BERNHARD BECKERT AND REINER HÄHNLE

ANALYTIC TABLEAUX

1. INTRODUCTION

1.1. *Overview of this Chapter*

The aim of this chapter is twofold: first, introducing the basic concepts of analytic tableaux and, secondly, presenting state-of-the-art techniques for using non-clausal tableaux in automated deduction.

An important point involves problems arising with implementing tableau calculi, in particular with designing a deterministic proof procedure (although no concrete implementation is presented). The most important optimizations of analytic tableaux are discussed; but there are too many to give a complete list here. Instead, we present examples for the important types of optimizations, and describe the general techniques for proving soundness and completeness of different tableau variants.

In Section 2, we introduce tableaux for full first-order logic, including unifying notation, both ground and free variable versions of tableau rules, and the (non-deterministic) construction of tableau proofs. In Section 3, the semantics of tableaux is defined, which is used to prove soundness and completeness of free variable tableau in Section 4. Besides the completeness proof that uses the notion of Hintikka sets, an alternative proof is presented, based on an induction on the number of different symbols in the signature (Section 4.3). In Section 5, we discuss difficulties that emerge while resolving the inherent indeterminism of the tableau calculus and in defining a concrete, deterministic (and complete) procedure for systematic free variable tableau proof search. Finally, in Section 6 examples for optimizations of tableaux are presented, in particular those that are different from the corresponding refinements of tableaux for clause logic; it is shown how to adapt the two types of completeness proofs to certain types of optimizations.

1.2. *The General Idea of Tableaux*

A tableau, in the present chapter, is a (partial) formal proof of a logical formula. The *method* (or, as one should rather say, the family of methods) of

tableaux is a way to find such proofs in a systematic manner: a bunch of inference rules and some instructions on how to combine them, in other words, a logical calculus.

It takes more than this, of course, to distinguish tableaux from, say, natural deduction or resolution. This can be difficult, because some variants of tableaux are virtually indistinguishable from Gentzen systems, others yet can alternatively be viewed as a certain form of resolution. Nevertheless we try to give a number of characteristic properties of tableau systems which are widely agreed upon:

Proof Methodology
In most cases (and certainly in the present chapter) a tableau proof can be envisioned as a proof by (a) contradiction and (b) case distinction, i.e., each separate case must give rise to a logical contradiction.[1] The tableau framework essentially provides a sound and systematic way to generate an exhaustive set of cases for each given formula. It thus can be seen as a systematic way to derive a counter example to an assertion.

Semantic Aspects
The inference rules of tableau systems follow closely the semantics of logical connectives. Often, semantic elements are explicitly introduced as syntactic objects into a tableau system (for example, constants that refer to truth values or to worlds in modal frames). In the propositional case (and sometimes even in the first-order case), a failed tableau proof attempt can readily be turned into a counter example for the assertion to be proven.

Unrestricted Syntax
Although tableau systems for syntactically restricted input exist (see the following chapters for a thorough discussion of tableaux for clause logic), they have been developed to deal with full logic syntax including any kind of connective. In the present chapter we consider tableau systems for full first-order logic.

Analyticity and Cut-Freeness
With rare exceptions tableaux are analytic[2], i.e., only (possibly negated instances of) subformulae of the formula to be refuted occur in a tableau and no others. Usually, an even stronger property holds: each formula occurring

[1] Dually, one can interpret tableaux as systematic composition of tautologies.

[2] "Analytic" is even a constituent of the calculus' name (the term "analytic tableaux" was first used by Smullyan (1968)).

in the conclusion of a tableau inference rule is (a possibly negated instance of) a direct subformula occurring in its premiss. In particular, the cut rule is not used anywhere.

Because of the aforementioned properties of tableaux, their use is favored in deductive tasks, where natural proof representation and the ability to handle non-classical logics are important. Typical scenarios include formal software verification or modeling of intelligent agents.

1.3. *Other Sources of Information*

The most comprehensive available source of information on tableau systems is the *Handbook of Tableau Methods* (D'Agostino et al., 1998). Its introductory chapter contains a detailed historical account. Other chapters cover the main variants of clausal and non-clausal, of propositional and first-order tableau systems, as well as their implementation. The most important families of non-classical logics are given treatment in specific chapters. There is also an annotated bibliography. While the Handbook's approach is encyclopedic, it does not contain selected in-depth treatments as does the present collection whose task is to provide widely readable access to vanguard research topics.

Although Smullyan's classic text (Smullyan, 1968) is somewhat outdated it still constitutes an excellent introduction for those interested primarily in proof theory. As a more contemporary introduction to tableau methods covering also aspects of automated deduction, Fitting's book (Fitting, 1996) is highly recommended.

New results in the area of tableaux are mainly presented at the *Conference on Tableaux and Related Methods* and at the *Conference on Automated Deduction*, both held annually. There is as yet no journal devoted to tableaux. Papers are published in journals devoted to logic or deduction.

1.4. *Notation*

A *first-order signature* $\Sigma = \langle P_\Sigma, F_\Sigma \rangle$ consists of a non-empty set P_Σ of predicate symbols and a set F_Σ of function symbols. For skolemization we do not use symbols from F_Σ but from a special infinite set F_{sko} of *Skolem function symbols* that is disjoint from F_Σ; the extended signature $\langle P_\Sigma, F_\Sigma \cup F_{sko} \rangle$ is denoted by Σ^*. The symbols in P_Σ, F_Σ and F_{sko} may be used with any arity $n \geq 0$; in particular function symbols can be used as constant symbols (arity 0). In addition, there is an infinite set Var of *object variables*.

The *logical operators* are the connectives $\vee$ (disjunction), $\wedge$ (conjunction) and $\neg$ (negation), the quantifier symbols $\forall$ and $\exists$, and the constant operators *true* and *false*. Formulae that are identical up to associativity (but not commutativity) of $\vee$ and $\wedge$ are identified. Implication and equivalence are considered

to be defined operators, i.e., $\phi \rightarrow \psi$ is the same as $\neg\phi \vee \psi$, and $\phi \leftrightarrow \psi$ is the same as $(\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$.

