A Tableaux Calculus for Reducing Proof Size

Michael Peter Lettmann, Nicolas Peltier

A Tableaux Calculus for Reducing Proof Size

Michael Peter Lettmann[1⋆] and Nicolas Peltier[2]

[1] Technische Universität Wien, Institute of Logic and Computation, Vienna, Austria
michael.lettmann@tuwien.ac.at
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble France
Nicolas.Peltier@univ-grenoble-alpes.fr

Abstract. A tableau calculus is proposed, based on a compressed representation of clauses, where literals sharing a similar shape may be merged. The inferences applied on these literals are fused when possible, which can reduce the size of the proof. It is shown that the obtained proof procedure is sound, refutationally complete and can reduce the size of the tableau by an exponential factor. The approach is compatible with all usual refinements of tableaux.

1 Introduction

Tableau methods (see for instance [2] or [6]) always played a crucial role in the development of new techniques for automated theorem proving. They are easy to comprehend and implement, well-adapted to interactive theorem proving, and, therefore, normally form the basis of the first proof procedure for any newly defined logic [4]. Nonetheless, they cannot compete with resolution-based calculi both in terms of efficiency and deductive power (i.e. proof length, see for instance [3]). This is partly due to the ability of resolution-based methods to generate lemmas and to simulate atomic cuts[3] in a feasible way. There have been attempts to integrate some restricted forms of cut into tableau methods, improving both efficiency and proof size (see for instance [11,6]). But, for more general forms of cuts, it is difficult to decide whether an application of the cut rule is useful or not, thus the rule is not really applicable during proof search. Instead, cuts may be introduced after the proof is generated, to make it more compact by introducing lemmas and fusing recurring patterns [8,9].

In this paper, rather than trying to integrate cuts into the tableau calculus, we devise a new tableau procedure in which a proof compression, that is similar to the compressive power of a Π_2-cut, is achieved by employing a shared representation of literals. Formal definitions will be given later, but we now provide a simple example to illustrate our ideas. Consider the schema of clause sets: $\{\bigvee_{i=1}^{n} p_0(a_i), \forall y.\neg p_n(y)\} \cup \{\forall x.\neg p_{i-1}(x) \vee p_i(x) \mid i \in [1,n]\}$. A closed tableau can

⋆ Funded by FWF project W1255-N23.

[3] We recall that the cut rule consists in expanding a tableau by adding two branches with $\neg\phi$ and ϕ respectively, where ϕ is any formula (intuitively ϕ can be viewed as a lemma). A cut is atomic if ϕ is atomic.

be constructed by adding n copies of the clauses $\neg p_n(y^j)$ and $\neg p_{i-1}(x_i^j) \vee p_i(x_i^j)$ (for $i, j \in [1, n]$) and unifying all variables x_i^j and y^j with a_j. One gets a tableau of size $O(n^2)$. To make the proof more compact, we may merge the inferences applied for each a_j, since each of these constants are handled in the same way. This can be done by first applying the cut rule on the formula $\exists x.p_0(x)$. The branch corresponding the $\neg\exists x.p_0(x)$ can be closed by using the first clause. In the branch corresponding to $\exists x.p_0(x)$ a constant c is generated by skolemization and the branch can be closed by unifying x_i and y with c. This yields a tableau of size $O(n)$. Since it is hard to guess in advance whether such an application of the cut rule will be useful or not, we investigate another solution allowing the same proof compression. We represent the disjunction $\bigvee_{i=1}^{n} p_0(a_i)$ by a single literal $p_0(\alpha)$, together with a set of substitutions $\{[\alpha \backslash a_i] \mid i \in [1, n]\}$. Intuitively, this literal states that $p_0(\alpha)$ holds for some term α, and the given set of substitutions specifies the possible values of α. In the following, we call such variables α *abstraction variables*. The clauses are kept as compact as possible by grouping all literals with the same heads and in some cases inferences may be performed uniformly regardless of the value of α. In our example we get a tableau of size $O(n)$ by unifying x_i^j and y^j with α, this tableau may be viewed as a compact representation of an ordinary tableau, obtained by making n copies of the tree, with $\alpha = a_1, \ldots, a_n$. If we find out that an inference is applicable only for some specific value(s) of α (e.g., if one wants to close a branch by unifying $p_0(\alpha)$ with a clause $\neg p_0(a_1)$), then one may "separate" the literal by isolating some substitution (or sets of substitutions) before proceeding with the inference.

In this paper, we formalize these ideas into a tableau calculus called *M-tableau*. Basic inference rules are devised to construct M-tableaux and a strategy is provided to apply these rules efficiently, keeping the tableau as compact as possible. We prove that the procedure is sound and refutationally complete and that it may reduce the size of the proofs by an exponential factor. Our approach may be combined with all the usual refinements of the tableau procedure.[4]

2 Notations

We briefly review usual definitions (we refer to, e.g., [13] for details). Terms, atoms and clauses are built as usual over a (finite) set of function symbols Σ (including constants, i.e. nullary function symbols), an (infinite and countable) set of variables $\mathcal{V}$ and a (finite) set of predicate symbols Ω. The set of variables occurring in an expression (term, atom or clause) e is denoted by $\mathcal{V}(e)$. For readability, a term $f(t)$ is sometimes written ft. Ordinary (clausal) tableaux are trees labelled by literals and built by applying Expansion and Closure rules, the Expansion rule expands a leaf by n children labelled by literals $l_1, \ldots, l_n$, where a copy of $l_1 \vee \cdots \vee l_n$ occurs in the clause set at hand, and the Closure rule closes a branch by unifying the atoms of two complementary literals. A substitution

[4] Due to the limited number of pages, we omit proofs that are not necessary for the understanding of the method. A full version can be found in [10].

is a function (with finite domain) mapping variables to terms. A substitution mapping x_i to t_i (for $i \in [1, n]$) is written $[(x_1, \ldots, x_n)\backslash(t_1, \ldots, t_n)]$. The identity substitution (for $n = 0$) is denoted by id. The image of an expression e by a substitution σ is defined inductively as usual and written $e\sigma$.

3 A Shared Representation of Literals

We introduce the notion of an M-literal, that is a compact representation of a disjunction of ordinary literals with the same shape. The interest of this representation is that it will allow us to perform similar inferences in parallel on all these literals. We assume that $\mathcal{V}$ is partitioned into two (infinite) sets $\mathcal{V}_o$ and $\mathcal{V}_a$. The variables in $\mathcal{V}_o$ are ordinary variables. They may be either universally quantified variables in clauses, or rigid variables in tableaux. The variables in $\mathcal{V}_a$ are called *abstraction variables*. These are not variables in the standard sense, but can been seen rather as placeholders for a term that may take different values in different literals or branches. These variables will permit to share inferences applied on different literals. The set of ordinary variables (resp. abstraction variables) that occur in a term t is denoted by $\mathcal{V}_o(t)$ (resp. $\mathcal{V}_a(t)$). A *renaming* is an injective substitution σ such that $x \in \mathcal{V}_o \Rightarrow x\sigma \in \mathcal{V}_o$ and $\alpha \in \mathcal{V}_a \Rightarrow \alpha\sigma \in \mathcal{V}_a$.

