A Semantic Completeness Proof for TaMeD

Richard Bonichon[1] and Olivier Hermant[1]

Université Paris 6 - LIP6
8 rue du Capitaine Scott, 75015 Paris, France
email: (richard.bonichon | olivier.hermant)@lip6.fr

Abstract. Deduction modulo is a theoretical framework designed to introduce computational steps in deductive systems. This approach is well suited to automated theorem proving and a tableau method for first-order classical deduction modulo has been developed. We reformulate this method and give an (almost constructive) semantic completeness proof. This new proof allows us to extend the completeness theorem to several classes of rewrite systems used for computations in deduction modulo. We are then able to build a counter-model when a proof fails for these systems.

1 Introduction

Efficient treatment of equality and equational theories is still a challenging problem to tackle in the domain of automated theorem proving. The proof of as simple a statement as $(a+b)+((c+d)+e) = a+((b+c)+(d+e))$ might take a long time within a theory with the usual associativity and identity axioms if one uses an ineffective strategy. The resolving process should eventually be a deterministic and terminating method where we only have to check if the two terms are indeed the same modulo our theory. We would like to use computation (blind execution) instead of deduction (non-deterministic search), thus expressing the associativity axiom as a *rewrite rule on terms.*

Orienting equational theories through rewriting is not unusual, but *rewrite rules on propositions* are almost never considered. However it can be useful to allow them. One framework to handle such rewrite rules is deduction modulo [9]. The axiom $\forall x\ \forall y\ (x * y = 0 \iff (x = 0 \vee y = 0))$ yields by orientation the rewrite rule $x * y = 0 \rightarrow x = 0 \vee y = 0$, which is useful to prove $\exists z(a * a = z \Rightarrow a = z)$ by adapted automated deduction methods (see [2, 9]) which handle propositional rewriting by extended narrowing.

The use of both propositional and term rewrite rules in deduction modulo instead of unoriented axioms should result in a speed up in the proof search. However, deduction modulo has other interesting consequences: propositional rewrite rules like $P(a) \rightarrow \forall x P(x)$ can be used to restart the deductive process. The deduction modulo is powerful enough to express axiomatic theories such as arithmetic [11] or HOL [10] as sets of rewrite rules. Deduction modulo also produces shorter (regarding the size of the proof tree), more readable, proofs containing only purely deductive steps (the "important" ones to humans) and no computational details anymore.

In [2], a syntactic completeness proof of a tableau method for deduction modulo (TaMeD) is given. Our semantic completeness proof sheds.

- We first recall in Sec. 2 the sequent calculus modulo as the setting of our proof-search procedure.
- Then we define in Sec. 3 a new free-variable tableau calculus with constraints as a basis for a systematic tableau construction algorithm upon which we build a completeness proof. The need for a model construction forced us to give this proof-search algorithm. This is an improvement over [2].
- Section 4 eventually presents this new semantic completeness proof. The use of general semantic methods allows us to precisely give some categories of rewrite systems for which the completeness theorem holds. It is an improvement over both [2, 9] and [19]. The completeness proofs of the first two papers assume that certain properties such as cut elimination in the sequent calculus modulo are enjoyed by the considered rewrite systems, without describing any possible candidate. More precise conditions than just cut elimination are given in this paper. The classes of rewrite systems described in Sec. 4 also subsumes those generated by the order condition of the completeness proof of [19].
- We finally illustrate our systematic tableau procedure on an example in Sec. 5 and discuss further (mainly practical) improvements.

2 Deduction modulo for first-order classical logic

We present the sequent calculus modulo we use as a basis for the tableau method. We consider first-order classical logic without equality where formulas are built using atoms, connectors ($\wedge, \vee, \neg, \Rightarrow$) and quantifiers ($\forall, \exists$). In the rest of the paper, formulas are denoted by capital letters such as $A, B, P, Q, ...$, (multi)sets of formulas by Γ, Δ, constants by $a, b, c, ...$, function symbols by $f, g, h, ...$, terms by $t, u, v, ...$, variables by $x, y, z, ..$, free variables by $X, Y, Z, ...$; the usual substitution avoiding capture of x by t in P is denoted $P[x := t]$. A constraint is an equation $t \approx u$ where t and u are terms to unify. An immediate subformula is defined as usual : $P(t)$ is an immediate subformula of $\forall x P(x)$ for any t. Similarly, A is an immediate subformula of $A \wedge B$, and so on. We consider *confluent* rewrite systems ($\mathcal{R}$) formed by term rewrite rules and propositional rewrite rules where left members are atomic propositions. We use the single arrow for rewriting: $\longrightarrow_{\mathcal{R}}$ is a single step, $\longrightarrow^{n}_{\mathcal{R}}$ n steps, $\longrightarrow^{\star}_{\mathcal{R}}$ is the reflexive and transitive closure of $\longrightarrow_{\mathcal{R}}$. $A{\downarrow_{\mathcal{R}}}$ represents the normal form of A by a rewrite system $\mathcal{R}$. $P \equiv_{\mathcal{R}} Q$ if they have a common reduct.