DEFINITION 1. *The set* $\mathcal{L}_\Sigma$ *of* well-formed formulae *(wffs) over a signature* Σ *is defined by: (1) true, false and atoms over* Σ *are wffs. (2) If* ϕ *is a wff, then* $\neg\phi$ *is a wff. (3) If* $\phi_1, \ldots, \phi_n$, $n \geq 2$, *are wffs but not conjunctions (disjunctions), then* $\phi_1 \wedge \ldots \wedge \phi_n$ *(resp.* $\phi_1 \vee \ldots \vee \phi_n$*) is a conjunction (disjunction) and a wff. (4) If* ϕ *is a wff and* $x \in$ Var, *then* $\forall x(\phi)$ *and* $\exists x(\phi)$ *are wffs.*

The complement $\overline{\phi}$ *of a wff* ϕ *is defined by:* $\overline{\phi} = \psi$ *if* ϕ *is of the form* $\neg\psi$, *and* $\overline{\phi} = \neg\phi$ *otherwise.*

As we deal with arbitrary formulae, we have to account for the fact that a disjunctive subformula may occur negated and thus is implicitly a conjunctive formula etc. Also, the sign of a literal may be implicitly complemented.

DEFINITION 2. *An occurrence of a subformula* ρ *of* $\phi \in \mathcal{L}_\Sigma$ *is (1)* positive *if* $\phi = \rho$, *(2)* negative *(*positive*) if* ϕ *is of the form* $\neg\psi$ *and the occurrence of* ρ *is positive (negative) in* ψ, *(3)* positive *(*negative*) if the occurrence of* ρ *is positive (negative) in an immediate subformula* ψ *of* ϕ *such that* $\phi \neq \neg\psi$.

The notions of free and bound variable, term, atom, literal (*true* and *false* are literals but not atoms), (immediate) subformula, substitution, and sentence (a formula not containing free variables) are defined as usual. If in doubt, the reader is referred to (Fitting, 1996) for the exact definitions.

2. Ground and Free Variable Tableaux

2.1. *Unifying Notation*

Following Smullyan (1968), the set of formulae that are not literals is divided into four classes: α for formulae of conjunctive type, β for formulae of disjunctive type, γ for quantified formulae of universal type and finally δ for quantified formulae of existential type (unifying notation). This classification is motivated by the *tableau expansion rules* that are associated with each (non-literal) formula. The rules characterize the assignment of a truth value to a formula by means of assigning truth values to its direct subformulae. For example, $\phi \wedge \psi$ holds if and only if ϕ and ψ hold.

Tableau systems come in two versions, namely unsigned and signed; for first-order logic the signs T (true) and F (false) are used. Although signed tableaux are more flexible and for most (non-classical) logics it is necessary to use signs, we will (mainly) be using the unsigned version; all techniques presented apply as well to the signed version or can easily be adapted.

Table I. Correspondence between formulae and rule types (unsigned version).

α	$\alpha_1, \ldots, \alpha_n$
$\phi_1 \wedge \ldots \wedge \phi_n$	$\phi_1, \ldots, \phi_n$
$\neg(\phi_1 \vee \ldots \vee \phi_n)$	$\neg\phi_1, \ldots, \neg\phi_n$
$\neg\neg\phi$	ϕ

β	$\beta_1, \ldots, \beta_n$
$\phi_1 \vee \ldots \vee \phi_n$	$\phi_1, \ldots, \phi_n$
$\neg(\phi_1 \wedge \ldots \wedge \phi_n)$	$\neg\phi_1, \ldots, \neg\phi_n$

γ	γ_1
$\forall x(\phi(x))$	$\phi(x)$
$\neg\exists x(\phi(x))$	$\neg\phi(x)$

δ	δ_1
$\neg\forall x(\phi(x))$	$\neg\phi(x)$
$\exists x(\phi(x))$	$\phi(x)$

Table II. Correspondence between formulae and rule types (signed version).

α	$\alpha_1, \ldots, \alpha_n$
$\mathsf{T}(\phi_1 \wedge \ldots \wedge \phi_n)$	$\mathsf{T}\phi_1, \ldots, \mathsf{T}\phi_n$
$\mathsf{F}(\phi_1 \vee \ldots \vee \phi_n)$	$\mathsf{F}\phi_1, \ldots, \mathsf{F}\phi_n$
$\mathsf{T}\neg\phi$	$\mathsf{F}\phi$
$\mathsf{F}\neg\phi$	$\mathsf{T}\phi$

β	$\beta_1, \ldots, \beta_n$
$\mathsf{T}(\phi_1 \vee \ldots \vee \phi_n)$	$\mathsf{T}\phi_1, \ldots, \mathsf{T}\phi_n$
$\mathsf{F}(\phi_1 \wedge \ldots \wedge \phi_n)$	$\mathsf{F}\phi_1, \ldots, \mathsf{F}\phi_n$

γ	γ_1
$\mathsf{T}\forall x(\phi(x))$	$\mathsf{T}\phi(x)$
$\mathsf{F}\exists x(\phi(x))$	$\mathsf{F}\phi(x)$

δ	δ_1
$\mathsf{F}\forall x(\phi(x))$	$\mathsf{F}\phi(x)$
$\mathsf{T}\exists x(\phi(x))$	$\mathsf{T}\phi(x)$

DEFINITION 3. *The non-literal formulae in $\mathcal{L}_{\Sigma^*}$ are assigned a* type *according to Table I (resp. Table II for the signed version of tableaux). A formula of type $\xi \in \{\alpha, \beta, \gamma, \delta\}$ is called a* ξ-formula.

The letters α, β, γ, and δ are used to denote formulae of (and only of) the appropriate type. The variable x that is bound by the (top-most) quantifier in γ- and δ-formulae is made explicit by writing $\gamma(x)$ (resp. $\delta(x)$); accordingly, $\gamma_1(t)$ denotes the result of replacing all occurrences of x in γ_1 by t.