Definition 1 (Syntax of M-Clauses). *An* M-literal *is either* **true** *or a triple* $\langle L, \bar{t}, \mathcal{S}\rangle$*, where:*

- *L is either a predicate symbol P or the negation of a predicate symbol $\neg P$,*
- *$\bar{t}$ is an n-tuple of terms, where n is the arity of P,*
- *and $\mathcal{S}$ is a set of substitutions σ with the same domain $D \subseteq \mathcal{V}_a(\bar{t})$ (by convention D is empty if $\mathcal{S} = \emptyset$) and such that $\mathcal{V}(\bar{t}\sigma) \cap D = \emptyset$.*

An M-clause *is a set of M-literals, often written as a disjunction.*

With a slight abuse of words, we will call the set D in the above definition the *domain of* $\mathcal{S}$ (denoted by $\mathrm{dom}(\mathcal{S})$). The semantics of M-clauses is defined by associating each M-literal with an ordinary clause (or **true**):

Definition 2 (Semantics of M-Clauses). *For every M-literal l, we denote by* $\mathrm{formula}(l)$ *the formula defined as follows (with the convention that empty disjunctions are equivalent to* **false***):*

$$\begin{aligned} \mathrm{formula}(\langle L, \bar{t}, \mathcal{S}\rangle) &\stackrel{def}{=} \bigvee_{\theta \in \mathcal{S}} L(\bar{t}\theta) \\ \mathrm{formula}(\mathbf{true}) &\stackrel{def}{=} \mathbf{true} \end{aligned}$$

For every M-clause C, we denote by $\mathrm{formula}(C)$ *the clause* $\bigvee_{l \in C} \mathrm{formula}(l)$*. For every set of M-clauses $\mathcal{C}$, we denote by* $\mathrm{formula}(\mathcal{C})$ *the formula (in conjunctive normal form)* $\bigwedge_{C \in \mathcal{C}} \mathrm{formula}(C)$*.*

We write $E \simeq E'$ iff $\mathrm{formula}(E) = \mathrm{formula}(E')$ *(up to the usual properties of $\vee$ and $\wedge$: associativity, commutativity and idempotence).*

Example 1. Let P be a unary predicate, Q be a binary predicate, c be a constant, f be a unary function, x be an ordinary variable, and α, β, γ be abstraction variables. The triples $l_1 = \langle P, \alpha, \{[\alpha \backslash f(c)]\}\rangle$ and

$$l_2 = \langle Q, (\beta, f(\gamma)), \{[(\beta, \gamma) \backslash (f(c), c)], [(\beta, \gamma) \backslash (c, f(c))]\}\rangle$$

are M-literals, and

$$\begin{array}{l} \text{formula}(l_1) = P(f(c)) \\ \text{formula}(l_2) = Q(f(c), f(c)) \vee Q(c, f(f(c))) \end{array}$$

The common shape $Q(\cdot, f(\cdot))$ is shared between the two literals in the second clause.

Remark 1. Observe that if $\mathcal{S} = \emptyset$ then $\text{formula}(\langle L, \bar{t}, \mathcal{S}\rangle) = \textbf{false}$, i.e. $\langle L, \bar{t}, \mathcal{S}\rangle$ denotes an empty clause. Moreover, any ordinary literal may be encoded as an M-literal where the set of substitutions is a singleton, e.g., the formula corresponding to $\langle P, (a, x), \{[\alpha \backslash a]\}\rangle$ is $P(a, x)$. Also, an M-literal $\langle L, \bar{t}, \{\sigma\}\rangle$ is always equivalent to $\langle L, \bar{t}\sigma, \{id\}\rangle$.

The application of a substitution σ to an M-literal is defined as follows:

$$\begin{array}{l} (\textbf{true})\sigma \stackrel{\text{def}}{=} \textbf{true} \\ \langle L, \bar{t}, \mathcal{S}\rangle\sigma \stackrel{\text{def}}{=} \langle L, \bar{t}\sigma', \{\theta\sigma \mid \theta \in \mathcal{S}\}\rangle \end{array}$$

where σ' denotes the restriction of σ to the variables not occurring in $\text{dom}(\mathcal{S})$.

Example 2. Let $l = \langle P, (\alpha, x), \{[\alpha \backslash x], [\alpha \backslash y]\}\rangle$ and $\sigma = [x \backslash a]$. Then:

$$l\sigma = \langle P, (\alpha, a), \{[\alpha \backslash a], [\alpha \backslash y]\}\rangle$$

Let $l' = \langle Q, (\alpha), \{[\alpha \backslash a], [\alpha \backslash b]\}\rangle$ and $\theta = [\alpha \backslash a]$. Then $l'\theta = l'$.

Proposition 1. *Let $l = \langle L, \bar{t}, \mathcal{S}\rangle$ be an M-literal. If $\text{dom}(\mathcal{S}) = \emptyset$, then one of the following conditions hold:*

- $\mathcal{S} = \emptyset$ *and* $\text{formula}(l) = \textbf{false}$*;*
- $\mathcal{S} = \{id\}$ *and* $\text{formula}(l) = L(\bar{t})$*.*

Proof. The identity is the only substitution with empty domain.

A given ordinary clause may be represented by many different M-clauses, for instance $P(a) \vee P(b)$ may be represented as $\langle P, (a), \{id\}\rangle \vee \langle P, (b), \{id\}\rangle$ or $\langle P, (\alpha), \{[\alpha \backslash a], [\alpha \backslash b]\}\rangle$, or even $\langle P, (\alpha), \{[\alpha \backslash a], [\alpha \backslash b]\}\rangle \vee \langle Q, (\beta), \emptyset\rangle$. In practice it is preferable to start with a representation in which useless literals are deleted and in which the remaining literals are grouped when possible. This motivates the following:

Definition 3. *An M-clause C is in* bundled normal form *(short:* BNF*) if it satisfies the following conditions.*

– *For every M-literal $\langle L, \bar{t}, \mathcal{S}\rangle \in C$, $\mathcal{S} \neq \emptyset$.*
– *If* $\mathbf{true} \in C$ *then* $C = \{\mathbf{true}\}$.
– *For all distinct literals $\langle L_1, \bar{t}_1, \mathcal{S}_1\rangle, \langle L_2, \bar{t}_2, \mathcal{S}_2\rangle \in C$, L_1 is distinct from L_2.*

An M-clause set $\mathcal{C}$ is in BNF *if all M-clauses of $\mathcal{C}$ are in* BNF.

Example 3. Let

$$l_1 \stackrel{\text{def}}{=} \langle P, \alpha, \{[\alpha \backslash f(c)]\}\rangle,$$

$$l_2 \stackrel{\text{def}}{=} \langle P, \beta, \{[\beta \backslash f(c)]\}\rangle,$$

$$l_3 \stackrel{\text{def}}{=} \langle Q, (\beta, f\gamma), \{[(\beta,\gamma)\backslash(f(c), c)], [(\beta,\gamma)\backslash(c, f(c))]\}\rangle,$$

and

$$l_4 \stackrel{\text{def}}{=} \langle Q, (\beta, \gamma), \{[(\beta,\gamma)\backslash(f(c), c)], [(\beta,\gamma)\backslash(c, f(c))]\}\rangle$$

be M-literals. The M-clause $\{l_3, l_4\}$ is not in BNF while the M-clauses $\{l_1, l_4\}$ and $\{l_2, l_4\}$ are in BNF.

Definition 4. *An M-clause C is* well-formed *if for all distinct literals $l = \langle L_1, \bar{t}_1, \mathcal{S}_1\rangle$ and $m = \langle L_2, \bar{t}_2, \mathcal{S}_2\rangle$ in C,* $\mathrm{dom}(\mathcal{S}_1) \cap \mathrm{dom}(\mathcal{S}_2) = \emptyset$.