A similar sequent calculus is presented in [9]. Therefore we will only exhibit a representative fragment of the rules (see Fig. 1). Extending the transformations (i.e. adding side conditions) to the whole set of rules of LK is not more difficult than what we have in Fig. 1. Notice that the side condition is not a constraint: it liberalizes the corresponding rule of LK.

$$\frac{}{P \vdash_{\mathcal{R}} Q} \text{ axiom if } P \equiv_{\mathcal{R}} Q \qquad \frac{\Gamma, P \vdash_{\mathcal{R}} \Delta \qquad \Gamma \vdash_{\mathcal{R}} Q, \Delta}{\Gamma \vdash_{\mathcal{R}} \Delta} \text{ cut if } P \equiv_{\mathcal{R}} Q$$

$$\frac{\Gamma, Q[t/x] \vdash_{\mathcal{R}} \Delta}{\Gamma, P \vdash_{\mathcal{R}} \Delta} \forall\text{-l if } P \equiv_{\mathcal{R}} \forall x\, Q \qquad \frac{\Gamma, P \vdash_{\mathcal{R}} \Delta \qquad \Gamma, Q \vdash_{\mathcal{R}} \Delta}{\Gamma, R \vdash_{\mathcal{R}} \Delta} \vee\text{-l if } R \equiv_{\mathcal{R}} P \vee Q$$

Fig. 1: Some inference rules of sequent calculus modulo

3 Tableaux modulo revisited

We introduce in this section a new version of the tableaux modulo of [2]. We use constrained a form of constrained tableaux which borrows ideas from those of [5, 6, 7, 13]. We represent tableaux as multisets of branches. A *branch* is thus a multiset of formulas written Γ, Δ.... Γ, P denotes the multiset $\Gamma \cup \{P\}$; $\mathcal{T}|\Gamma$ stands for $\mathcal{T} \cup \{\Gamma\}$. A constrained tableau is a pair $\mathcal{T} \cdot \mathcal{C}$ of a tableau $\mathcal{T}$ and a set of unification constraints $\mathcal{C}$. A branch can be *closed* when two opposite unifiable formulas can be found on it (the base case consists in having P and $\neg P$). A tableau is closed when all its branches can be (simultaneously) closed, i.e. there is a unifier closing all branches simultaneously.

Tableaux for deduction modulo We present tableaux for deduction modulo as an extension of first-order classical tableaux. We define an alternate *non-destructive* version which combines the usual tableau expansion rules (see [12, 18] for details) on formulas — called α (conjunction), β (disjunction), γ (universal) and δ (existential) — with a rule handling explicitly rewriting on terms and propositions.. Our tableau calculus works more precisely on *constrained labelled formulas*: every formula has an added label storing variables used in δ-rule for skolemization and a local constraint store which keeps track of unification constraints produced by rewriting steps. Hence formulas are written P^l_c where c is the constraint and l is the label. Note that there are two type of constraints: local constraints (attached to formulas) and global ones (attached to a tableau) which interfere only in the case of the closure rule. We also assume that variables of input formulas are renamed in order to avoid unification problems between syntactically equal but semantically independent variables.

In a γ-rule, a *globally fresh* free variable is substituted for the universally quantified variable x. This free variable is also added to the label of the formula.

In a δ-rule, the skolemization is done as follows: `sko` is a fresh Skolem function symbol whose arguments are the variables in the label of the formula. This δ-rule guarantees for example if we have the rewrite rule $x*0 \longrightarrow_{\mathcal{R}} 0$, the two equivalent (with respect to the rewrite rules) formulas $\forall x \exists y P(0, y)$ and $\forall x \exists y P(x * 0, y)$ are skolemized in the same way.

In the `rw` rule, we add a constraint to formulas resulting of rewriting steps. It keeps track of the needed unification between the originating occurrence ω of a rewritable formula in P ($P_{|\omega}$) and the left part of the applied rewrite rule (l) if we are to use the rewritten formula ($P[r]_{|\omega}$) later in our tableau. There is still no need to put the constraints globally as we are unable to guess if the formula

$$\frac{\Gamma_1, \beta(P,Q)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, \beta(P,Q)^l_c, P^l_c \mid \Gamma_1, \beta(P,Q)^l_c, Q^l_c \mid ... \mid \Gamma_n}\ \beta \qquad \frac{\Gamma_1, \alpha(P,Q)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, \alpha(P,Q)^l_c, P^l_c, Q^l_c \mid ... \mid \Gamma_n}\ \alpha$$

$$\frac{\Gamma_1, \gamma(x,P)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, P(x := X)^{l\cup\{X\}}_c, \gamma(x,P)^l_c \mid ... \mid \Gamma_n}\ \gamma \qquad \frac{\Gamma_1, \delta(x,P)^l_c \mid ... \mid \Gamma_n}{\Gamma_1, P^l_c[\mathtt{x}{:=}\mathtt{sko}(l)], \delta(x,P)^l_c \mid ... \mid \Gamma_n}\ \delta$$

$$\frac{\Gamma_1, P^l_c \mid ... \mid \Gamma_n \cdot \mathcal{C}}{\Gamma_1, P^l_c, P^l_{\mathcal{K}}[r]_\omega \mid ... \mid \Gamma_n \cdot \mathcal{C}}\ \mathtt{rw} \text{ if } l \longrightarrow_{\mathcal{R}} r, \text{ and } \mathcal{K} = (c \cup \{P_{|\omega} \approx l\}$$

$$\frac{\Gamma_1, P^{l_1}_{c_1}, \neg P^{l_2}_{c_2} \mid ... \mid \Gamma_n \cdot C}{(\Gamma_2 \mid ... \mid \Gamma_n) \cdot \mathcal{C} \cup c_1 \cup c_2 \cup \{P^{l_1} \approx P^{l_2}\}}\ \mathtt{closure}\ (\odot)$$

Fig. 2: Tableau modulo expansion and closure rules.

will be used in a closure rule. Note that no new variables are created during rewriting in deduction modulo, so there is no risk of capture.

The `closure` rule erases the closable branch provided its constraints are unifiable. It is the usual binary closure rule. Note that the labels of P and $\neg P$ need not be same (this would be the case of $P(X)$ and $\neg P(a)$), therefore we must unify some subterms of the two formulas (this is denoted by $P^{l_1} \approx P^{l_2}$) and apply the substitution to the whole tableau. The local formula constraints are transferred to the global store in `closure`. In the global constraint store, syntactically equal variables are semantically the same: what looked like possible separate branch closure might be non unifiable globally due to rigid free-variables introduced in the calculus.