2.2. *Ground Tableau Expansion Rules*

We start with the *ground* version of tableaux for first-order logic, called so, because universally quantified variables are replaced by *ground* terms when the γ-rule is applied.

Table III. Rule schemata for the ground version of tableaux.

$\dfrac{\alpha}{\alpha_1 \\ \vdots \\ \alpha_n}$	$\dfrac{\beta}{\beta_1 \mid \cdots \mid \beta_n}$	$\dfrac{\gamma(x)}{\gamma_1(t)}$ where t is any ground term.	$\dfrac{\delta(x)}{\delta_1(c)}$ where $c = sko(\delta)$.

In Table III the ground expansion rule schemata for the various formula types are given schematically. Premisses and conclusions are separated by a horizontal bar, while vertical bars in the conclusion denote different *extensions*. The formulae in an extension are implicitly conjunctively connected, and different extensions are implicitly disjunctively connected. We use n-ary α- and β-rules, i.e., when the β-rule is applied to a formula $\psi = \phi_1 \vee \ldots \vee \phi_n$, then ψ is broken up into n subformulae (instead of splitting it into two formulae $\phi_1 \vee \ldots \vee \phi_r$ and $\phi_{r+1} \vee \ldots \vee \phi_n$, $1 \leq r < n$).

Besides using n-ary rules we deviate from the classic definition of tableaux (as given by Smullyan) by using an improved δ-rule that, for the purpose of constructing the Skolem term, does not introduce a *new* Skolem function symbol. Rather, each equivalence class of δ-formulae identical up to variable renaming is assigned its own unique Skolem symbol (which can be seen as a Gödelization of that class) (Beckert et al., 1993). This δ-rule is easier to implement than the classical one; and it guarantees that only a finite number of different symbols is required, thus restricting the search space.

DEFINITION 4. *Given a signature $\Sigma = \langle P_\Sigma, F_\Sigma \rangle$, the function sko assigns to each $\delta \in \mathcal{L}_{\Sigma^*}$ a symbol $sko(\delta) \in F_{sko}$ such that (a) $sko(\delta) > f$ for all $f \in F_{sko}$ occurring in δ, where $>$ is an arbitrary but fixed ordering on F_{sko}, and (b) for all $\delta, \delta' \in \mathcal{L}_\Sigma$ the symbols $sko(\delta)$ and $sko(\delta')$ are identical if and only if δ and δ' are identical up to variable renaming (including renaming of the bound variables).*

The purpose of condition (a) in the above definition of *sko* is to avoid cycles like: $sko(\delta) = f$, f occurs in δ', $sko(\delta') = g$, g occurs in δ.

2.3. *Free Variable Tableau Expansion Rules*

Using free variable[3] quantifier rules (Prawitz, 1960; Wang, 1960; Brown, 1978; Broda, 1980; Reeves, 1987; Fitting, 1996) is crucial for efficient implementation. They reduce at each step the number of possible next steps

[3] In the literature, several other names have been used for the same concept, e.g., parameters, dummy variables, meta variables.

Table IV. γ- and δ-rule schemata for free variable tableaux.

$\dfrac{\gamma(x)}{\gamma_1(y)}$	$\dfrac{\delta(x)}{\delta_1(f(x_1,\ldots,x_n))}$
where $y \in$ Var is new to the tableau.	where $f = sko(\delta)$ and $x_1,\ldots,x_n$ are the free variables in δ.

in the construction of a tableau proof and thus the size of the search space. When γ-rules are applied, a new free variable is substituted for the quantified variable, instead of replacing it by a ground term, that has to be "guessed." Free variables can later be instantiated "on demand," when a tableau branch is closed.

To preserve correctness, the schema for δ-rules has to be changed as well: the Skolem terms introduced now contain the free variables occurring in the formula to which a δ-rule is applied;[4] the free variable rule schemata for γ- and δ-formulae are shown in Table IV; the rules for propositional formulae are identical to those of the ground version of tableaux (see Table III).

2.4. *Tableau Subformulae*

The formulae being derived from a formula φ and added to a tableau by applying tableau expansion rules are called *tableau subformulae* of φ. They are closely related to, but not identical to the subformulae of φ.

DEFINITION 5. *To each non-literal formula* φ *in* $\mathcal{L}_{\Sigma^*}$*, a sequence of* immediate tableau subformulae *is assigned, which are the formulae in the conclusion when the appropriate tableau rule is applied to* φ *(see Tables III and IV). The* tableau subformula *relation is the reflexive, transitive closure of the immediate tableau subformula relation.*

A ξ*-formula* ψ *that is a tableau subformula of a formula* φ *is called a* ξ-subformula *of* φ.

The index i ($1 \leq i \leq n$) of the immediate tableau subformulae is an operator; thus, β_2 is by definition the second β-subformula of $\beta \in \mathcal{L}_{\Sigma^*}$. Note that the

[4] In earlier versions of free variable tableaux, all free variables occurring on the tableau *branch* were made part of the Skolem term, which can lead to longer proofs. The δ-rule we present here has been shown to be correct in (Beckert et al., 1993); a similar δ-rule has already been used in (Brown, 1978). δ-rules that allow even shorter proofs have been investigated in (Baaz and Fermüller, 1995). See Section 4.4 of Chapter I.1.4 for a detailed discussion of different δ-rules and their relation to the analytic cut rule.

definition of tableau subformulae depends on the tableau rules that are used; in particular, the tableau subformulae of γ- and δ-formulae differ in ground and free variable tableaux.

EXAMPLE 1. *Let* β *be the* β*-formula* $\neg(p \wedge \neg\exists x(Q(x)))$*; then* $\beta_1 = \neg p$ *and* $\beta_2 = \neg\neg\exists x(Q(x))$. β_2 *is an* α*-subformula of* β *(because it is an* α*-formula), and* $\exists x(Q(x))$ *is a* δ*-subformula of* β*. The tableau subformulae of* β *are:* β *itself,* $\neg p$*,* $\neg\neg\exists x(Q(x))$*,* $\exists x(Q(x))$*, and* $Q(c)$ *where* $c = sko\exists x(Q(x))$ *(they are the same for both ground and free variable tableau rules). The subformulae of* β *are:* β *itself,* $p \wedge \neg\exists x(Q(x))$*,* p*,* $\neg\exists x(Q(x))$*,* $\exists x(Q(x))$ *and* $Q(x)$.