Example 4. Consider the two M-clauses $C_1 := \{l_1, l_4\}$ and $C_2 := \{l_2, l_4\}$ of Example 3. C_1 is well-formed, C_2 is not well-formed. By renaming, C_2 can be transformed into C_1.

It is clear that every M-clause can be transformed into an equivalent well-formed M-clause by renaming. In the following, we shall implicitly assume that all the considered M-clauses are well-formed.

Lemma 1. *Let* F *be a formula in conjunctive normal form. Then there is an M-clause set $\mathcal{C}$ in* BNF *such that* $\mathrm{formula}(\mathcal{C}) \simeq$ F.

Example 5. Consider the clause $\{l_3, l_4\}$ of Example 3. It can be written in BNF as $\langle Q, (\beta, \gamma), \mathcal{S}\rangle$, where $\mathcal{S}$ denotes the following set of substitutions:

$$\{[(\beta,\gamma)\backslash(f(c), f(c))], [(\beta,\gamma)\backslash(c, f(f(c)))], [(\beta,\gamma)\backslash(f(c), c)], [(\beta,\gamma)\backslash(c, f(c))]\}$$

4 A Tableaux Calculus for M-Clauses

In this section, we devise a tableaux calculus for refuting sets of M-clauses. This calculus is defined by a set of inference rules, that, given an existing tableau $\mathcal{T}$, allow one to:

1. Expand a branch with new children, by introducing a new copy of an M-clause of the set at hand.
2. Instantiate some of the (rigid) variables occurring in the tableau.
3. Separate shared literals inside an M-clause, so that different inferences can be applied on each of the corresponding branches. The rule can be applied on nodes that are not leaves.

Steps 1 and 2 are standard, but Step 3 is original.

Definition 5 (Pre-Tableau). *A* pre-tableau *is a tree $\mathcal{T}$ where vertices are labelled by M-literals or by* **false**. *We call the direct successors of a node its* children. *The* root *is the (unique) node that is not a child of any node in $\mathcal{T}$ and a* leaf *is a node with no child. A* path *P is a sequence of nodes $(\nu_1, \ldots, \nu_n)$ such that ν_{i+1} is a child of ν_i for $i \in [1, n-1]$. Furthermore, we call ν_1 the* initial node *of P and ν_n the* last node *of P. A* branch *is a path such that the initial node is the root and the last node is a leaf. With a slight abuse of words we say that a branch contains an M-literal l if it contains a node labelled by l.*

The descendants *of a node ν are inductively defined as ν and the descendants of the children of ν. The* subtree of root ν in $\mathcal{T}$ *is the subtree consisting of all the descendants of ν, as they appear in $\mathcal{T}$.*

If ν is a non-leaf node with exactly $n > 0$ children $\nu_1, \ldots, \nu_n$ labelled by M-literals $l_1, \ldots, l_n$ respectively, then the formula associated with *ν is defined as: $\bigvee_{i=1}^{n} \text{formula}(l_i)$.*

We say that an M-literal $\langle L, \bar{t}, \mathcal{S} \rangle$ (resp. a node ν labelled by $\langle L, \bar{t}, \mathcal{S} \rangle$) introduces an abstraction variable *α if $\alpha \in \text{dom}(\mathcal{S})$ (as shown in Proposition 2, the abstraction variables are introduced by exactly one M-literal or node in an M-tableau).*

Definition 6. *Let $\mathcal{T}$ be a pre-tableau and σ be a substitution. Then $\mathcal{T}\sigma$ denotes the result of applying σ to all M-literals labelling the nodes of $\mathcal{T}$.*

Definition 7 (Tableau for a Set of M-Clauses). *An* M-tableau *$\mathcal{T}$ for a set of M-clauses $\mathcal{C}$ is a pre-tableau built inductively by applying the rules Expansion, Instantiation and Separation to an initial tableau containing only one node, labelled by* **true** *(also called the* initial M-literal*).*

In the following, the word "tableau" always refers to an M-tableau, unless specified otherwise (we use the expression "ordinary tableau" for standard ones).

The rules are defined as follows (in each case, $\mathcal{T}$ denotes a previously constructed tableau for a set of M-clauses $\mathcal{C}$).

Expansion Rule. Let λ be a leaf of $\mathcal{T}$, and C be an element of $\mathcal{C}$ not containing **true**. Let C' be a copy of C where all variables that occur also in $\mathcal{T}$ are renamed such that C' share no variable[5] with $\mathcal{T}$. The pre-tableau $\mathcal{T}'$ constructed by adding a new child labelled by l to λ for each $l \in C'$ is a tableau for $\mathcal{C}$.

Instantiation Rule. Let t be a term and $x \in \mathcal{V}_o$ such that for all nodes ν, ν', if ν is labelled by an M-literal containing x and ν' introduces an abstraction variable $\alpha \in \mathcal{V}_a(t)$, then ν is a proper descendant of ν'. Then $\mathcal{T}[x \backslash t]$ is a tableau for $\mathcal{C}$.

Remark 2. Observe that if t contains no abstraction variables then the condition always holds, since no node ν' satisfying the above property exists. In practice, the Instantiation rule should of course not be applied with an arbitrary variable

[5] Note that both ordinary and abstraction variables are renamed.

and term. Unification will be used instead to find the most general instantiations closing a branch. A formal definition will be given later (see Definition 11).

Example 6. Let

be two tableaux for some set of M-clauses $\mathcal{C}$. The pre-tableau $\mathcal{T}_1[x \backslash \alpha]$ is not a tableau, because x is substituted by a term containing an abstraction variable α, and x occurs above the literal introducing α. On the other hand, $\mathcal{T}_2[x \backslash \alpha]$ is a tableau.

Separation Rule. The rule is illustrated in Figure 1 towards Figure 3. Let ν be a non-leaf node of $\mathcal{T}$. Let μ be a child of ν, labelled by $l = \langle L, \bar{t}, \mathcal{S} \rangle$. Let $\bar{r} = \bar{t}\theta$ be an instance of $\bar{t}$, with $\mathrm{dom}(\theta) = \mathrm{dom}(\mathcal{S})$ and $\mathcal{V}_a(\bar{t}\theta) \cap \mathrm{dom}(\mathcal{S}) = \emptyset$. Let $\mathcal{S}_1$ be the set of substitutions $\sigma \in \mathcal{S}$ such that there exists a substitution σ' with $\bar{t}\sigma = \bar{r}\sigma'$ and every variable in $\mathrm{dom}(\sigma')$ is an abstraction variable not occurring in $\mathcal{T}$, and let $\mathcal{S}_2 \stackrel{\text{def}}{=} \mathcal{S} \setminus \mathcal{S}_1$. Assume that $\mathcal{S}_1 \neq \emptyset$. We define the new literal $l' \stackrel{\text{def}}{=} \langle L, \bar{r}, \{\sigma' \mid \sigma \in \mathcal{S}_1\} \rangle$. The Separation rule is defined as follows:

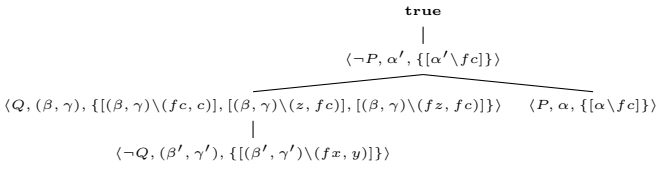
1. We apply the substitution θ to $\mathcal{T}$[6] .
2. We replace the label l of μ by l'.
3. We add a new child to the node ν, labelled by a literal $\langle L, \bar{t}, \mathcal{S}_2 \rangle$.