There are some differences between TaMeD in [2] and these tableaux: TaMeD combines in effect a normalization procedure (yielding a kind of disjunctive normal form) of the formula which occurs *before* any application of extended narrowing and branch closure rules. Our tableau calculus here mixes formula decomposition (α, β, γ and δ rules) with closure and rewriting steps — and this allows non-atomic closure. The extended closure rule with $\mathcal{RE}$-unification of[2] constraints is now a binary closure with first-order unification constraints as $\mathcal{R}$-unification is encoded in the application of `rw`. The `rw` rule rewriting propositions do not necessitate an immediate post-decomposition of the branch it is applied on : for example, if we use $P \longrightarrow_{\mathcal{R}} Q \wedge R$ on the branch Γ, P, we keep the rewritten form $\Gamma, Q \wedge R$ instead of requiring to immediately have Γ, Q, R .

The soundness theorem holds for the following notion of model, that is a natural extension of the usual boolean model definition to deduction modulo:

Definition 1. *A boolean model is said to be a model of a rewrite system* $\mathcal{R}$ *if and only if for any propositions* $P \equiv_{\mathcal{R}} Q$, $|P| = |Q|$. $|.|$ *is then noted* $|.|_{\mathcal{R}}$.

As usual, a boolean model is mainly a total interpretation function from propositions into $\{0, 1\}$ satisfying the conditions of Def. 2. In the later, the term *model* will always mean a boolean model of the considered rewrite system $\mathcal{R}$ (clear from context).

Theorem 1 (Soundness). *If Γ has a model, then we can not derive the closed tableau $\odot$ from Γ using the rules of Fig. 2.*

Proof. The proof is standard: we check by induction on the tableau derivation that any tableau rooted at Γ, at least one branch remains true in the model. $\square$

Systematic tableau generation We now define a systematic tableau procedure which resembles the incremental closure of [13] as the first step towards our semantic completeness proof. We must ensure that our strategy is fair, which is now more complicated due to the addition of rewriting steps. We construct step by step an approximation of a complete tableau (if Γ is not provable).

First, attach a boolean (of value `true`) to the input branch. Its value will be used to alternatively use γ-rules or `rw`: ensuring completeness forces us to use them infinitely many times while remaining fair. Two orderings are used to select the formula to expand: on branches, $\mathcal{B}_1 \preceq_{\mathcal{B}} \mathcal{B}_2$ if $\text{size}(\mathcal{B}_1) \leq \text{size}(\mathcal{B}_2)$ where $\text{size}(\mathcal{B})$ returns the number of formulas on $\mathcal{B}$ and on formulas on a given branch $\mathcal{B}$, $\preceq_{\text{f}}$ is defined as $\alpha \prec_{\text{f}} \delta \prec_{\text{f}} \beta \prec_{\text{f}}$ `rw` $\prec_{\text{f}} \gamma$ if $\text{boolean}(\mathcal{B}) =$`true`, and $\alpha \prec_{\text{f}} \delta \prec_{\text{f}} \beta \prec_{\text{f}} \gamma \prec_{\text{f}}$ `rw` otherwise. Now proceed as follows:

1. Tag every formula of the input branch as *unused*. If we apply α, β, δ: the considered formula becomes *used* — to forbid the same expansion again as in a destructive method — and the branch boolean is inherited by the produced branch(es). In the case of `rw` for a given rewrite rule $r \in \mathcal{R}$ used on formula P on branch $\mathcal{B}$, P is tagged as *used(r)* to forbid applying twice the same rewrite rule to it. Name b the boolean of the expanded branch, then $b \leftarrow \neg\text{boolean}(\mathcal{B})$. In any case, produced formulas are *unused*.
2. (a) On each branch, generate from unifiable formulas on it the set of constraints which can close it. Take the intersection of these local sets to get the global set of constraints which could simultaneously close the tableau. If a unifier exist for the global store then *return 'unsatisfiable'*. Otherwise if a branch can be closed *without* unification, remove it from the tableau as in the `closure` rule (and its set of constraints is also removed from the global set).
   (b) Then select the smallest expandable branch (according to $\prec_{\mathcal{B}}$) and its set S of smallest formulas according to $\prec_{\text{f}}$. Apply the related expansion rule on every *unused* (or unused(r) if we apply `rw` with r) formulas of S. If we could not apply any expansion *return 'satisfiable'*, else **go to 2a**.

We do not rewrite formulas when we know (we check it) that we cannot unify them with the left part of the rewrite rule. More generally we do not add in the constraint stores provably non-unifiable terms.

4 Semantic completeness

In this section, we prove that the systematic tableau procedure of Sec. 3 is complete with respect to the models of Def. 1.

We will need to define a model interpretation for all propositions, even those that do not appear in the tableau. We also will need some conditions on $\mathcal{R}$, since the method cannot be complete for all confluent terminating rewrite systems, as shown in [10, 16]. Two of them are presented in Sec. 4.6.

4.1 Preliminaries

Semi-valuations have first been defined by Schütte, and correspond to Hintikka sets. The idea is that they correspond to open branches of the systematic tableau, and this is the first step toward a model. Partial valuations are a bottom-up extension of them. They differ from model interpretations by the fact that both are *partial* functions.

