in})$ be a directed acyclic graph, with $P \cap T = \emptyset$. Also, let $\{u_x : x \in P\}$ and R be respectively a family of sets and the realization of G relative to $\{u_x : x \in P\}$ and (P, T). Assume that:*

(a) $u_x \neq u_y$ for all distinct x, y in P;
(b) $u_x \neq Rt$, for all x in P and t in $P \cup T$.[3]

Then the following properties hold:

(i) if $s \mathrel{\widehat{\in}} t$ then $h(s) < h(t)$, for all s in $P \cup T$ and t in T;
(ii) if $Rt_1 = Rt_2$ then $h(t_1) = h(t_2)$, for all t_1, t_2 in $P \cup T$;
(iii) if $Rs \in Rt$ then $h(s) < h(t)$, for all s, t in $P \cup T$.

[3] Notice that conditions (a) and (b) can always be satisfied by letting the u_x's be pairwise distinct sets of cardinality no less than $|P \cup T|$, since $|Rt| < |P \cup T|$.

Proof. (i) is immediate.

(ii) If either t_1 or t_2 is in P, the claim is ensured by conditions (a) and (b). On the other hand, if $t_1, t_2 \in T$, we proceed by induction on $\max\{h(t_1), h(t_2)\}$. The base case ($\max\{h(t_1), h(t_2)\} = 0$) is trivial. For the inductive step, suppose $Rt_1 = Rt_2$ and, without lost of generality, that $h(t_1) > 0$. Then there exists s such that $s \mathrel{\widehat{\in}}_{t1}$ and $h(t_1) = h(s) + 1$. Since $s \mathrel{\widehat{\in}}_{t1}$, it follows that $Rs \in Rt_1$ and therefore $Rs \in Rt_2$. Moreover, there exists s' such that $Rs' = Rs$ and $s' \mathrel{\widehat{\in}}_{t2}$. By inductive hypothesis $h(s) = h(s')$. Finally, $h(t_2) \geq h(s') + 1 = h(s) + 1 = h(t_1)$. Exchanging the rôles of t_1 and t_2 it is also possible to deduce $h(t_2) \leq h(t_1)$. Thus $h(t_1) = h(t_2)$.

(iii) If $Rs \in Rt$ then there exists some s' such that $Rs' = Rs$ and $s' \mathrel{\widehat{\in}}_t$. For (i) $h(s') < h(t)$ and for (ii) $h(s) = h(s')$. Thus $h(s) < h(t)$. □

3 The Tableau Calculus

In this section we describe a tableau calculus for **MLSS**. See [8] for a complete introduction to semantic tableaux.

We extend the notion of closed tableau as follows:

Definition 2. *A branch θ of a tableau $\mathcal{T}$ is closed if it contains:*

- *two complementary formulae ψ, $\neg\psi$, or*
- *a membership cycle of the form $t_0 \sqsubseteq t_1 \sqsubseteq \ldots \sqsubseteq t_0$, or*
- *a literal of the form $t \not\approx t$, or*
- *a literal of the form $s \sqsubseteq \emptyset$.*

A tableau is CLOSED *if all its branches are closed.*

3.1 Saturation Rules

Our calculus has two kinds of rules: *saturation* and *fulfilling* rules. Moreover, we impose the restriction that *no new term will be created by any application of a saturation rule.* Thus, for instance, the rule

$$s \sqsubseteq t_1 \Longrightarrow s \sqsubseteq t_1 \sqcup t_2$$

can be applied to a branch θ of a tableau for φ *only* if the term $t_1 \sqcup t_2$ is already in T_φ. Under this fundamental restriction, the full collection of saturation rules is shown in Table 1. Notice also that in the first two rules for equality, ℓ stands for a literal, and the substituted term is restricted to be a top-level term occurring in ℓ. This will prevent the search space from exploding.