Observe that if $\mathcal{S}_2 = \emptyset$ then $\mathrm{formula}(\langle L, \bar{t}, \mathcal{S}_2 \rangle) = \mathbf{false}$ hence the third step may be omitted, since the added branch is unsatisfiable anyway. The rule does not apply if $\mathcal{S}_1$ is empty.

Example 7. Let

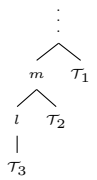
$$
\begin{aligned}
l_1 &:= \langle P, \alpha, \{[\alpha \backslash fc]\} \rangle, \\
l_2 &:= \langle \neg P, \alpha', \{[\alpha' \backslash fc]\} \rangle, \\
l_3 &:= \langle \neg Q, (\beta', \gamma'), \{[(\beta', \gamma') \backslash (fx, y)]\} \rangle, \text{ and} \\
l_4 &:= \langle Q, (\beta, \gamma), \{[(\beta, \gamma) \backslash (fc, c)], [(\beta, \gamma) \backslash (z, fc)], [(\beta, \gamma) \backslash (fz, fc)]\} \rangle
\end{aligned}
$$

be M-literals and $\mathcal{C} = \{\{l_1, l_4\}, \{l_2\}, \{l_3\}\}$ be an M-clause set in BNF. Applying three times the Expansion rule, we can derive the tableau



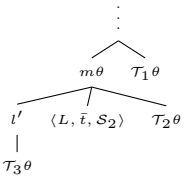
[6] Actually, due to the above conditions, the variables in $\mathrm{dom}(\theta)$ only occur in the subtree of root μ, hence θ only affects this subtree.

Fig. 1. The initial subtree in the Separation rule.



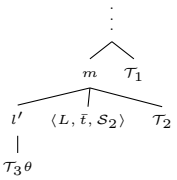
where m is the label of node ν and $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}_3$ are possibly empty subtrees.

Fig. 2. The subtree after an application of the Separation rule.



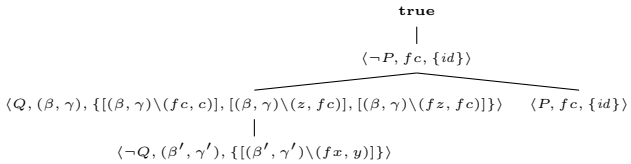
where m is the label of node ν and $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}_3$ are possibly empty subtrees.

Fig. 3. The subtree without redundant substitutions after an application of the Separation rule.

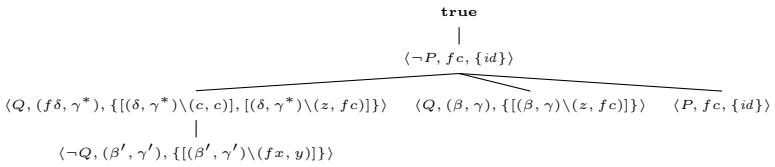


where m is the label of node ν and $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}_3$ are possibly empty subtrees.

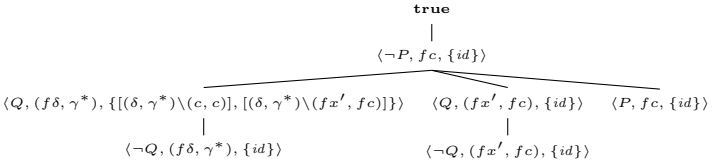
Now, we apply two times the Separation rule. First we choose m of Figure 1 to be **true** and l to be $\langle \neg P, \alpha', \{[\alpha' \backslash fc]\} \rangle$, where the substitution θ is $[\alpha' \backslash fc]$ (hence we get $\mathcal{S}_1 = \{id\}$). Afterwards, we choose analogously $\langle \neg P, fc, \{id\} \rangle$ (which is the result of the first application) and $\langle P, \alpha, \{[\alpha \backslash fc]\} \rangle$, with the substitution $[\alpha \backslash fc]$. Both times, the tuple $\bar{r}$ of the Separation rule is fc (with $\mathcal{S}_2 = \emptyset$ in both cases). This leads to the tableau:



Afterwards, we again apply the Separation rule, to modify the node labelled with l_4 where $\bar{r} = (f\delta, \gamma^*)$ and $\mathcal{S}_2 = \{[(\beta, \gamma) \backslash (z, fc)]\}$.



After some further applications of the Separation and Expansion rules, we are able to construct the following tableau by applying the Instantiation rule with the substitutions $[z \backslash fx'], [y \backslash fc], [x \backslash \delta], [y \backslash \gamma^*]$.



5 Soundness

Definition 8. *Let $\mathcal{T}$ be a pre-tableau or a tableau. A branch B of $\mathcal{T}$ is* closed *if it contains* **false** *or two nodes labelled by literals $\langle L_1, \bar{t}_1, \mathcal{S}_1\rangle, \langle L_2, \bar{t}_2, \mathcal{S}_2\rangle$ such that $\bar{t}_1 = \bar{t}_2$, $L_1 = P$, $L_2 = \neg P$ for some predicate symbol P. The (pre)-tableau $\mathcal{T}$ is* closed *iff all branches of $\mathcal{T}$ are closed.*

Example 8. The final tableau of Example 7 contains three branches, i.e.

$\{\langle \neg P, fc, \emptyset\rangle, \langle Q, (f\delta, \gamma^*), \{[(\delta, \gamma^*)\backslash(c, c)], [(\beta, \gamma)\backslash(c, ffx')]\}\rangle, \langle \neg Q, (f\delta, \gamma^*), \emptyset\rangle\}$,
$\{\langle \neg P, fc, \emptyset\rangle, \langle Q, (fx', fc), \emptyset\rangle, \langle \neg Q, (fx', fc), \emptyset\rangle\}$, and
$\{\langle \neg P, fc, \emptyset\rangle, \langle P, fc, \emptyset\rangle\}$.

All of them are closed and so the tableau is closed. Observe that the inferences closing the branches corresponding to the literals $Q(f(c), c)$ and $Q(f(z), f(c))$ in $\{l_1, l_4\}$ are shared in the constructed tableau (both branches are closed by introduced suitable instances of l_3), whereas the literal $Q(z, f(c))$ is handled separately (by instantiating z by $f(x')$ and using yet another instance of l_3).

Proposition 2. *Let $\mathcal{T}$ be a tableau. If ν_1 and ν_2 are distinct nodes in $\mathcal{T}$, labelled by the M-literals $\langle L_1, \bar{t}_1, \mathcal{S}_1\rangle$ and $\langle L_2, \bar{t}_2, \mathcal{S}_2\rangle$ respectively, then $\mathcal{S}_1$ and $\mathcal{S}_2$ have disjoint domains.*

Proposition 3. *Let $\mathcal{T}$ be a tableau for a set of M-clauses $\mathcal{C}$. For every non-leaf node ν in $\mathcal{T}$, the formula associated with ν (as defined in Definition 5) is an instance of a formula* formula(C)*, where C is a renaming of an M-clause in $\mathcal{C}$.*

Proof. It suffices to show that all the construction rules preserve the desired property.