2.5. *Tableau Proofs*

We consider the formulae in the given set Φ, whose unsatisfiability is to be proven, to be implicit elements of all tableau branches. Thus, the construction of a tableau proof starts with the initial tableau consisting of the single node *true*. Neither is a tableau rule for explicitly adding formulae from Φ to a tableau branch needed, nor is it necessary to make the formulae in Φ elements of the initial tableau (which would restrict Φ to be finite).

For pedagogical reasons, we prefer to view tableaux as trees in our presentation, but we regard this choice as inessential: implementations naturally avoid copying formulae and thus are closer to the path-based view preferred by other authors. On the other hand, we believe there is a crucial difference between normal form and non-normal form calculi: it is not always obvious how to transform a proof of the normalized version of a problem to a non-normal form proof. One should not confuse this issue with truly presentational aspects such as whether one prefers trees over matrices or vice versa.

DEFINITION 6. *Let* Σ *be a first-order signature. A* tableau *(over* Σ*) is a finitely branching tree whose nodes are formulae from* $\mathcal{L}_{\Sigma^*}$*. A* branch *in a tableau* T *is a maximal path in* T*.*[5] *Given a set* Φ *of sentences from* $\mathcal{L}_\Sigma$*, the tableaux for* Φ *are (recursively) defined by:*

1. *The tree consisting of a single node labeled with true is a tableau for* Φ *(initialization).*
2. *Let* T *be a tableau for* Φ*,* B *a branch of* T*, and* ψ *a formula in* $B \cup \Phi$*. If the tree* T' *is constructed by extending* B *by as many new linear subtrees as the tableau expansion rule corresponding to* ψ *has extensions, where the nodes of the new subtrees are labeled with the formulae in the extensions, then* T' *is a tableau for* Φ *(expansion).*

[5] Where no confusion can arise, branches are often identified with the set of formulae they contain.

3. *Let T be a tableau for Φ, B a branch of T, and ψ and ψ' literals in $B \cup \Phi$. If ψ and $\overline{\psi'}$ are unifiable with a most general unifier (MGU) σ, and T' is constructed by applying σ to all formulae in T (i.e., $T' = T\sigma$), then T' is a tableau for Φ (closure).*[6]

The tableau expansion rule corresponding to a formula ϕ is obtained by looking up the formula type of ϕ in Table I (resp. Table II) and instantiating the matching rule schema in Table III or Table IV; the quantifier rules in Table III are used for the ground version and the quantifier rules in Table IV for the free-variable version of tableaux.

The closure rule in Def. 6 only allows the application of *most general* closing substitutions (MGU closure rule) and only uses complementary pairs of *literals*. Instead one could allow the application of arbitrary substitutions and use complementary non-literal formulae for closure; these alternative closure rules, however, increase the number of choice points in the construction of a tableau proof (see Section 5).

DEFINITION 7. *Given a tableau T for a set Φ of sentences, a branch B of T is* closed *iff $B \cup \Phi$ contains a pair $\phi, \neg\phi \in \mathcal{L}_{\Sigma^*}$ of complementary formulae, or false or $\neg$true; otherwise it is* open. *A tableau is closed if* all *its branches are closed.*

DEFINITION 8. *A* tableau proof *for (the unsatisfiability of) a set $\Phi \subset \mathcal{L}_\Sigma$ of sentences consists of a tableau T for Φ that is closed.*

EXAMPLE 2. *We give a tableau proof for the set-theoretic theorem that, given sets P, Q, R, S such that the propositions (1) $S \cap Q = \emptyset$, (2) $P \subset Q \cup R$, (3) $P \neq \emptyset$ or $Q \neq \emptyset$, and (4) $Q \cup R \subset S$ hold, we have (5) $P \cap R \neq \emptyset$. To formalize this theorem, we use the signature $\Sigma = \langle \{P, Q, R, S\}, \{\} \rangle$; the theorem holds iff the set Φ is unsatisfiable that consists of the following five $\mathcal{L}_\Sigma$-sentences: (1) $\neg\exists x(S(x) \wedge Q(x))$, (2) $\forall x(\neg P(x) \vee Q(x) \vee R(x))$, (3) $\exists x(P(x)) \vee \exists x(Q(x))$, (4) $\forall x(\neg(Q(x) \vee R(x)) \vee S(x))$, and (5) $\neg\exists x(P(x) \wedge R(x))$.*

Figure 1 shows a tableau T for Φ. The nodes of the tableau are numbered starting from 6 (the numbers 1–5 refer to the formulae in Φ); a pair $[i; j]$ is attached to the i-th node N_i, the number j denotes that N_i has been created by applying an expansion rule to the formula in N_j (resp. formula j in Φ).

When the δ-rule is applied to nodes 7 resp. 8 to create nodes 9 and 25, the Skolem symbols $c = sko(\exists x(P(x)))$ and $d = sko(\exists x(Q(x)))$ are introduced.

All branches of T except the one with leaf 25 can be closed by applying the closure rule; the triple $[i; k; \sigma]$ below each of these branches denotes that

[6] The complement of signed formulae is defined by $\overline{\mathsf{T}\phi} = \mathsf{F}\phi$ and $\overline{\mathsf{F}\phi} = \mathsf{T}\phi$.