A branch is said to be LINEARLY SATURATED if no saturation rule produces new formulae.

propositional rules		rules for $\sqcup$	
$p \wedge q$	$\Longrightarrow p, q$	$s \not\sqsubseteq t_1 \sqcup t_2$	$\Longrightarrow s \not\sqsubseteq t_1, s \not\sqsubseteq t_2$
$\neg(p \vee q)$	$\Longrightarrow \neg p, \neg q$	$s \sqsubseteq t_1$	$\Longrightarrow s \sqsubseteq t_1 \sqcup t_2$
$p \vee q, \neg p$	$\Longrightarrow q$	$s \sqsubseteq t_2$	$\Longrightarrow s \sqsubseteq t_1 \sqcup t_2$
$p \vee q, \neg q$	$\Longrightarrow p$	$s \sqsubseteq t_1 \sqcup t_2, s \not\sqsubseteq t_1$	$\Longrightarrow s \sqsubseteq t_2$
$\neg(p \wedge q), p$	$\Longrightarrow \neg q$	$s \sqsubseteq t_1 \sqcup t_2, s \not\sqsubseteq t_2$	$\Longrightarrow s \sqsubseteq t_1$
$\neg(p \wedge q), q$	$\Longrightarrow \neg p$	$s \not\sqsubseteq t_1, s \not\sqsubseteq t_2$	$\Longrightarrow s \not\sqsubseteq t_1 \sqcup t_2$
rules for $\sqcap$		rules for $-$	
$s \sqsubseteq t_1 \sqcap t_2$	$\Longrightarrow s \sqsubseteq t_1, s \sqsubseteq t_2$	$s \sqsubseteq t_1 - t_2$	$\Longrightarrow s \sqsubseteq t_1, s \not\sqsubseteq t_2$
$s \not\sqsubseteq t_1$	$\Longrightarrow s \not\sqsubseteq t_1 \sqcap t_2$	$s \not\sqsubseteq t_1$	$\Longrightarrow s \not\sqsubseteq t_1 - t_2$
$s \not\sqsubseteq t_2$	$\Longrightarrow s \not\sqsubseteq t_1 \sqcap t_2$	$s \sqsubseteq t_2$	$\Longrightarrow s \not\sqsubseteq t_1 - t_2$
$s \not\sqsubseteq t_1 \sqcap t_2, s \sqsubseteq t_1$	$\Longrightarrow s \not\sqsubseteq t_2$	$s \not\sqsubseteq t_1 - t_2, s \sqsubseteq t_1$	$\Longrightarrow s \sqsubseteq t_2$
$s \not\sqsubseteq t_1 \sqcap t_2, s \sqsubseteq t_2$	$\Longrightarrow s \not\sqsubseteq t_1$	$s \not\sqsubseteq t_1 - t_2, s \not\sqsubseteq t_2$	$\Longrightarrow s \not\sqsubseteq t_1$
$s \sqsubseteq t_1, s \sqsubseteq t_2$	$\Longrightarrow s \sqsubseteq t_1 \sqcap t_2$	$s \sqsubseteq t_1, s \not\sqsubseteq t_2$	$\Longrightarrow s \sqsubseteq t_1 - t_2$
rules for $[\bullet]$		rules for equality	
	$\Longrightarrow t_1 \sqsubseteq [t_1]$	$t_1 \approx t_2, \ell$	$\Longrightarrow \ell\{t_2/t_1\}$
$s \sqsubseteq [t_1]$	$\Longrightarrow s \approx t_1$	$t_1 \approx t_2, \ell$	$\Longrightarrow \ell\{t_1/t_2\}$
$s \not\sqsubseteq [t_1]$	$\Longrightarrow s \not\approx t_1$	$s \sqsubseteq t, s' \not\sqsubseteq t$	$\Longrightarrow s \not\approx s'$

Table 1. Saturation rules.

3.2 Fulfilling Rules

A fulfilling rule can be applied to an open linearly saturated branch, provided that its associated *precondition* and *subsumption requirement* are, respectively, true and false. Table 2 summarizes the fulfilling rules and their associated preconditions and subsumption requirements. Notice that even fulfilling rules (that, incidentally, in our calculus are exactly the splitting rules) are not allowed to introduce new terms, with the exception of the last one, which introduces fresh parameters x not occurring in the branch to which it is applied.