- **Expansion.** The property immediately holds for the nodes on which the rule is applied, by definition of the rule. The other nodes are not affected.
- **Instantiation.** By definition, the formula associated with a node ν in the final tableau is an instance of the formula associated with ν in the initial one. Thus the property holds.
- **Separation.** The nodes occurring outside of the subtree of root μ are not affected. By definition, the formula associated with the descendants of μ in the new tableau are instances of formulas associated with nodes of the initial tableau. Thus it only remains to consider the node ν. The formula associated with ν in the final tableau is obtained from that of the initial one by removing the formula corresponding to an M-literal $l = \langle L, \bar{t}, \mathcal{S}\rangle$ and replacing it by formula$(l') \vee$ formula$(\langle L, \bar{t}, \mathcal{S}_2\rangle)$. Since $l' = \langle L, \bar{r}, \{\sigma' \mid \sigma \in \mathcal{S}_1\}\rangle$, we have formula$(l') = \bigvee_{\sigma\in\mathcal{S}_1} L(\bar{r}\sigma') = \bigvee_{\sigma\in\mathcal{S}_1} L(\bar{t}\sigma)$ (since $\bar{t}\sigma = \bar{r}\sigma'$ by definition of the Separation rule). Thus formula$(l') \vee$ formula$(\langle L, \bar{t}, \mathcal{S}_2\rangle) =$ formula$(\langle L, \bar{t}, \mathcal{S}\rangle)$ and the proof is completed.

Proposition 4. *Let $\mathcal{T}$ be a tableau for $\mathcal{C}$. Let $\alpha \in \mathcal{V}_a$ be a variable introduced in a node ν and assume that α occurs in an M-literal labelling a node μ. Then μ is a descendant of ν.*

Lemma 2. *Let $\mathcal{T}$ be a closed tableau for $\mathcal{C}$ and let $\mathcal{T}'$ be the tableau after applying once the Separation rule to a node ν of $\mathcal{T}$ with a child μ labelled with l. Then there is at most one branch B in $\mathcal{T}'$ that is not closed. Moreover, this branch necessarily contains the node labelled by $\langle L, \bar{t}, \mathcal{S}_2 \rangle$ (see Figure 3). In particular, if $\mathcal{S}_2$ is empty then $\langle L, \bar{t}, \mathcal{S}_2 \rangle$ is* **false** *and $\mathcal{T}'$ is closed.*

Theorem 1 (Soundness). *If a set of M-clauses $\mathcal{C}$ admits a closed tableau $\mathcal{T}$ then $\mathcal{C}$ is unsatisfiable.*

Proof. We prove soundness by transforming $\mathcal{T}$ into a closed tableau that contains only M-literals where the substitution set is $\{id\}$ or $\emptyset$. Due to Proposition 3, the resulting tableau then corresponds to an ordinary tableau. The soundness of ordinary tableau then implies the statement.

We transform the tableau $\mathcal{T}$ by an iterative procedure. We always take an arbitrary topmost node ν labelled with an M-literal $l = \langle L, \bar{t}, \mathcal{S} \rangle$ where $\mathcal{S} \neq \{id\}$ and $\mathcal{S} \neq \emptyset$. Then we consider a substitution $\theta \in \mathcal{S}$ and we apply the Separation rule with the tuple $\bar{t}\theta$. We have $\bar{t}\theta = \bar{t}\theta id$ and if $\sigma \in \mathcal{S} \setminus \{\theta\}$, then $\bar{t}\sigma \neq \bar{t}\theta$, hence there is no substitution σ' with $\bar{t}\sigma\sigma' = \bar{t}\theta$, such that $\mathrm{dom}(\sigma')$ only contains fresh variables. Consequently, the rule splits $\mathcal{S}$ into a singleton $\mathcal{S}_1 = \{\theta\}$ and $\mathcal{S}_2 = \mathcal{S}_1 \setminus \{\theta\}$. The literal l gets replaced by $l' = \langle L, \bar{t}\theta, \{id\} \rangle$ and we add the node μ labelled with $\langle L, \bar{t}, \mathcal{S}_2 \rangle$. By Lemma 2, there is at most one non-closed branch, i.e. the branch ending with the node μ is the only open branch. We consider a copy $\mathcal{T}_\nu$ of the subtree of root ν in $\mathcal{T}$, renaming all variables introduced in $\mathcal{T}_\nu$ by fresh variables. We replace the root node of $\mathcal{T}_\nu$ by $\langle L, \bar{t}, \mathcal{S}_2 \rangle$ and replace the subtree of root μ in the tableau by $\mathcal{T}_\nu$. It is easy to check that the obtained tableau is a closed tableau for $\mathcal{C}$. Furthermore, the length of the branches does not increase, the number of non-empty substitutions occurring in the M-literals does not increase, and it decreases strictly in l' and $\langle L, \bar{t}, \mathcal{S}_2 \rangle$. This implies that the multiset of multisets $\{|\{\sigma \in \mathcal{S}'_1 \mid \sigma \neq id\}|, \ldots, |\{\sigma \in \mathcal{S}'_n \mid \sigma \neq id\}|\}$ of natural numbers, where $\{\langle L_i, \bar{t}_i, \mathcal{S}'_i \rangle \mid i \in [1, n]\}$ is a branch in $\mathcal{T}$ is strictly decreasing according to the multiset extension of the usual ordering. Since this ordering is well-founded, the process eventually terminates, and after a finite number of applications of this procedure we get a tableau only containing nodes labelled with M-literals whose substitution set is equal to $\{id\}$ or $\emptyset$.

Remark 3. As a by-product of the proof, we get that the size of the minimal ordinary tableau for a clause set formula($\mathcal{C}$) is bounded exponentially by the size of any closed tableau for $\mathcal{C}$. Indeed, we constructed an ordinary tableau from an M-tableau $\mathcal{T}$ in which every branch $(l_1, \ldots, l_n)$ in $\mathcal{T}$ is replaced by (at most) k^n branches, where k is the maximal number of substitutions in l_i. In Section 7, we shall prove that this bound is precise, i.e. that our tableau calculus allows exponential reduction of proof size w.r.t. ordinary (cut-free) tableaux.

6 Completeness

Proving completeness of M-tableau is actually a trivial task, since one could always apply the Separation rule in a systematic way on all M-literals to trans-

form them into ordinary literals (as it is done in the proof of Theorem 1), and then get the desired result by completeness of ordinary tableau. However, this strategy would not be of practical use. Instead, we shall devise a strategy that keeps the M-tableau as compact as possible and at the same time allows one to "simulate" any application of the ordinary expansion rules. In this strategy, the Separation rule is applied on demand, i.e. only when it is necessary to close a branch. No hypothesis is assumed on the application of the ordinary expansion rules, therefore the proposed strategy is "orthogonal" to the usual refinements of ordinary tableaux, for instance connection tableaux[7] [12] or hyper-tableaux[8] [1]. Thus our approach can be combined with any refutationally complete tableau procedure [6].