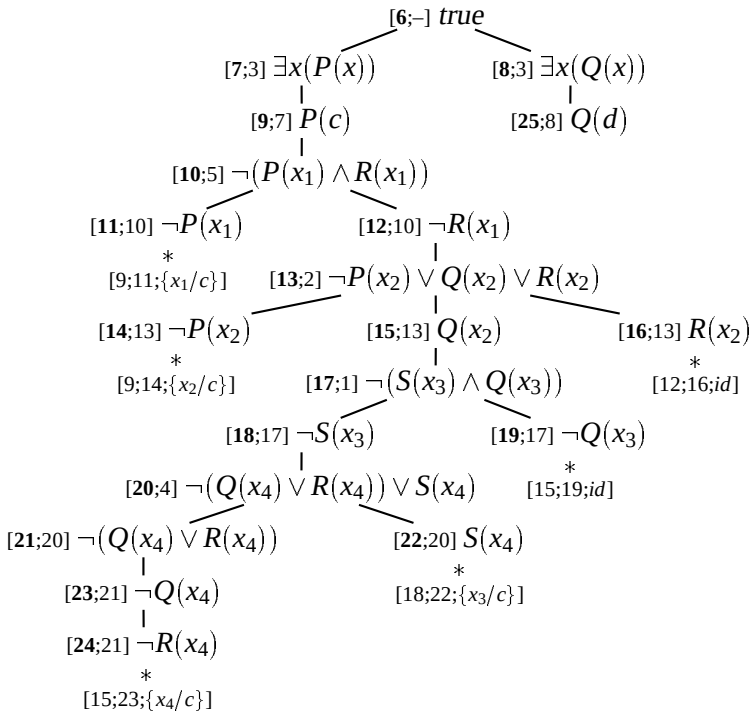


Figure 1. Partial free variable tableau proof for the set Φ from Example 2.

the closure rule can be applied to the complementary literals i and j using the unifier σ (assuming that the branches are closed from left to right). When these substitutions have been applied, the resulting taleau T' is not a tableau proof yet, but it can be extended to a closed tableau by adding a copy of the subtableau with root node 17 below node 25 and instantiating the free variables in that copy with d (instead of c).

2.6. *A Tableau Construction Procedure*

The—non-deterministic—procedure shown in Table V constructs a free variable tableau proof for the unsatisfiability of a given set Φ of sentences.

The main loop of this procedure contains the following four choice points: (1) A branch *B* has to be chosen (*select branch*); (2) it has to be decided whether *B* is to be closed or to be expanded (*select mode*); (3) if *B* is to be closed, a pair of complementary literals and thus a closing substitution has to be chosen (*select pair*); (4) if *B* is to be expanded, a formula has to be chosen to which an expansion rule is applied (*select formula*).

Table V. A tableau construction procedure.

input Φ;
$T :=$ tableau whose single node is *true*;
while T is not closed **do**
    select a branch B of T that is not closed;
    $Compl := \{\langle \phi, \phi' \rangle \mid \phi, \phi'$ literals in $B \cup \Phi$, $\phi, \overline{\phi'}$ unifiable$\}$;
    if $Compl \neq \emptyset$ **then** select a mode $M \in \{close, expand\}$
    else $M := expand$
    fi;
    if $M = close$ **then**
        select $\langle \phi, \phi' \rangle \in Compl$;
        $\sigma :=$ most general unifier of ϕ, ϕ';
        $T := T\sigma$
    else
        select non-literal formula $\phi \in B \cup \Phi$;
        $T :=$ result of applying the appropriate rule to ϕ on B
    fi
od;
output T

In the ground version of tableaux, the third choice point does not exist, because complementary literals do not contain variables and therefore all have the same most general unifier (the empty substitution). Instead there is an additional choice point when the γ-rule is applied: the ground term has to be chosen that replaces the quantified variable.

3. Semantics of Tableaux

In this section we first introduce the (standard) semantics for first-order logic, and then extend these semantics to free variable tableaux.

DEFINITION 9. *A* structure $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ *for a signature* Σ *consists of a domain* $\mathbf{D}$ *and an interpretation* $\mathbf{I}$*, which gives meaning to the function and predicate symbols of* Σ*. A structure* $\mathbf{M}$ *is a* term structure *if* $\mathbf{D}$ *is the set of all ground terms over* Σ*.*

A variable assignment *is a mapping* $\mu : \mathrm{Var} \to \mathbf{D}$ *from the set of variables to the domain* $\mathbf{D}$*. Specifically,* $\mu[x \leftarrow d]$ *denotes the variable assignment that is defined by:* $\mu[x \leftarrow d](x) = d$ *and* $\mu[x \leftarrow d](y) = \mu(y)$ *for all other variables* y*.*

An interpretation $\mathbf{I}$ *and an assignment* μ *associate (by structural recursion) with each term* t *over* Σ *an element* $t^{\mathbf{I},\mu}$ *in* $\mathbf{D}$*.*

The evaluation function $val_{\mathbf{I},\mu}$ *is, for all well-formed formulae* ϕ *in* $\mathcal{L}_\Sigma$*, defined by:* $val_{\mathbf{I},\mu}(\phi) = true$ *in case (1)* $\phi = P(t_1,\ldots,t_n)$ *and* $\langle t_1^{\mathbf{I},\mu},\ldots,t_n^{\mathbf{I},\mu}\rangle \in P^{\mathbf{I}}$*; (2)* $\phi = \neg P(t_1,\ldots,t_n)$ *and* $\langle t_1^{\mathbf{I},\mu},\ldots,t_n^{\mathbf{I},\mu}\rangle \notin P^{\mathbf{I}}$*; (3)* $\phi = true$*; (4)* $\phi = \neg false$*; (5)* $\phi = \alpha$ *and* $val_{\mathbf{I},\mu}(\alpha_i) = true$ *for all* α_i*; (6)* $\phi = \beta$ *and* $val_{\mathbf{I},\mu}(\beta_i) = true$ *for some* β_i*; (7)* $\phi = \gamma$ *and* $val_{\mathbf{I},\mu[x\leftarrow d]}(\gamma_1) = true$ *for all* $d \in D$*; (8)* $\phi = \delta$ *and* $val_{\mathbf{I},\mu[x\leftarrow d]}(\delta_1) = true$ *for some* $d \in D$*; and* $val_{\mathbf{I},\mu}(\phi) = false$ *otherwise.*