Remark 1. Notice that literals of type $s \not\sqsubseteq t_1 \sqcap t_2$ and $s \not\sqsubseteq t_1 - t_2$ do not trigger any split rule, as would happen in an exhaustive search strategy.

Notice also the asymmetry in the precondition for $\sqcap$: no split needs to occur if for some term $t_1 \sqcap t_2$ in T_φ a literal $s \sqsubseteq t_2$ occurs in a branch.

In early versions of this work all sorts of cut rules were allowed, whereas a careful analysis of the correctness proof has pointed out that most of them can be avoided.

Remark 2. Observe that if the literals $s_1 \approx s_2$, $t_1 \approx t_2$, $s_1 \not\approx t_1$, $s_1 \not\approx t_2$, $s_2 \not\approx t_1$, $s_2 \not\approx t_2$ occur in a branch, an exhaustive search strategy would apply a splitting rule to each inequality, thereby generating 2^4 branches, whereas in our calculus at most 2 branches will eventually be created.

Remark 3. It is possible to further strengthen the subsumption requirement associated to the last fulfilling rule by noticing that if a literal $t \not\approx \emptyset$ occurs in a

fulfilling rule	precondition	subsumption requirement
$\overline{p \mid \neg p}$	$p \vee q$ is in θ	p is in θ or $\neg p$ is in θ
$\overline{\neg p \mid p}$	$\neg(p \wedge q)$ is in θ	$\neg p$ is in θ or p is in θ
$\overline{s \sqsubseteq t_1 \mid s \not\sqsubseteq t_1}$	$t_1 \sqcup t_2 \in T_\varphi$ $s \sqsubseteq t_1 \sqcup t_2$ is in θ	$s \sqsubseteq t_1$ is in θ or $s \not\sqsubseteq t_1$ is in θ
$\overline{s \sqsubseteq t_2 \mid s \not\sqsubseteq t_2}$	$t_1 \sqcap t_2 \in T_\varphi$ $s \sqsubseteq t_1$ is in θ	$s \sqsubseteq t_2$ is in θ or $s \not\sqsubseteq t_2$ is in θ
$\overline{s \sqsubseteq t_2 \mid s \not\sqsubseteq t_2}$	$t_1 - t_2 \in T_\varphi$ $s \sqsubseteq t_1$ is in θ	$s \sqsubseteq t_2$ is in θ or $s \not\sqsubseteq t_2$ is in θ
$\overline{x \sqsubseteq t_1 \mid x \not\sqsubseteq t_1}$ $\overline{x \not\sqsubseteq t_2 \mid x \sqsubseteq t_2}$	$t_1, t_2 \in T_\varphi$ $t_1 \not\approx t_2$ is in θ	$\exists x : (x \sqsubseteq t_1$ is in θ and $x \not\sqsubseteq t_2$ is in $\theta)$ or $\exists x : (x \not\sqsubseteq t_1$ is in θ and $x \sqsubseteq t_2$ is in $\theta)$

Table 2. Fulfilling rules.

branch θ, then it is enough to require that $x \sqsubseteq t$ occurs in θ for some x, thus obtaining the *linear* fulfilling rule

$$t \not\approx \emptyset \Longrightarrow x \sqsubseteq t \ (x \text{ new parameter})$$

This improvement will be used in Example 1.

More generally, one can maintain a *transitivity graph* [3] whose nodes are labeled with terms in $P_\theta \cup T_\varphi$ and edges are labeled with $\sqsubseteq$, $\not\approx$ or $\not\sqsubseteq$. Then, if a literal $t_1 \not\approx t_2$ occurs in a branch θ, we may check whether there exists a path from t_1 to t_2 (or from t_2 to t_1) with edges labeled with $\sqsubseteq$, and the fulfilling rule would then be:

$$t_1 \not\approx t_2, t_1 \sqsubseteq t_2 \Longrightarrow x \sqsubseteq t_2, x \not\sqsubseteq t_1 \ (x \text{ new parameter}) .$$

We should also notice that if the literals $t_1 \not\approx t_2$, $t_1 \sqsubseteq \ldots \sqsubseteq t_2$ occur in a branch θ, we do not need to apply any fulfilling rule at all. Soundness of such optimizations is an easy matter.