The main idea denoted by *simulate a strategy* is to do the same steps as in ordinary tableau, while keeping M-clauses as compressed as possible. If ordinary tableau expands the tableau by a clause, we expand the tableau with the corresponding M-clause, and if a branch is closed in the ordinary tableau, then the corresponding branch is closed in the M-tableau. This last step is not trivial: Given two ordinary literals $P(\bar{t})$ and $\neg P(\bar{r})$ the ordinary tableau might compute the most general unifier (mgu) of $\bar{t}$ and $\bar{r}$. But in the presented formalism, the two literals might not appear as such, i.e. there are no literals $m = \langle P, \bar{t}, \{id\}\rangle$ and $k = \langle \neg P, \bar{r}, \{id\}\rangle$. In general, there are only M-literals $m' = \langle P, \bar{t}', \mathcal{S}_1\rangle$ and $k' = \langle \neg P, \bar{r}', \mathcal{S}_2\rangle$ such that $\bar{t}'\theta = \bar{t}$ and $\bar{r}'\vartheta = \bar{r}$, where θ and ϑ denote the compositions of the substitutions occurring in the M-literals in the considered branch. Note also that, although $\bar{t}'$ and $\bar{r}'$ are unifiable, the Instantiation rule cannot always be applied to unify them and close the branch. Indeed, the domain of the mgu may contain abstraction variables, whereas the Instantiation rule only handles universal variables. For showing completeness, it would suffice to apply the Separation rule on each ancestor of k' and m' involved in the definition of θ or ϑ, to create a branch where the literals m and k appear explicitly. Thereby, we would lose a lot of the formalisms benefit. Instead, we shall introduce a strategy that uses the Separation rule only if this is necessary for making the unification of $\bar{t}'$ and $\bar{r}'$ feasible (by mean of the Instantiation rule). Such applications of the Separation rule may be seen as preliminary steps for the Instantiation rule. This follows the maxim to stay as general as possible because a more general proof might be more compact.

In the formalisation of the Instantiation rule we ensured soundness by allowing abstraction variables only to occur in descendants of the literal that introduced the variable. This has a drawback to our strategy: The unification process that we try to simulate can ask for an application of the Instantiation rule which would cause a violation of this condition for abstraction variables if

[7] Connection tableaux can be seen as ordinary tableaux in which any application of the Expansion rule must be followed by the closure of a branch, using one of the newly added literals and the previous literal in the branch.

[8] Hyper-tableaux may be viewed in our framework as ordinary tableaux in which the Expansion rule must be followed by the closure of all the newly added branches containing negative literals.

we follow the procedure in the former paragraph. We thus have to add further applications of the Separation rule to ensure that this condition is fulfilled.

Definition 9. *Let $B = (\nu_0, \nu_1, \ldots, \nu_n)$ be a path in a tableau $\mathcal{T}$ where ν_0 is the initial node of $\mathcal{T}$ and each node ν_i (with $i > 0$) is labelled by $\langle L_i, \bar{t}_i, \mathcal{S}_i \rangle$.*

An abstraction substitution *for B is a substitution $\eta_n \ldots \eta_1$ with $\eta_i \in \mathcal{S}_i$, for $i = 1, \ldots, n$.*

A conflict *in a branch B is a pair $(\bar{t}_i, \bar{t}_j)$ with $i, j \in [1, n]$, L_i and L_j are dual and $\bar{t}_i$ and $\bar{t}_j$ are unifiable. A conflict is* η-realizable *if η is an abstraction substitution for B such that $\bar{t}_i\eta$ and $\bar{t}_j\eta$ are unifiable.*

In practice, we do not have to check that a conflict is realizable (this would be costly since we have to consider exponentially many substitutions).

If $(\bar{t}, \bar{t}')$ is a conflict then $\bar{t}$ and $\bar{t}'$ are necessarily unifiable, with some mgu θ. As mentioned before, this does not mean that a branch with conflict can be closed. Moreover, according to the restriction on the Instantiation rule, a variable x cannot be instantiated by a term containing an abstraction variable α, if x occurs in some ancestor of the literal introducing α in the tableau. This motivates the following:

Definition 10. *A variable $\alpha \in \mathcal{V}_a$ is* blocking *for a conflict $(\bar{t}, \bar{t}')$, where $\theta = mgu(\bar{t}, \bar{t}')$ if $\alpha \in \mathrm{dom}(\theta)$ or α occurs in a term $x\theta$, where $x \in \mathcal{V}_o$ and x occurs in a literal labelling an ancestor of the node introducing α.*

Finally, we introduce a specific application of the Separation rule which allows one either to "isolate" some literals in order to ensure that they have a specific "shape" (as specified by a substitution), or to eliminate abstraction variables completely if needed.

Definition 11. *If σ is a substitution, we denote by $\mathrm{dom}_a(\sigma)$ the set of variables $\alpha \in \mathcal{V}_a$ such that $\alpha\sigma \notin \mathcal{V}_a$.*

A tableau is compact *if it is constructed by a sequence of applications of the tableau rule in which:*

- *The Instantiation rule is applied only if the tableau contains a branch with a conflict $(\bar{t}, \bar{t}')$, with no blocking variable. Each variable $x \in \mathrm{dom}(\sigma)$ is replaced by a term $t\sigma$, where $\sigma = mgu(\bar{t}, \bar{t}')$ (since there is no blocking variable it is easy to check that the conditions on the Instantiation rule are satisfied). Afterwards, it is clear that the branch is closed.*
- *The Separation rule is applied on a node labelled by $\langle L, \bar{t}, \mathcal{S} \rangle$, using a substitution θ only if there exists a conflict $(\bar{t}, \bar{t}')$ with $\sigma = mgu(\bar{t}, \bar{t}')$ such that one of the following conditions holds:*
  1. *$\mathrm{dom}(\mathcal{S})$ contains a blocking variable in $\mathrm{dom}_a(\sigma)$ and θ is defined as follows: $\mathrm{dom}(\theta) = \mathrm{dom}(\mathcal{S}) \cap \mathrm{dom}_a(\sigma)$, $x\theta = x\sigma$ if $x\sigma$ is a variable, otherwise $x\theta$ is obtained from $x\sigma$ by replacing all variables by pairwise distinct fresh abstraction variables.*
  2. *Or $\mathrm{dom}(\mathcal{S})$ contains a blocking variable not occurring in $\mathrm{dom}_a(\sigma)$, and $\theta \in \mathcal{S}$.*

The applications of the Separation rule in Definition 11 are targeted at making the closure of the branch possible by getting rid of blocking variables, while keeping the tableau as compact as possible (thus useless separations are avoided).

Example 9. For instance, assume that we want to close a branch containing two literals $l = \langle L, (\alpha), \{[\alpha\backslash f(a)], [\alpha\backslash f(b)], [\alpha\backslash a]\}\rangle$ and $l' = \langle \neg L, f(x), \{id\}\rangle$. To this aim, we need to ensure that α is unifiable with $f(x)$. This is done by applying the separation rule (Case 1 of Definition 11) with the substitution $[\alpha\backslash f(\beta)]$, so that α has the desired shape. This yields: $\langle L, (f(\beta)), \{[\beta\backslash a], [\beta\backslash b]\}\rangle$ Afterwards, if x does not occur before l in the branch then the branch is closed by unifying x with β. If x occurs before l, then this is not feasible since this would contradict the condition on the Instantiation rule, and we have to apply the Separation rule again (Case 2) to eliminate β, yielding (for instance) $\langle L, (f(a)), \{id\}\rangle$. The direct application of the Separation rule with, e.g., $\theta = [\alpha\backslash f(a)]$ is forbidden in the strategy.