Definition 2 (Semi-valuation(Partial valuation)). *An interpretation is a partial function $V : \mathcal{P} \mapsto \{0,1\}$. It is called* semi- *(resp.* partial*) valuation iff:*

- *if $V(\neg P) = 0$ then (resp. iff) $V(P) = 1$*
- *if $V(\neg P) = 1$ then (resp. iff) $V(P) = 0$*
- *if $V(P \vee Q) = 0$ then (resp. iff) $V(P) = V(Q) = 0$*
- *if $V(P \vee Q) = 1$ then (resp. iff) $V(P) = 1$ or $V(Q) = 1$*
- *if $V(P \wedge Q) = 0$ then (resp. iff) $V(P) = 0$ or $V(Q) = 0$*
- *if $V(P \wedge Q) = 1$ then (resp. iff) $V(P) = V(Q) = 1$*
- *if $V(P \Rightarrow Q) = 0$ then (resp. iff) $V(P) = 1$ and $V(Q) = 0$*
- *if $V(P \Rightarrow Q) = 1$ then (resp. iff) $V(P) = 0$ or $V(Q) = 1$*
- *if $V(\forall x P) = 0$ then (resp. iff) for some ground term t, $V(P[x := t]) = 0$*
- *if $V(\forall x P) = 1$ then (resp. iff) for any ground term t, $V(P[x := t]) = 1$*
- *if $V(\exists x P) = 0$ then (resp. iff) for any ground term t, $V(P[x := t]) = 0$*
- *if $V(\exists x P) = 1$ then (resp. iff) for some ground term t, $V(P[x := t]) = 1$*

Definition 3 (Semi-valuation in deduction modulo). *A semi-valuation (resp. partial valuation) V is said to be compatible with a rewrite system $\mathcal{R}$ iff when $P \equiv_{\mathcal{R}} Q$ and $V(P)$ is defined, we have $V(Q) = V(P)$.*

This definition is a natural extension to deduction modulo of the previous one. The next four sections deal with the construction of a partial valuation in the sense of Def. 3.

4.2 Defining a semi-valuation from a complete branch of a tableau

Because of the free variables, the construction of a semi-valuation from an open complete branch is not so easy: their meaning is not determined once and for all, and an open branch might be closed by some unifier θ. A tableau is non closable when at any step n of our tableau construction no (finite) unifier θ can be found such that all branches can be closed at the same time.

We have to generate a σ, that enumerates all γ-terms. Remark that if a γ-formula P appears in a non closed branch of a complete tableau, then for infinitely many fresh variables X_n, $P[x := X_n]$ appears (the γ rule is infinitely repeated). Since in the open branch γ-formulas are countable, and *free variables are new when introduced*, we define an enumeration of $\langle \gamma_i, X^i_j \rangle$, the couples of a γ-formula (indexed by i) and its corresponding free variables (indexed by j). We fix also an enumeration of the terms of the full language (including the Skolem symbols introduced). Then, we define successive approximations of the needed substitution σ:

- σ_0 is the empty substitution.
- $\sigma_{n+1} = \sigma_n + \langle X^{i_n}_{j_n} := t_{j_n} \rangle$

Section 3 describes a process to construct $T_0, ..., T_n, ...$ that represent successive approximations to a complete tableau, that is a tableau where all possible rules have been applied. We supposed it non closable: hence for any n, there exists some open branch $\mathcal{B}$ of T_n under σ_n. Moreover, we can choose this branch such that for any further step of the systematic tableau procedure, it is never closed by any σ_p.

Given a formula γ, a term t, for some step n of the systematic tableau generation, $\gamma(t)$ appears on this branch, since it is also open under σ_n with n such that $t = t_{j_n}$ and $\gamma = \gamma_{i_n}$ (from the enumeration).

We obtain a complete open branch under σ that possesses all the needed properties, and we define our semi-valuation V:

- if a proposition P_c with free variables and constraints c appears on the branch, and if σ satisfies c, set $V(P\sigma) = 1$.
- if $\neg P_c$ appear with constraints c, and if σ satisfies c, set $V(P\sigma) = 0$.
- if the constraints are not satisfiable, drop the proposition.

It is easy to prove that all properties of Def. 2 for a semi-valuation hold: the constraints are not modified by the application of any of the α-, β-, γ-, δ-rules so if, say, an α-formula is interpreted by V, so are its two immediate subformulas. The semi-valuation hereby defined is well defined. Forcing a proposition to have two different interpretations, means that σ closes the branch: we cannot have $V(P\sigma) \neq V(P'\sigma)$ for $P\sigma = P'\sigma$.

4.3 Basic results on V

We must have that V is a semi-valuation for the rewrite system $\mathcal{R}$. This is not yet the case (think about $A \to B$, $V(B)$ defined although $V(A)$ is not defined), so it has to be extended. We first need to prove some technical properties of V.

Our starting point is: the systematic tableau generation of Sec. 3 ensures that for any atomic proposition, if $P = R\sigma$ appears, then each of its one-step reduct $Q = R'\sigma$ appear, since we exhaustively try to apply rewrite rules to any atomic proposition, and if $P \to Q$, σ unifies the constraints of R'.

Lemma 1. *Let P_0, P_n atomic formulas such that $V(P_0)$ is defined and $P_0 \to^1 P_1 \to^1 \ldots \to^1 P_n \to^1 P_{n+1}$ (P_{n+1} atomic or not), with $\to^1$ the one-step reduct relation. For any $i \leq n+1$, $V(P_i)$ is defined.*

Proof. By induction on n. We first show that the one-step reduct P_1 is interpreted under V. The process of Sec. 3 tries to one-step rewrite any literal with all possible rewrite rules. So if $P_0 = R\sigma \rightarrow^1 P_1$, with R_c appearing on the tableau, there exists by construction a $R'_{c'}$ such that $P' = R'\sigma$ and the associated constraints c' are satisfied by σ (since σ satisfies c and $P \rightarrow P'$).

Hence $V(P_1)$ is defined. If $n = 0$ we are done, else we apply the induction hypothesis on $P_1 \rightarrow^1 ... \rightarrow^1 P_n \rightarrow^1 P_{n+1}$. □

Lemma 2. *Let P, Q be two propositions such that $P \equiv_{\mathcal{R}} Q$. $V(P) = V(Q)$ if they are defined.*

Proof. By confluence there exists a proposition R such that $P \rightarrow^n R \;\; {}^m\!\leftarrow Q$. We prove Lemma 2 by induction on the pair $\langle n + m, \min(\#P, \#Q)\rangle$, where $\#P$ stands for P's number of logical connectors.