Example 1. Table 1 contains a closed tableau with 3 branches for proving the validity of the formula $\neg(x \approx [y] \wedge x \approx y \sqcup z) \vee (y \approx \emptyset \wedge x \approx z)$.

We denote with φ_i the formula labeling node i, and provide justifications for the construction of the tableau.

- $\varphi_2, \varphi_3, \varphi_4, \varphi_5$ and φ_7 are obtained by means of propositional rules;
- φ_6 and φ_{17} are obtained by means of the second fulfilling rule;
- $\varphi_8, \varphi_9, \varphi_{13}$ and φ_{14} are obtained by means of the last fulfilling rule;
- $\varphi_{10}, \varphi_{12}, \varphi_{16}, \varphi_{20}, \varphi_{21}$ and φ_{23} are obtained by means of equality rules;
- $\varphi_{11}, \varphi_{15}$ and φ_{19} are obtained by means of rules for $\sqcup$;

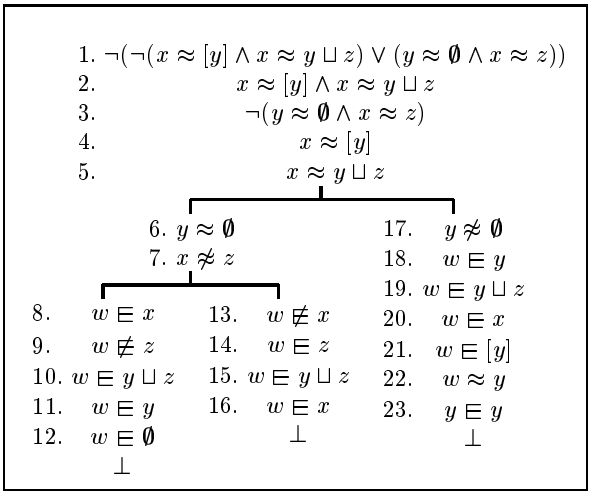

Fig. 1. A closed tableau for $\neg(\neg(x \approx [y] \wedge x \approx y \sqcup z) \vee (y \approx \emptyset \wedge x \approx z))$

– the optimization promised in Remark 3 is used to deduce φ_{18};
– φ_{22} is obtained by means of a rule for $[\bullet]$.

The tableau is closed since the leftmost branch contains the contradiction φ_{12}, the central branch contains two complementary literals $\varphi_{13}, \varphi_{16}$, and the rightmost branch contains a membership cycle (φ_{23}).

Finally, notice that to prove the same formula, the approach described in [4] produced a tableau with 8 branches

4 The Decision Procedure

In this section, after introducing some definition and terminology, we state our decision procedure and prove its correctness.

Definition 3. *To any branch θ of a tableau $\mathcal{T}$ for a formula φ we associate the following objects:*

P_θ: *the collection of parameters added to θ;*
V_θ: *the collection of variables and parameters occurring in θ;*
P'_θ: *the collection of parameters $\{x \in P_\theta$: there is no t in T_φ such that $x \approx t$ occurs in $\theta\}$;*
T'_θ: *the set $T_\varphi \cup (P_\theta \setminus P'_\theta)$;*
G_θ: *the oriented graph $(P'_\theta \cup T'_\theta, \widehat{\sqsubseteq})$, where $s \,\widehat{\sqsubseteq}_t$ if and only if the literal $s \sqsubseteq t$ occurs in θ;*
R_θ: *a realization of G_θ relative to the partition (P'_θ, T'_θ) and to pairwise distinct sets u_x, for $x \in P'_\theta$, each having cardinality no less than $|P'_\theta \cup T'_\theta|$;*

M_θ: the assignment over V_θ defined by $M_\theta v = R_\theta v$, for each v in V_θ.

Definition 4. *An open branch θ is* SATURATED *if it is linearly saturated and all its subsumption requirements are fulfilled.*

Definition 5. *A branch θ is said to be* COHERENT *if $R_\theta t = M_\theta t$, for all t in $P_\theta \cup T_\varphi$.*

Procedure 1 (MLSS-satisfiability test).

Input: an **MLSS**-formula φ.