Theorem 2 (Completeness). *Let* $\mathtt{C}$ *be a clause set and* $\mathcal{C}$ *be an* M*-clause set with* formula$(\mathcal{C}) \simeq \mathtt{C}$*. If* $\mathtt{C}$ *is unsatisfiable, then there is a closed compact tableau for* $\mathcal{C}$*.*

Proof. For an M-clause $C_i \in \mathcal{C}$, we denote by $\mathtt{C}_i$ the ordinary clause in $\mathtt{C}$ with formula$(\mathtt{C}_i) \simeq$ formula(C_i). W.l.o.g. we can assume that clauses do not appear twice, neither in $\mathcal{C}$ nor in $\mathtt{C}$. Now, we can perform a proof search based on ordinary tableau starting with $\mathtt{C}$ (using any complete strategy) yielding a closed tableau, built by applying the usual Expansion and Closure rules. In the following, $\mathtt{T}$ will denote an already constructed ordinary tableau for $\mathtt{C}$ and $\mathcal{T}$ will represent the corresponding M-tableau for $\mathcal{C}$. More precisely, the tableau $\mathcal{T}$ is constructed in such a way that there exists an injective mapping h from the nodes in $\mathcal{T}$ to those in $\mathtt{T}$, a function $\nu \mapsto \eta_\nu$ mapping each node ν in $\mathcal{T}$ labelled by some literal $\langle L, \bar{t}, \mathcal{S}\rangle$ to an abstraction substitution η and a substitution ϑ (with $\mathrm{dom}(\vartheta) \subseteq \mathcal{V}_o$) such that the following property holds (denoted by $(\star)$):

1. The root of $\mathcal{T}$ is mapped to the root of $\mathtt{T}$.
2. If ν is a child of ν' then $h(\nu)$ is a child of $h(\nu')$.
3. For any path $(\nu_0, \ldots, \nu_n)$, if ν_n is labelled by an M-literal $\langle L, \bar{t}, \mathcal{S}\rangle$, then $h(\nu_n)$ is labelled by a literal $L(\bar{t})\eta_{\nu_n} \ldots \eta_{\nu_1}\vartheta$.
4. If a branch of the form $(h(\nu_0), \ldots, h(\nu_n))$ is closed in $\mathtt{T}$, then $(\nu_0, \ldots, \nu_n)$ is closed in $\mathcal{T}$.

Note that the mapping is not surjective in general ($\mathtt{T}$ may be bigger than $\mathcal{T}$). By $(\star)$, if $\mathtt{T}$ is closed then $\mathcal{T}$ is also closed, which gives us the desired result. It is easy to check that applying the Separation rule on a node ν_i in a branch $(\nu_0, \ldots, \nu_n)$ in $\mathcal{T}$ preserves $(\star)$, provided it is applied using a substitution θ (as defined in the Separation rule) that is more general than $\eta_{\nu_n} \circ \cdots \circ \eta_{\nu_1}$.

The tableau $\mathcal{T}$ is constructed inductively as follows. For the base case, we may take $\mathcal{T} = \mathtt{T} =$ **true** and $(\star)$ trivially holds.

Expansion: The Expansion rule of ordinary tableaux allows one to expand the tableau by an arbitrary clause $\mathtt{C}_i$ of $\mathtt{C}$. We define the corresponding tableau

for $\mathcal{C}$ as the tableau expanded by C_i. The mappings h and η_ν may be extended in a straightforward way so that $(\star)$ is preserved (the unique new node ν in $\mathcal{T}$ may be mapped to an arbitrary chosen new node in $\mathtt{T}$).

Closure: Assume that a branch in $\mathtt{T}$ is closed by applying some substitution σ, using two literals $L(\bar{t})$ and $\neg L(\bar{r})$, where $\sigma = \mathrm{mgu}(\bar{t}, \bar{r})$. If one of these literals do not occur in the image of a branch of $\mathcal{T}$ then it is clear that the operation preserves $(\star)$ (except that the substitution ϑ is replaced by $\vartheta\sigma$), hence no further transformation is required on $\mathcal{T}$. Otherwise, by $(\star)$, there is a branch $B = (\nu_0, \ldots, \nu_n)$ in $\mathcal{T}$ where for every $i \in [1, n]$, ν_i is labelled by $l_i = \langle L_i, \bar{t}'_i, \mathcal{S}_i \rangle$, two numbers $j, k \in \mathbb{N}$ such that $L_j = L$, $L_k = \neg L$, $\bar{t}'_j \eta\vartheta = \bar{t}$ and $\bar{t}'_k \eta\vartheta = \bar{r}$, with $\eta = \eta_{\nu_n} \ldots \eta_{\nu_1}$.

By definition, $(\bar{t}'_j, \bar{t}'_k)$ is an η-realizable conflict. Let σ' be the mgu of $\bar{t}'_j$ and $\bar{t}'_k$. If there is no blocking variable for $(\bar{t}'_j, \bar{t}'_k)$, then the Instantiation rule applies, replacing every variable $x \in \mathrm{dom}(\sigma')$ by $x\sigma'$, and the branch may be closed. By definition $\eta\vartheta\sigma$ is a unifier of $\bar{t}$ and $\bar{r}$, hence $\eta\vartheta\sigma = \sigma'\theta'$, for some substitution θ'. By definition, the co-domain of η contains no abstraction variables, thus $\theta' = \eta\vartheta'$, with $\mathrm{dom}(\vartheta') \subseteq \mathcal{V}_o$, and $\vartheta\sigma = \sigma'\vartheta'$. The application of the rule preserves $(\star)$, where ϑ is replaced by the substitution ϑ'. Indeed, consider a node ν in $\mathcal{T}$, initially labelled by a literal $\langle L, \bar{t}, \mathcal{S} \rangle$, where $h(\nu)$ is labelled by $L(\bar{t})\eta_{\nu_n} \ldots \eta_{\nu_1}\vartheta$. After the rule application, ν is labelled by $\langle L, \bar{t}\sigma', \mathcal{S}\sigma' \rangle$, $h(\nu)$ is labelled by $L(\bar{t})\eta_{\nu_n} \ldots \eta_{\nu_1}\vartheta\sigma$, and the substitutions η_{ν_i} are replaced by $\eta_{\nu_i}\sigma'$. Since $\mathrm{dom}(\sigma') \cap \mathcal{V}_a = \emptyset$, it is clear that $\sigma'\eta_{\nu_n} \ldots \eta_{\nu_1}\sigma'\vartheta' = \eta_{\nu_n} \ldots \eta_{\nu_1}\sigma'\vartheta' = \eta_{\nu_n} \ldots \eta_{\nu_1}\vartheta\sigma$.

Otherwise, the set of blocking variables is not empty, and since all abstraction variables occurring in the tableau must be introduced in some node, there exists $l \in [1, n]$ such that $\mathrm{dom}(\mathcal{S}_l)$ contains a blocking variable. According to Definition 11, the Separation rule may be applied on ν_l (it is easy to check that all the application conditions of the rule are satisfied). In Case 1 (of Definition 11), θ is more general than σ' by definition, and since $\eta_{\nu_n} \circ \cdots \circ \eta_{\nu_1}\vartheta\sigma$ is a unifier of $\bar{t}'_j$ and $\bar{t}'_k$, the mgu σ' must be more general than $\eta_{\nu_n} \circ \cdots \circ \eta_{\nu_1}$. In Case 2, we can take $\theta = \eta_{\nu_l}$, which is more general than $\eta_{\nu_n} \circ \cdots \circ \eta_{\nu_1}$ (since the substitutions η_ν have disjoint domains). Thus the property $(\star)$ is preserved.