If $n = m = 0$, $Q = P = R$ and the result is trivial.

If P or Q is an atomic proposition (suppose it is P, without loss of generality), then $P \rightarrow^p P' \rightarrow^1 R' \rightarrow^q R$ with P' atomic. By Lemma 1, $V(R')$ is defined. We use induction hypothesis on R' and Q, since $p_2 < n$.

If both P and Q are compound propositions, remember that V is a semi-valuation and apply induction hypothesis.

For instance, if $P = \forall x S$, confluence implies $Q = \forall x S'$, and $R = \forall x S''$. Suppose that $V(P) = 1$ and $V(Q) = 0$. Then there exists a t such that $V(S'[x := t]) = 0$. But $V(S[x := t]) = 1$ by definition 2. We find a contradiction by applying induction hypothesis on $S[x := t] \rightarrow^n S''[x := t]^m \leftarrow S'[x := t]$ □

4.4 Extending V into a semi-valuation for $\mathcal{R}$

Lemma 2 is not sufficient: to fit with Def. 3 we need to ensure that $V(Q)$ is defined whenever $V(P)$ is. So we extend the semi-valuation V into a semi-valuation $\mathcal{V}$ for $\mathcal{R}$. Fix an enumeration P_n of the propositions of the language, such that a proposition Q is seen infinitely many times (it boils down to an enumeration of $\langle Q, n\rangle$ where Q describes the propositions and n describes $\mathbb{N}$).

Set $V_0 = V$. Set $V_{n+1} = V_n$ and extend it with P_n if $V_n(P_n)$ is not defined:

1. set $V_{n+1}(P_n) = V_n(Q)$ if for some Q, $Q \equiv_{\mathcal{R}} P_n$ and $V_n(Q)$ is defined.
2. if P_n is a compound proposition, look at the interpretation under V_n of its immediate subformulas. If we have enough information, then set $V_{n+1}(P_n)$ accordingly. For instance, if $P_n = \forall x R$ and for any t, $V_n(R[x := t]) = 1$, set $V_{n+1}(P_n) = 1$. Conversely if there is at least one t such that $V_n(R[x := t]) = 0$, set $V_{n+1}(P_n) = 0$. Even if for some t',$V_n(R[x := t'])$ is not defined.

We set $\mathcal{V}$ as the limit of the V_n. $\mathcal{V}(P) = V_n(P)$, if it is defined for some n. We do not know anything yet about $\mathcal{V}$ (a compound proposition can be defined by the first rule). That's why we need the following technical lemmas, where we adopt the convention $n - 1 = n$ if $n = 0$.

Lemma 3. *Let n an integer, P an* atomic *formula, suppose $V_n(P)$ defined. Then there exist Q such that $P \equiv_{\mathcal{R}} Q$ and either $V(Q)$ is defined, or Q is compound and $V_{n-1}(Q)$ is defined. In any case all interpretations are equal.*

Proof. By induction on n. If $n = 0$, $V_0 = V$ and we are done. If $V_{n-1}(P)$ is defined, apply induction hypothesis. Else, since P is atomic, $V_n(P)$ is equal to $V_{n-1}(P')$ for some $P' \equiv_{\mathcal{R}} P$. If P' is a compound proposition, take $Q = P'$. If it is an atomic formula, then apply the induction hypothesis on $V_{n-1}(P')$, we find a fitting $Q \equiv_{\mathcal{R}} P' \equiv_{\mathcal{R}} P$. □

Corollary 1. *Let P an atomic formula, such that $V_n(P)$ is defined, and that there exists a compound formula Q such that $P \equiv_{\mathcal{R}} Q$.*

Then there exists a compound formula R such that $P \equiv_{\mathcal{R}} R$ and $V_{n-1}(R)$ is defined and equal to $V_n(P)$.

Proof. Apply Lemma 3. If we obtain an atomic $S \equiv_{\mathcal{R}} P$ such that $V(S)$ is defined, we use confluence and apply Lemma 1 on the following reduction chain, obtained by confluence:

$$S \rightarrow^* S_q \rightarrow^1 R \rightarrow^* Q\downarrow$$

such that R is the first non-atomic proposition ($Q\downarrow$ is not atomic). From Lemma 3, Lemma 2, and since every V_{p+1} is a conservative extension of V_p, we get:

$$V_n(P) = V(S) = V(R) = V_{n-1}(R)$$

Else, S is already compound and all interpretations are equal. □

Lemma 4. *Let $P \equiv_{\mathcal{R}} Q$ be two propositions. Suppose that $\mathcal{V}(P)$ is defined. Then $\mathcal{V}(Q) = \mathcal{V}(P)$.*

Proof. Let m the least integer for which $V_m(P)$ or $V_m(Q)$ is defined, and suppose without loss of generality that $V_m(Q)$ is defined.

If $m = 0$ and $V_0(P)$ is also defined, the result comes from Lemma 2.

Else, since the enumeration has infinite repetition, P is considered at some later step n. And $V_n(P)$ is defined by the first extension rule, since the conditions for its application hold. Therefore, $\mathcal{V}(P) = V_n(P) = V_m(Q) = \mathcal{V}(Q)$.

□

Lemma 5. *$\mathcal{V}$ is a semi-valuation.*

Proof. We prove by induction that if a (compound) proposition P is defined at step n, then enough of its immediate subformulas are interpreted in $\mathcal{V}$ so as to ensure that $\mathcal{V}$ is a semi-valuation.

If $n = 0$, then this is because V is a semi-valuation.

If P is defined at step n by the second extension rule: it is immediate.