1. Let $\mathcal{T}$ be the tableau consisting of a single node labeled with φ;
2. linearly saturate $\mathcal{T}$ by strictly applying to it all possible saturation rules until either $\mathcal{T}$ is closed or no new formula can be produced;
3. if $\mathcal{T}$ is closed, announce that φ is unsatisfiable;
4. otherwise, if there exists an open and saturated branch θ in $\mathcal{T}$, announce that φ is satisfied by the model M_θ;
5. otherwise, let θ be a non-saturated open branch; apply to θ any fulfilling rule whose subsumption requirement is false and go to step 2.

4.1 Termination

We begin to prove total correctness of Procedure 1 by first showing termination, and leave the proof of partial correctness for the next subsection.

Theorem 1. *Procedure 1 always terminate.*

Proof. Let φ be the root formula of the tableau limit $\mathcal{T}$ constructed by Procedure 1. Since steps 3 and 4 cause the procedure to terminate, and step 5 always add new formulae, to show termination it is enough to prove that $\mathcal{T}$ must be finite. Now, let θ be any branch in $\mathcal{T}$. Because of the restriction imposed to the application of the rules, $|T_\theta|$ is bounded by $|T_\varphi|^2$, and therefore $T_\varphi \cup T_\theta$ is finite. It follows that the number of literals occurring in θ is finite, as well as the number of formulae involving propositional connectives. Having shown that any branch in $\mathcal{T}$ is finite, in view of the König Lemma even $\mathcal{T}$ is finite. □

4.2 Partial Correctness

Let again φ be the root formula of the tableau $\mathcal{T}$ constructed by Procedure 1. Since all rules are plainly sound, if $\mathcal{T}$ is closed then φ is unsatisfiable. Otherwise the tableau $\mathcal{T}$ must contain an open and saturated branch θ. Thus, in order to establish the correctness of Procedure 1, it is enough to prove that the assignment M_θ (cf. Definition 3) satisfies the branch θ and, therefore, the formula φ.

The following lemma is easily proved by induction on the number of applications of the inferences rules.

Lemma 2. *In any branch θ if $x \in P'_\theta$ then:*

(a) there can be no term t in $T_\varphi \cup P_\theta$ different from x such that $x \approx t$ occurs in θ;
(b) there can be no term s in $T_\varphi \cup P_\theta$ such that $s \sqsubset x$ occurs in θ.

In order to show that the assignment M_θ models correctly all formulae occurring in an open and saturated branch θ, we first show in the following lemma that the realization R_θ models correctly all literals in an open and saturated branch θ, provided that terms are just considered as "complex names" for variables (namely operators are not interpreted).

Lemma 3. *Let θ be an open and saturated branch. Then:*

(i) if $s \sqsubset t$ occurs in θ, then $R_\theta s \in R_\theta t$;
(ii) if $t_1 \approx t_2$ occurs in θ, then $R_\theta t_1 = R_\theta t_2$;
(iii) if $t_1 \not\approx t_2$ occurs in θ, then $R_\theta t_1 \neq R_\theta t_2$;
(iv) if $s \not\sqsubset t$ occurs in θ, then $R_\theta s \notin R_\theta t$.

Proof. (i) Let $s \sqsubset t$ be in θ. By Lemma 2, $t \notin P'_\theta$, and by construction of R_θ it trivially follows that $R_\theta s \in R_\theta t$.

(ii) Let $t_1 \approx t_2$ be in θ. If either $t_1 \in P'_\theta$ or $t_2 \in P'_\theta$ then by Lemma 2 it must be $t_1 = t_2$ and therefore $R_\theta t_1 = R_\theta t_2$. If $t_1, t_2 \in T'_\theta$ but $R_\theta t_1 \neq R_\theta t_2$, suppose w.l.o.g. that there is some a such that $a \in R_\theta t_1$ and $a \notin R_\theta t_2$. Then there exists s such that $R_\theta s = a$ and $s \sqsubset t_1$ occurs in θ. Since θ is saturated, $s \sqsubset t_2$ must also occur in θ, and by (i) $a = R_\theta s \in R_\theta t_2$, a contradiction.