If $val_{\mathbf{I},\mu}(\phi) = true$*, which is denoted by* $(\mathbf{M},\mu) \models \phi$*, holds for all assignments* μ*, then* $\mathbf{M}$ *is a* model *of* ϕ*.*

In the sequel we only consider term structures, which is justified by the following well known theorem:

THEOREM 1. *A set* Φ *of formulae has a model if and only if it has a term model, i.e., a model that is a term structure.*

DEFINITION 10. *A tableau* T *for* $\Phi \subset \mathcal{L}_\Sigma$ *is* satisfiable *if there is a (term) model* $\mathbf{M}$ *of* Φ *such that for every variable assignment* μ *there is a branch* B *of* T *with* $(\mathbf{M},\mu) \models B$*. In that case we say that* $\mathbf{M}$ *is a model of* T*, denoted by* $\mathbf{M} \models T$*.*

Note that in the above definition there has to be a *single* model $\mathbf{M}$ satisfying a branch of T for *all* variable assignments; $\mathbf{M}$ has to be a model of Φ, because the formulae in Φ are implicit elements of all branches of T.

4. Soundness and Completeness

We prove soundness and completeness of free variable tableaux. The proofs can easily be adapted for ground tableaux. In addition to the usual completeness proof based on Hintikka sets, an alternative proof for propositional tableaux is presented.

4.1. *Soundness of Free Variable Tableaux*

The main part of the proof is to show that all tableau expansion and closure rules preserve satisfiability—which implies that, if the set Φ and thus the initial tableau is satisfiable, then all tableaux for Φ are satisfiable and, thus, cannot be closed. Then, the existence of a closed tableau for Φ implies the unsatisfiability of Φ.

For the soundness proof we restrict all considerations to *canonical* structures, where Skolem symbols are interpreted "in the right way," such that the δ-rule preserves satisfiability (in canonical models).

DEFINITION 11. *A term structure* $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ *is* canonical *iff for all variable assignments* μ *and all* $\delta(x) \in \mathcal{L}_{\Sigma^*}$ *the following holds: If* $(\mathbf{M}, \mu) \models \delta(x)$ *then* $(\mathbf{M}, \mu) \models \delta_1(f(x_1, \ldots, x_n))$*, where* $f = sko(\delta)$ *and* $x_1, \ldots, x_n$ *are the free variables in* δ*.*

Restriction to canonical structures makes sense, because every structure $\mathbf{M}$ for a signature Σ that satisfies a set $\Phi \subset \mathcal{L}_\Sigma$ of sentences can be extended to a canonical structure for Σ^*, which still satisfies Φ.

LEMMA 1. *Given a signature* Σ*, if the set* $\Phi \subset \mathcal{L}_\Sigma$ *of sentences is satisfiable, then there is a canonical structure* $\mathbf{M}^*$ *over* Σ^* *such that* $\mathbf{M}^* \models \Phi$*.*

Proof. Since Φ is satisfiable, there is a structure $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ over Σ with $\mathbf{M} \models \Phi$. The Skolem function symbols in F_{sko} do not occur in Φ, therefore it suffices to choose their interpretation such that the resulting structure $\mathbf{M}^*$ is canonical, leaving the interpretation of the symbols in Σ unchanged.

The rank $rk(f)$ of the symbols $f \in F_{sko}$ is defined as follows: (1) if no $\delta \in \mathcal{L}_{\Sigma^*}$ exists such that $sko(\delta) = f$, then $rk(f) = 0$; (2) if $f = sko(\delta)$ for some $\delta \in \mathcal{L}_\Sigma$, then $rk(f) = 1$; (3) if $f = sko(\delta)$ for some δ not in $\mathcal{L}_\Sigma$, then $rk(f) = 1 + \max\{rk(f') \mid f' \in F_{sko} \text{ occurs in } \delta\}$. Because of condition (a) in Def. 4, $rk(f)$ is well defined for all function symbols f.

We inductively define a sequence $(\mathbf{M}^n)_{n \geq 0}$ of structures that all have the domain $\mathbf{D}$. $\mathbf{M}^n = \langle \mathbf{D}, \mathbf{I}^n \rangle$ is a structure over the signature Σ^n that is the restriction of Σ^* to function symbols of rank not greater than n; $\mathbf{I}^{n+1}$ coincides with $\mathbf{I}^n$ on all symbols in $\Sigma^n \cup \Sigma$. I^0 is defined by $f^{\mathbf{I}^0} = f^{\mathbf{I}}$ for all $f \in F_\Sigma$, and for all $f \in F_{sko}$ of rank 0 the value of $f^{\mathbf{I}^0}$ is chosen arbitrarily. The function symbols $f \in F_{sko}$ of rank $r \leq n$ have already been interpreted in $\mathbf{M}^n$. Consider $f \in F_{sko}$ of rank $n+1$ with $f = sko(\delta(x))$; for all argument tuples $b_1, \ldots, b_k \in \mathbf{D}$ of the same length as the tuple $x_1, \ldots, x_k$ of free variables in $\delta(x)$ we define $f^{\mathbf{I}^{n+1}}$ by: if there is a variable assignment μ with $\mu(x_i) = b_i$ (for $1 \leq i \leq n$), and $(\mathbf{M}^n, \mu) \models \delta$, choose an element $c \in \mathbf{D}$ with $(\mathbf{M}^n, \mu[x \leftarrow c]) \models \delta_1(x)$, and set $f^{\mathbf{I}^{n+1}}(b_1, \ldots, b_k) = c$. Since f is of rank $n+1$, the symbols in δ are from the signature Σ^n. Otherwise, if $(\mathbf{M}^n, \mu) \not\models \delta(x)$, choose $f^{\mathbf{I}^{n+1}}(b_1, \ldots, b_k)$ to be an arbitrary element in $\mathbf{D}$.