If P is defined at step n by the first rule, let Q be the proposition such that $P \equiv_{\mathcal{R}} Q$ and $V_n(P)$ is defined equal to $V_{n-1}(Q)$. Corollary 1 allows us to choose Q non-atomic. By confluence they have the same main connector. By induction hypothesis, enough immediate subformulas of Q receive an interpretation under $\mathcal{V}$. Let R_i those subformulas and R'_i their counterparts in P. We have $R_i \equiv_{\mathcal{R}} R'_i$. Applying Lemma 4, we get $\mathcal{V}(R_i) = \mathcal{V}(R'_i)$, and therefore enough subformulas of P are interpreted in $\mathcal{V}$. □

Lemma 6. *$\mathcal{V}$ is a semi-valuation for the rewrite system $\mathcal{R}$.*

Proof. This is the combination of Lemmas 4 and 5. □

4.5 Extending a semi-valuation into a partial valuation

$\mathcal{V}$ is not a partial valuation for $\mathcal{R}$. Indeed, suppose $V(\forall x\ (A(x) \wedge B(x)))$ is not defined and that for any t, $V(A(t)) = V(B(t)) = 1$. We would like to have $\mathcal{V}(\forall x\ Q) = 1$, since $\mathcal{V}(A(t) \wedge B(t)) = 1$ for any t. But we can find no n such that $V_n(A(t) \wedge B(t)) = 1$ for any t, therefore we can never have $V_{n+1}(\forall x\ (A(x) \wedge B(x))) = 1$. So we need to extend $\mathcal{V}$ as in section 4.4. But no finite step of this process is sufficient.

We define $\tilde{V}$ as the least fixpoint of this semi-valuation extension operation: this is possible, since we can define a partial order on the semi-valuations: $V \prec V'$ if V interprets less formulas than V', and if V' is conservative over V.

It is then immediate to prove the following lemma:

Lemma 7. *$\tilde{V}$ is a partial valuation for $\mathcal{R}$ and agrees with V.*

Proof. $\tilde{V}$ is a semi-valuation for $\mathcal{R}$ by construction (the extension operation respects those properties). It is also a partial valuation: if we know enough information about the subformulas of Q, then the extension operation sets the interpretation of Q. Since $\tilde{V}$ is its own extension, $\tilde{V}(Q)$ is defined. □

4.6 Transforming a partial valuation into a model for $\mathcal{R}$

We prove the following theorem:

Theorem 2 (Completeness). *Let $\mathcal{R}$ a terminating confluent rewrite system, and Γ a set of propositions. If the systematic tableau procedure of Sec. 3 rooted at Γ does not terminate, then Γ has a model, under both conditions below.*

In usual first-order logic, once one has a semi-valuation, one ends the completeness proof very easily: we extend the semi-valuation into a model by defining randomly the truth value of uninterpreted atoms by V. This is no more the case in deduction modulo, since we have to ensure that the model we construct is a model of $\mathcal{R}$. At this point, we must introduce some more conditions on $\mathcal{R}$, since model construction differ with respect to those conditions.

An order condition has been introduced by Stuber in [19] and used in [15, 16] for proving semantic completeness theorems (of the resolution ENAR and the cut-free sequent calculus). We consider a confluent rewrite system and a well-founded order $\prec$ such that:

- if $P \rightarrow Q$ then $Q \prec P$.
- if A is a subformula of B then $A \prec B$.

The domain of the model is the set of the ground terms appearing in the partial valuation $\tilde{V}$ constructed in the previous section. And we construct the interpretation by induction on the order:

- if A is a normal atom, set $|A|_{\mathcal{R}} = \tilde{V}(A)$, if defined. Else set $|A|_{\mathcal{R}}$ arbitrarily.

– if A is not a normal atom, set $|A|_{\mathcal{R}} = |A{\downarrow}\,|_{\mathcal{R}}$.
– if P is a compound proposition, set $|P|_{\mathcal{R}}$ from the interpretation of its immediate subformulas.

This definition is well-founded. We prove as in [15] the following results (in this order):

– $P \mapsto |P|_{\mathcal{R}}$ defines a model interpretation.
– $|P|_{\mathcal{R}} = |P{\downarrow}\,|_{\mathcal{R}}$.
– $P \mapsto |P|_{\mathcal{R}}$ defines a model of $\mathcal{R}$.
– $P \mapsto |P|_{\mathcal{R}}$ is a conservative extension of $\tilde{V}$.

Turning back to the proof of Theorem 2, in the model defined, $|P|_{\mathcal{R}} = 1$, for any $P \in \Gamma$. The tableau method is thus proved complete for all rewrite systems $\mathcal{R}$ that verifies this order condition.

A positivity condition. We now suppose that the rewrite system $\mathcal{R}$, besides confluence and termination, verifies a **positivity** condition: all propositional rewrite rules $l \to r \in \mathcal{R}$ are such that all atoms occurring in r occur positively: they occur under an even times of negations and as left member of an implication. r will be called a positive formula. For instance, the following rewrite rule respects the positivity condition: $P(0) \to \forall x P(x)$

The domain of the model is the ground terms. We define the interpretation rather differently:

– if A is an atom, and if $\tilde{V}(A)$ is defined, set $|A|_{\mathcal{R}} = \tilde{V}(A)$.
– if A is an atom, and if $\tilde{V}(A)$ is not defined, set $|A|_{\mathcal{R}} = 1$.
– if P is a compound proposition, set $|P|_{\mathcal{R}}$ accordingly to Def. 2.

Notice that we defined the interpretation of even non-normal atoms, disregarding rewrite rules (for now).

Lemma 8. *$P \mapsto |P|_{\mathcal{R}}$ defines a model interpretation. Let P be a proposition. If $\tilde{V}(P)$, is defined, then $|P|_{\mathcal{R}} = \tilde{V}(P)$.*

Proof. The first part of the lemma is by construction of the interpretation. The second part is proved by induction on the structure of P, using the fact that $\tilde{V}$ is a partial valuation (Def. 2). □

So if any tableau rooted at Γ cannot be closed, Γ has a boolean model. But we have not yet spoken about a needed property of the interpretation $|\ |_{\mathcal{R}}$: it has to be a model of $\mathcal{R}$.