(iii) Let $t_1 \not\approx t_2$ be in θ but $R_\theta t_1 = R_\theta t_2$. W.l.o.g. we can assume that $t_1, t_2 \in T_\varphi$ (otherwise either at least one among t_1, t_2 is in P'_θ, and the claim easily follows from Lemma 2, or θ would contain a literal $t'_1 \not\approx t'_2$ with $t'_1, t'_2 \in T_\varphi$ and such that $t_1 \approx t'_1$ and $t_2 \approx t'_2$ are in θ; then $t'_1 \not\approx t'_2$ could play the rôle of $t_1 \not\approx t_2$ in the following discussion). By Lemma 1 we have $h(t_1) = h(t_2)$. We proceed by induction on $h(t_1)$. In the base case ($h(t_1) = 0$) we reach a contradiction, since by saturation there is some x such that either $x \sqsubset t_1$ and $x \not\sqsubset t_2$ occur in θ, or $x \not\sqsubset t_1$ and $x \sqsubset t_2$ occur in θ, and we would have $h(t_1) > 0$ in either cases. For the inductive step, w.l.o.g. let $x \sqsubset t_1$ and $x \not\sqsubset t_2$ be in θ (their occurrence is due to saturation), for some x. Then $R_\theta x \in R_\theta t_1$ that implies $R_\theta x \in R_\theta t_2$, so that there exists x' such that $R_\theta x = R_\theta x'$ and $x' \sqsubset t_2$ occurs in θ. Notice that $x' \neq x$ (otherwise θ would be closed). Since by Lemma 1 we have $h(x) = h(x') < h(t_1)$, we can apply the inductive hypothesis and obtain the contradiction $R_\theta x \neq R_\theta x'$.

(iv) Let $s \not\sqsubset t$ be in θ but $R_\theta s \in R_\theta t$. Then there exists s' different from s such that $R_\theta s = R_\theta s'$ and $s' \sqsubset t$ occurs in θ. By saturation $s \not\approx s'$ is in θ, and by (iii) $R_\theta s \neq R_\theta s'$, a contradiction. □

Next we show that even operators are correctly modeled by R_θ (and therefore by M_θ), for an open and saturated branch θ.

Lemma 4. *If a branch θ is open and saturated, then it is coherent.*

Proof. Let θ be an open and saturated branch. We prove that $R_\theta t = M_\theta t$, for each t in $P_\theta \cup T_\varphi$, by structural induction on t. The base case is trivial for variables. Concerning $\emptyset$, notice that trivially $M_\theta \emptyset = \emptyset$ and that $R_\theta \emptyset = \emptyset$ since θ is open. For the inductive step we prove only that $R_\theta(t_1 \sqcap t_2) = M_\theta(t_1 \sqcap t_2)$ (other cases are similar). Suppose that $a \in R_\theta(t_1 \sqcap t_2)$. Then there exists s such that $R_\theta s = a$ and $s \sqsubseteq t_1 \sqcap t_2$ occurs in θ, and since θ is saturated both $s \sqsubseteq t_1$ and $s \sqsubseteq t_2$ occur in θ. By Lemma 3 $R_\theta s \in R_\theta t_1$ and $R_\theta s \in R_\theta t_2$, and by inductive hypothesis $a \in M_\theta t_1 \cap M_\theta t_2 = M_\theta(t_1 \sqcap t_2)$. Conversely, if $a \in M_\theta(t_1 \sqcap t_2)$ then $a \in M_\theta t_1 \cap M_\theta t_2$, and by inductive hypothesis $a \in R_\theta t_1 \cap R_\theta t_2$. After noticing that, because of the restrictions imposed to the application of the rules, it must be the case that $t_1, t_2 \in T_\varphi$, it follows that there exist s', s'' such that $Rs' = Rs'' = a$ and both $s' \sqsubseteq t_1$ and $s'' \sqsubseteq t_2$ occur in θ. By saturation, either $s' \sqsubseteq t_2$ or $s' \not\sqsubseteq t_2$ occurs in θ. In the former case $s' \sqsubseteq t_1 \sqcap t_2$ occurs in θ, and therefore $a \in R_\theta(t_1 \sqcap t_2)$. In the latter case $s' \not\approx s''$ occurs in θ, and therefore $R_\theta s' \neq R_\theta s''$, a contradiction. □

The following theorem concludes the proof of partial correctness.