We can think of the sequence $(\mathbf{M}^n)_{n \geq 0}$ as an approximation to the structure $\mathbf{M}^* = \langle \mathbf{D}, \mathbf{I}^* \rangle$ over Σ^*. $\mathbf{I}^*$ coincides with $\mathbf{I}^n, \mathbf{I}^{n+1}, \ldots$ on the symbols in Σ^n. $\mathbf{M}^*$ is canonical by construction and satisfies the formulae in Φ.

The δ-rule is a special case of the more general concept of *skolemization*, which, according to the following lemma, preserves satisfiability by canonical models.

LEMMA 2. *Let $\mathbf{M}^*$ be a canonical model over Σ^*, μ a variable assignment, and $\phi \in \mathcal{L}_{\Sigma^*}$; and let ϕ' be constructed from ϕ by replacing (a) a positive occurrence of some $\delta(x)$ in ϕ by $\delta_1(f(x_1,\ldots,x_n))$ or (b) a negative occurrence of $\overline{\delta(x)}$ in ϕ by $\overline{\delta_1(f(x_1,\ldots,x_n)}$; where $f = sko(\delta(x))$ and $x_1,\ldots,x_n$ are the free variables in $\delta(x)$. Then $(\mathbf{M}^*,\mu) \models \phi$ implies $(\mathbf{M}^*,\mu) \models \phi'$.*

Proof. If $(\mathbf{M}^*,\mu) \models \overline{\delta(x)}$ then $(\mathbf{M}^*,\mu) \models \overline{\delta_1(f(x_1,\ldots,x_n))}$, as $\mathbf{M}^*$ is canonical; and if $(\mathbf{M}^*,\mu) \not\models \overline{\delta(x)}$ then $(\mathbf{M}^*,\mu) \not\models \overline{\delta_1(f(x_1,\ldots,x_n))}$ (by double negation). Using these relations, the lemma is easily proven by induction on the structure of ϕ.

EXAMPLE 3. *In the formula $\phi = \neg(\neg\exists x(P(x,y)) \vee \forall x(Q(x)))$, the δ-formula $\delta(x) = \exists x(P(x,y))$ occurs positively and this occurrence can be replaced by $\delta_1(f(y)) = P(f(y),y)$, where $f = sko(\exists x(P(x,y)))$. The complement of the δ-formula $\delta'(x) = \neg\forall x(Q(x))$ is $\overline{\delta'(x)} = \forall x(Q(x))$; it occurs negatively in ϕ and that occurrence can be replaced by $\overline{\delta'_1(c)} = Q(c)$, where $c = sko(\neg\forall x(Q(x)))$. Thus, $(\mathbf{M}^*,\mu) \models \phi$ implies $(\mathbf{M}^*,\mu) \models \neg(\neg(P(f(y),y)) \vee Q(c))$.*

Next we prove that satisfiability by canonical models is preserved by the tableau expansion and closure rules and, therefore, all tableaux for a satisfiable set $\Phi \subset \mathcal{L}_\Sigma$ are satisfiable (by a canonical model).

LEMMA 3. *If T is a tableau for a satisfiable set $\Phi \subset \mathcal{L}_\Sigma$ of sentences, then T is satisfiable.*

Proof. By definition of tableaux for Φ there has to be a sequence $T_1,\ldots,T_m$ ($m \geq 1$), where $T = T_m$ and T_1 is the initial tableau whose single node is *true*, and where T_{i+1} is constructed from T_i by applying a single tableau expansion or closure rule. Since Φ is satisfiable, there is a canonical structure $\mathbf{M}^*$ over Σ^* such that $\mathbf{M}^* \models \Phi$ (Lemma 1). By induction on m we prove that $\mathbf{M}^*$ satisfies all the tableaux $T_1,\ldots,T_m$ (and in particular T).

$m = 1$: *true* is the only label of T_1, so trivially $\mathbf{M}^* \models T_1$.

$m \to m+1$, expansion rule: let B_m be a branch in T_m. T_{m+1} is obtained from T_m by applying a tableau expansion rule to a formula $\phi \in B_m \cup \Phi$.

Let μ be a fixed assignment. By assumption $\mathbf{M}^*$ satisfies T_m; thus we have $(\mathbf{M}^*,\mu) \models B_m^0$ for some branch B_m^0 of T_m. If B_m^0 is different from B_m, then B_m^0 is also a branch of T_{m+1} and we are through.

If, on the other hand, $B_m^0 = B_m$ and therefore $(\mathbf{M}^*,\mu) \models \phi$, we show that $(\mathbf{M}^*,\mu)$ satisfies one of the branches of T_{m+1} by cases according to which tableau rule is applied to obtain T_{m+1} from T_m.

β-rule ($\phi = \beta$): T_{m+1} is constructed from T_m by adding β_i to B_m obtaining B_{m+1}^i ($1 < i \leq k$, where k is the number of immediate tableau subformulae

of β). Since $(\mathbf{M}^*, \mu) \models \beta$ we have, by the property of β-formulae (Def. 9), $(\mathbf{M}^*, \mu) \models \beta_i$ for some $i \in \{1, \ldots, k\}$. Therefore $(\mathbf{M}^*, \mu) \models B^i_{m+1}$.

α-rule ($\phi = \alpha$): similar to the β-rule.

γ-rule ($\phi = \gamma(x)$): T_{m+1} is constructed from T_m by adding $\gamma_1(y)$ to B_m obtaining the branch B_{m+1}. Since $(\mathbf{M}^*, \mu) \models \gamma(x)$ we have, by definition of $\models$, that $(\mathbf{M}^*, \mu[x \leftarrow d]) \models \gamma_1(x)$ for all elements $d \in \mathbf{D}$. Since this is in particular true for $d = \mu(y)$, we get $(\mathbf{M}^*, \mu) \models \gamma_1(y)$ and therefore $(\mathbf{M}^*, \mu) \models B_{m+1}$.