If $P \equiv_{\mathcal{R}} Q$, and $\tilde{V}(P)$ is defined, $\tilde{V}$ being a partial valuation, and by Lemma 8:

$$|P|_{\mathcal{R}} = \tilde{V}(P) = \tilde{V}(Q) = |Q|_{\mathcal{R}}$$

The problem arises from the propositions that are *not* defined by the partial valuation $\tilde{V}$. We do not know anything, *a priori.*

We can restrain ourselves to a simpler subcase: consider only $P \to Q$ instead of its full reflexive-transitive-symmetric closure $\equiv_{\mathcal{R}}$. Also, considering P atomic is not harmful since the rewriting steps proceed always on atoms.

Now, remember that $\mathcal{R}$ is a positive rewrite system. Hence Q is a positive proposition (noted Q^+). The crucial point *was* to define the interpretation $|A|_{\mathcal{R}}$ of an non-valued atom under $\tilde{V}$ to be *the same for all atoms* (whatever: we could have set 0 or 1). Since Q^+ is positive, the intuition is that since its atoms will also be interpreted by 1, Q^+ itself will be interpreted by 1.

We however have to be very careful: first, it could have been that $\tilde{V}(Q^+)$ is defined, and set to 0. Fortunately, $\tilde{V}$ is a partial valuation, hence if $\tilde{V}(Q)$ is defined, so should be $\tilde{V}(P)$. But it still might be that some subformulas of Q are interpreted under $\tilde{V}$. We have to generalize a bit the result, in order to be able to prove it:

Lemma 9. *Let P^+ a positive and Q^- a negative ($\neg Q^-$ is positive) proposition that does not receive an interpretation under $\tilde{V}$. Then $|P^+|_{\mathcal{R}} = 1$ and $|Q^-|_{\mathcal{R}} = 0$.*

Proof. By induction over the proposition structure. The base case (P atomic) is immediate from the definition of $P \mapsto |P|_{\mathcal{R}}$. There is no base case for Q^- since an atom is positive.

We detail the case of an universally quantified proposition. If $P^+ = \forall x R^+$. Let t be a term. Since $\tilde{V}$ is a partial valuation, it can not be that $\tilde{V}(R^+[x := t]) = 0$, else $\tilde{V}(P)$ would have been defined. Hence, $|R^+[x := t]|_{\mathcal{R}} = 1$ either by Lemma 8 (if $\tilde{V}(R^+[x := t]) = 1$), or by induction hypothesis (if $\tilde{V}(R^+[x := t])$ is not defined). Since it is true for any term t, we conclude that $|P|_{\mathcal{R}} = 1$.

If $Q^- = \forall x R^-$. Since $\tilde{V}$ is a partial valuation, it can not be that $\tilde{V}(R^-[x := t]) = 1$ for any ground term t. By hypothesis we can find no t such that $\tilde{V}(R^-[x := t]) = 0$. Hence, there is a t_0 such that $\tilde{V}(R^-[x := t_0])$ is not defined, and we conclude by induction hypothesis that $|R^-[x := t_0]|_{\mathcal{R}} = 0$. Hence $|Q^-|_{\mathcal{R}} = 0$. □

We then can easily prove the lemma:

Lemma 10. *The interpretation $P \mapsto |P|_{\mathcal{R}}$ defines a model for $\mathcal{R}$.*

Proof. Let $A \to^1 P$. Either $\tilde{V}(A)$ is defined, and we conclude by Lemma 8, or it is not, and by the preceding lemma we have that $|A|_{\mathcal{R}} = |P|_{\mathcal{R}} = 1$. We then extend it to compound propositions by structural induction. At last, we extend it by induction on the number of rewrite steps to the relation $\equiv_{\mathcal{R}}$. □

The tableau method is then proved complete for the positive rewrite systems.

5 Example

We prove $2 = 2 \Rightarrow \exists x\ (x + x = 2)$ where $=$ has no special property. Insignificant formulas are omitted but we list every applied inference. Let $\mathcal{R}$ be the following fragment of Peano's arithmetic (see [11] for arithmetic as a theory modulo):

$$x + 0 \longrightarrow_{\mathcal{R}} 0 \tag{1}$$

$$x + s(y) \longrightarrow_{\mathcal{R}} s(x + y) \tag{2}$$

Notice that in the first rewrite step, rule 1 is not applied, since 0 trivially does not unify with $2 = s(s(0))$. Notice also that after that rewrite step, we could have replaced X_1 by $s(Y_1)$ yielding a faster solution. As a side comment, the two formulas under the fourth horizontal bar are produced from the formula over it using respectively $\texttt{rw}(1)$ and $\texttt{rw}(2)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{2 = 2, \forall x(x + x \neq 2)}{X + X \neq 2}\,\gamma}{s(X_1 + Y_1) \neq 2 \cdot \mathcal{C} := \{X \approx X_1, X \approx s(Y_1)\}}\,\texttt{rw}(2)}{X' + X' \neq 2}\,\gamma}{\begin{array}{c} s(X_2) \neq 2 \cdot \mathcal{C}' := \mathcal{C} \cup \{X_1 \approx X_2, \mathbf{Y_1} \approx \mathbf{0}\} \\ s(s(X_3 + Y_3)) \neq 2 \cdot \mathcal{C}'' := \mathcal{C} \cup \{X_1 \approx X_3, Y_1 \approx s(Y_3)\} \end{array}}\,\texttt{rw}(1),\texttt{rw}(2)}{\odot \cdot \{X \approx 1\}}\,\texttt{closure}(\mathcal{C}')$$

6 Conclusion and further work

We have given a new (more liberal) formulation of tableaux for deduction modulo which is better suited to automated theorem proving. For example, it does not force atomic closure anymore, which was done in [2] where branches were fully expanded before eventually applying closure. Moreover, rewriting is not a separated process anymore and can now occur after any tableau expansion (and not after all expansions have been done). We have shown its semantic completeness through a systematic generation and detailed the model construction for specific classes of rewrite systems.