Theorem 2. *If θ is an open and saturated branch, then it is satisfiable, and indeed it is satisfied by M_θ.*

Proof. Let θ be an open and saturated branch. By combining together Lemma 3 and 4, it follows that M_θ satisfies all literals occurring in θ. Finally, proceeding by structural induction, it easy to see that even formulae involving propositional connectives are satisfied by M_θ. □

5 Some Experimental Results

On a 200 Mhz ULTRA-Spark Sun workstation, the formulae $\neg(x \approx [y] \wedge x \approx y \sqcup z) \vee (y \approx \emptyset \wedge x \approx z)$ (cf. Example 1) and $a \sqcup (b \sqcup c) \approx (a \sqcup b) \sqcup c$ were proved valid in 0.03 seconds and 0.02 seconds, respectively, whereas the formula $\neg(x \sqsubseteq y \wedge x \not\sqsubseteq z_1 \wedge z_1 \sqcup z_2 \sqsubseteq [y])$ was recognized not to be valid in 0.04 seconds.

Moreover, using the basic notion of pair $(a, b) =_{\mathrm{Def}} \{\{a\}, \{a, b\}\}$ due to Kuratowski, it has been possible to prove the validity of $(a, b) \neq (u, b) \vee (a = u \wedge b = v)$ in 0.08 seconds. Similar theorems have also been proved for 3-tuples, 4-tuples and 5-tuples in 0.24, 0.58 and 1.79, respectively.

6 Future plans

We plan to further investigate heuristics which allow to strengthen subsumption requirements, as hinted in Remark 3.

Also, we intend to study thoroughly the cases in which cuts are really needed, in order to further optimize our calculus.

Finally, we plan to generalize our tableau calculus and relative saturation strategy to extensions of **MLSS** (cf. [6, 5]).

Acknowledgments

The authors wish to thank Bernhard Beckert, Nikolaj S. Bjørner, and Tomás E. Uribe for helpful comments. The second author wishes to thank Prof. Zohar Manna for having given him the opportunity to visit his REACT group.

References

1. Bernhard Beckert and Ulrike Hartmer. A tableau calculus for quantifier-free set theoretic formulae. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 93–107. Springer, 1998.
2. Nikolaj S. Bjørner, Anca Browne, Eddie S. Chang, Michael Colón, Arjun Kapur, Zohar Manna, Henny B. Sipma, and Tomás E. Uribe. STeP: Deductive-algorithmic verification of reactive and real-time systems. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 415–418. Springer, 1996.
3. Nikolaj S. Bjørner, Tomas E. Uribe, and Mark E. Stickel. A practical integration of first-order reasoning and decision procedures. In *Automated Deduction – CADE-14*, volume 1249 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 1997.
4. Domenico Cantone. A fast saturation strategy for set-theoretic tableaux. In Didier Galmiche, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1227 of *Lecture Notes in Artificial Intelligence*, pages 122–137. Springer, 1997.
5. Domenico Cantone and Alfredo Ferro. Techniques of computable set theory with applications to proof verification. *Communications on Pure and Applied Mathematics*, 48:1–45, 1995.
6. Domenico Cantone, Alfredo Ferro, and Eugenio G. Omodeo. *Computable Set Theory*, volume 6 of *International Series of Monographs on Computer Science*. Clarendon Press, 1989.
7. Marcello D'Agostino and Marco Mondadori. The taming of the cut. Classical refutations with analytic cut. *Journal of Logic and Computation*, 4(3):285–319, 1994.
8. Melvin C. Fitting. *First-Order Logic and Automated Theorem Proving*. Graduate Texts in Computer Science. Springer, 2nd edition, 1996. 1st ed., 1990.
9. Calogero G. Zarba. Dimostrazione automatica di formule insiemistiche con tagli analitici. Bachelor dissertation (in Italian), University of Catania, 1998.