This operation is repeated until the set of blocking variables is empty, which allows us to apply the Instantiation rule as explained before. The process necessarily terminates since each application of the Separation rule either increases the size of the tableau (either by adding new nodes, or by instantiating a variable by a non variable term), or does not increase the size of the tableau but strictly reduces the number of abstraction variables. Furthermore, by $(\star)$, the size of $\mathcal{T}$ is smaller than that of $\mathtt{T}$.

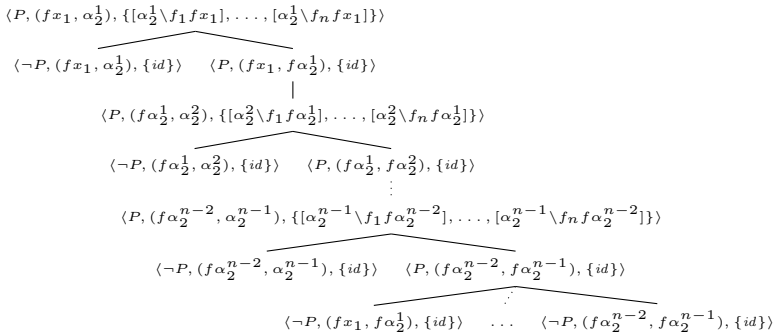
7 An Exponentially Compressed Tableau

In this section, we will show that the presented method is able to compress tableaux by an exponential factor. This corresponds to an introduction of a

single Π_2-cut[9] (see [9]). As a simplified measurement of the size of a tableau we consider the number of nodes. Let us consider the schema of M-clause sets $\mathcal{C}^n \stackrel{\text{def}}{=} \{\{l_1^n\}, \{l_2, l_3\}, \{l_4^n\}\}$ with

$$\begin{aligned}
l_1^n &= \langle P, \bar{\alpha}, [\bar{\alpha}\backslash(x, f_1 x)], \ldots, [\bar{\alpha}\backslash(x, f_n x)]\rangle \\
l_2 &= \langle \neg P, \bar{\beta}, [\bar{\beta}\backslash(x, y)]\rangle \\
l_3 &= \langle P, \bar{\gamma}, [\bar{\gamma}\backslash(x, fy)]\rangle \\
l_4^n &= \langle \neg P, \bar{\delta}, [\bar{\delta}\backslash(fx_1, fx_2)], \ldots, [\bar{\gamma}\backslash(fx_{n-1}, fx_n)]\rangle
\end{aligned}$$

where $\bar{\alpha} = (\alpha_1, \alpha_2), \bar{\beta} = (\beta_1, \beta_2), \bar{\gamma} = (\gamma_1, \gamma_2)$, and $\bar{\delta} = (\delta_1, \delta_2)$ for $n \in \mathbb{N}$. Then we can construct a closed tableau for $\mathcal{C}^n$ whose size is linear w.r.t. n:



The clause set schema $\mathcal{C}^n$ is a simplified variant of the example in [9, Section 3 and 9] and one can easily verify that an ordinary tableau method is of exponential size of n. Just consider the term instantiations which are necessary for an ordinary tableau:

$x \leftarrow \{fx_1, ff_{i_1}fx_1, \ldots, ff_{i_{n-2}}f \ldots f_{i_1}fx_1 | i_1, \ldots, i_{n-2} \in [1, n]\}$ in l_1^n,
$(x, y) \leftarrow \{(t, f_i t) | i \in [1, n] \wedge t$ is a substitution for x in $l_1^n\}$ in l_2 and l_3, and
$(x_1, \ldots, x_n) \leftarrow \{(x_1, f_{i_1}fx_1, \ldots, f_{i_{n-1}}f \ldots f_{i_1}fx_1) | i_1, \ldots, i_{n-1} \in [1, n]\}$ in l_4^n.

Obviously, x_n in l_4^n is substituted with n^{n-1} terms. These correspond to the instantiations defined in [9, Theorem 13].

8 Future Work

From a practical point of view, algorithms and data-structures have to be devised to apply the above rules efficiently, especially to identify conflicts and blocking substitutions in an incremental way. In the wake of this, experimental evaluations based on an implementation are reasonable. While the procedure is described for first-order logic, we believe that the same ideas could be profitably applied

[9] In a Π_2-cut, the cut formula is of the form $\forall x \exists y A$ where A is a quantifier-free formula.

to other logics, and even to other calculi, including saturation-based procedures. It would also be interesting to combine this approach with other techniques for reducing proof size, for instance variable splitting [7], or with techniques for the incremental construction of closures [5].

A current restriction of the calculus is that no abstraction variables may occur above their introduction (see the condition on the Instantiation rule in Section 4). This restriction is essential for soundness: without it, one could for instance construct a closed tableau for the (satisfiable) set of M-clauses $\{\{l\}, \{l'\}\}$, with $l = \langle L, (x, \alpha), \{[\alpha\backslash a, \alpha\backslash b]\}\rangle$ and $l' = \langle L, (\beta, y), \{[\beta\backslash a, \beta\backslash b]\}\rangle$, by replacing x by α and y by β. We think that this condition can be relaxed by defining an order over the abstraction variables. This would yield a more flexible calculus, thus further reducing proof size. It would be interesting to know whether the exponential bound of Remark 3 still holds for the relaxed calculus. An ambitious long-term goal is to devise extensions of M-tableaux with the same deductive power of cuts, i.e. enabling a non-elementary reduction of proof size.

References

1. P. Baumgartner. Hyper Tableaux - The Next Generation. In H. de Swaart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1397 of *LNAI*, pages 60–76. Springer, 1998.
2. M. de Rijke. Handbook of Tableau Methods, Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, eds. *Journal of Logic, Language and Information*, 10(4):518–523, 2001.
3. E. Eder. Relative Complexities of First-Order Logic Calculi, 1990.
4. M. Fitting. *First-Order Logic and Automated Theorem Proving.* Texts and Monographs in Computer Science. Springer Verlag, 1990.
5. M. Giese. Incremental Closure of Free Variable Tableaux. In R. Goré, A. Leitsch, and T. Nipkow, editors, *International Joint Conference on Automated Reasoning, Siena, Italy*, number 2083 in LNCS, pages 545–560. Springer-Verlag, 2001.
6. R. Hähnle. Tableaux and Related Methods. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 3, pages 100–178. Elsevier Science, 2001.
7. C. M. Hansen, R. Antonsen, M. Giese, and A. Waaler. Incremental variable splitting. *Journal of Symbolic Computation*, 47(9):1046–1065, 2012.
8. S. Hetzl, A. Leitsch, G. Reis, and D. Weller. Algorithmic Introduction of Quantified Cuts. *Theoretical Computer Science*, 549:1–16, 2014.
9. A. Leitsch and M. P. Lettmann. The problem of Π_2-cut-introduction. *Theoretical Computer Science*, 706:83–116, 2018.
10. M. P. Lettmann and N. Peltier. A Tableaux Calculus for Reducing Proof Size. *CoRR*, abs/1801.04163, 2018.
11. R. Letz, K. Mayr, and C. Goller. Controlled integration of the cut rule into connection tableau calculi. *Journal of Automated Reasoning*, 13(3):297–337, 1994.
12. R. Letz and G. Stenz. Model elimination and connection tableau procedures. In *Handbook of automated reasoning*, pages 2015–2112. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, 2001.
13. J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes).* Elsevier and MIT Press, 2001.