Our systematic tableau construction can be seen as the first step towards an effective (but not efficient) implementation of the method. Many improvements are to be made for it to be effective.

For example, labels are an explicit way to store arguments for Skolem functions, but they may contain variables which are not present in the formula we skolemize. Moreover, if an existential quantifier is inside a universal one (as in the formula $\forall x\ \exists y\ P(x, y)$), we generate a new Skolem symbol each time we use δ after a new γ although we could simply use the same one over and over. This inefficiency could be solved either by removing the rule and preskolemizing the input formula or by using one of the improved δ-rules: δ^+([14]) (or better δ^{+^+} ([1]) or one of the other ones surveyed in [4]. Our completeness proof should not be changed by the use of one the two δ^+ rules.

Then, the δ-rule in a calculus with free-variables complicates any syntactic soundness proofs with respect to sequent calculus modulo, especially cut-free. Indeed, we can no more ensure the freshness condition. So, we are not able to translate δ-rules into deduction steps in cut-free sequent calculus modulo. The workaround should be a Skolem theorem for cut-free sequent calculus modulo, that is yet to be investigated. The link between ground tableaux and a constructive cut elimination theorem is well-known in classical logic, and studied in the intuitionistic frame in [3].

The completeness proof is not constructive at only one point: we need König's lemma to identify an infinite branch. Such a branch is useful only because we consider *boolean* models, where truth and falsity are split by construction. The workaround is well-known ([17]): we should consider models base only on truth: $\neg P$ is true means that if P is true, then every proposition is true in the model. So, reconsidering the semi-valuation definition from this point of view, we believe that this work could be shifted in a perfectly constructive framework.

Adding rewrite rules to the tableau already complicates the completeness proof even with such conditions as confluence or termination of the rewrite system. We proved this for an order condition and a positivity condition. In [3, 15] some more conditions are studied from the point of view of semantic completeness (of cut-free sequent calculus and intuitionistic tableaux) such as a mix of the two previous conditions or the formulation of HOL in first-order logic modulo given in [10]. Those results should be easily extendable to the study of tableau completeness, since we already have a partial valuation.

Concerning deduction modulo, the restriction to atomicity of the left-hand side of propositional rewrite rules can sometimes be relaxed as shown in [8] where the (first-order) sequent calculus is presented as (non-atomic) propositional rewrite rules: the system obtained is similar to a tableau calculus (no surprise here) and it exhibits the fact that the real deductive process lies within the quantifiers. Sadly, no criterion exists regarding the safe introduction of non-atomic propositional rewrite rules: this remains yet another area to be explored, which could be interesting regarding the implementation of proof-search procedures modulo.

References

[1] B. Beckert, R. Hähnle, and P. Schmitt. *The even more liberalized δ-rule in free variable Semantic Tableaux*, volume 713. June 1993.

[2] R. Bonichon. Tamed: A tableau method for deduction modulo. In *IJCAR*, pages 445–459, 2004.

[3] R. Bonichon and O. Hermant. On constructive cut admissibility in deduction modulo (submitted). `http://www-spi.lip6.fr/~bonichon/papers/occad-fl.ps`.

[4] D. Cantone and M. Nicolosi Asmundo. A sound framework for delta-rule variants in free variable semantic tableaux. In *FTP*, 2005.

[5] A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid e-unification. *Theoretical Computer Science*, 166(1-2):291–300, 1996.

[6] A. Degtyarev and A. Voronkov. What you always wanted to know about rigid e-unification. *J. Autom. Reason.*, 20(1-2):47–80, 1998.

[7] A. Degtyarev and A. Voronkov. *Equality Reasoning in Sequent-based Calculi*, chapter 10. Elsevier Science Publishers B.V., 2001.

[8] E. Deplagne. Sequent calculus viewed modulo. In Catherine Pilière, editor, *Proceedings of the ESSLLI-2000 Student Session*, pages 66–76, Birmingham, england, August 2000. University of Birmingham.

[9] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, (31):33–72, 2003.

[10] G. Dowek and B. Werner. Proof normalization modulo. *The Journal of Symbolic Logic*, 68(4):1289–1316, December 2003.

[11] G. Dowek and B. Werner. Arithmetic as a theory modulo. In J. Giesel, editor, *Term rewriting and applications*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[12] M. Fitting. *First Order Logic and Automated Theorem Proving*. Springer-Verlag, 2nd edition, 1996.

[13] M. Giese. Incremental Closure of Free Variable Tableaux. In *Proc. Intl. Joint Conf. on Automated Reasoning, Siena, Italy*, number 2083 in LNCS, pages 545–560. Springer-Verlag, 2001.

[14] R. Hähnle and P. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13(2):211–221, 1994.

[15] O. Hermant. *Méthodes Sémantiques en Déduction Modulo*. PhD thesis, Université Paris 7 - Denis Diderot, 2005.

[16] O. Hermant. Semantic cut elimination in the intuitionistic sequent calculus. *Typed Lambda-Calculi and Applications*, pages 221–233, 2005.

[17] J.-L. Krivine. Une preuve formelle et intuitionniste du théorème de complétude de la logique classique. *The Bulletin of Symbolic Logic*, 2:405–421, 1996.

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

[19] J. Stuber. A model-based completeness proof of extended narrowing and resolution. *Lecture Notes in Computer Science*, 2083:195+, 2001.