δ-rule ($\phi = \delta(x)$): the tableau T_{m+1} is constructed from T_m by adding $\delta_1(f(x_1, \ldots, x_n))$ to B_m obtaining the branch B_{m+1}, where $f = sko(\delta(x))$ and $x_1, \ldots, x_n$ are the free variables in $\delta(x)$. Since $(\mathbf{M}^*, \mu) \models \delta(x)$ and $\mathbf{M}^*$ is canonical, $(\mathbf{M}^*, \mu) \models \delta_1(f(x_1, \ldots, x_n))$ (Lemma 2), and thus $(\mathbf{M}^*, \mu) \models B_{m+1}$.

$m \to m+1$, closure rule: it suffices to show that the application of any substitution σ to the satisfiable tableau T_m preserves satisfiability. To prove this claim we consider an arbitrary variable assignment ν and define the variable assignment μ by $\mu(x) = (x\sigma)^{\mathbf{I},\nu}$ for all $x \in$ Var. That implies for all terms over Σ^*, and in particular for all terms t in the tableau T_m, $(t\sigma)^{\mathbf{I},\nu} = t^{\mathbf{I},\mu}$ and therefore, since $(\mathbf{M}^*, \mu) \models B$ where B is a branch in T_m, as well $(\mathbf{M}^*, \nu) \models B\sigma$; $B\sigma$ is a branch in $T_m\sigma = T_{m+1}$ and thus $(\mathbf{M}^*, \nu) \models T_{m+1}$.

The construction of $\mathbf{M}^*$ does depend only on the set Φ and not on the tableaux, so we not only have shown that a tableau for a satisfiable formula set Φ is satisfiable, but that there is a single structure $\mathbf{M}^*$ satisfying all tableaux for Φ.

THEOREM 2. (Soundness). *If there is a tableau proof for a set $\Phi \subset \mathcal{L}_\Sigma$ of sentences, then Φ is unsatisfiable.*

Proof. There is a tableau proof for Φ, i.e., a closed tableau T for Φ. Since all branches in T are closed and thus contain a complementary pair of literals, $\neg$*true,* or *false,* there is no canonical structure $\mathbf{M}^*$ and no assignment μ such that $(\mathbf{M}^*, \mu)$ satisfies any of the branches of T; therefore T is unsatisfiable. If Φ were satisfiable, then T would be satisfiable as well (Lemma 3), which would lead to a contradiction.

4.2. *Completeness Proof Using Hintikka Sets*

We start with the "classical" version of the completeness proof for free variable semantic tableaux based on Hintikka sets proceeding as follows: Hintikka sets are downward saturated formula sets that do not contain contradictions on the literal level; it is shown that Hintikka sets are satisfiable. For a given set Φ of sentences an infinite tableau T_∞ is defined as the result of

an infinite sequence of expansion rule applications. It turns out that if the rules are applied in a *fair* manner and if T_∞ contains an open branch B, then there is a substitution σ_∞ such that $B\sigma_\infty \cup \Phi$ is a Hintikka set, implying that $B\sigma_\infty \cup \Phi$ and thus Φ is satisfiable. Therefore, if Φ is unsatisfiable, then $T_\infty\sigma_\infty$ must be closed. But then there is a *finite* subtableau T of T_∞ such that $T\sigma_\infty$ is closed. It remains to show that σ_∞ can be decomposed into most general closing substitutions $\sigma_1, \ldots, \sigma_r$ such that $T\sigma_1 \cdots \sigma_r$ is closed.

DEFINITION 12. *A set $H \subset \mathcal{L}_{\Sigma^*}$ of sentences is a* Hintikka *set if it satisfies the following conditions: (1) false $\notin H$, $\neg$true $\notin H$, and there are no complementary literals in H. (2) If $\alpha \in H$, then all α_i are in H. (3) If $\beta \in H$, then some β_i is in H. (4) If $\gamma(x) \in H$, then $\gamma_1(t) \in H$ for all ground Σ^*-terms t. (5) If $\delta(x) \in H$, then $\delta_1(t) \in H$ for some ground Σ^*-term t.*

LEMMA 4. (Hintikka). *If H is a Hintikka set, then H is satisfiable.*

Proof. A term model $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ of H can be defined by: $t^{\mathbf{I}} = t$ for all $t \in \mathbf{D}$, and $p^{\mathbf{I}}(t_1, \ldots, t_k) = \mathit{true}$ iff $P(t_1, \ldots, t_k) \in H$ for all ground atoms $P(t_1, \ldots, t_k)$ over Σ^*. By induction on the structure of formulae in H it is easy to prove that $\mathbf{M}$ satisfies H.

DEFINITION 13. *The construction of a sequence $(T_n)_{n\geq 1}$ of tableaux for $\Phi \subset \mathcal{L}_\Sigma$, where T_n is obtained from T_{n-1} by applying a tableau expansion rule, is* fair *if the following holds for all branches B in the infinite tableau that is approximated by that sequence: (1) All α, β, and δ occurring on B or in Φ have been used to expand B (by applying the appropriate expansion rule). (2) All γ occurring on B or in Φ have been used infinitely often to expand B (by applying the γ-rule).*

One may construct a fair sequence of tableaux for any set of sentences. Combining the fair application of the expansion rule with fair application of the closure rule, however, is a difficult problem (see Section 5).

THEOREM 3. (Completeness). *If the set $\Phi \subset \mathcal{L}_\Sigma$ of sentences is unsatisfiable, then there is a tableau proof for the unsatisfiability of Φ.*

Proof. Let $(T_n)_{n\geq 1}$ be a fair sequence of tableaux for Φ starting with the tableau consisting of the single node *true*; this sequence approximates the infinite tree T_∞. We define a particular substitution σ_∞ as follows: let $(B_k)_{k\geq 1}$ be an enumeration of the branches of T_∞. Let $(\phi_i)_{i\geq 1}$ be an enumeration of the γ-formulae in T_∞. For every γ-formula ϕ_i, if ϕ_i occurs on B_k then let x_{ijk} be the (new) variable that has been introduced by the j-th application of the γ-rule to ϕ_i on B_k. Let $(t_j)_{j\geq 1}$ be an enumeration of all ground terms over Σ^*.

To ensure that the branch B_k is a potential source of a model, the instances of ϕ_i on $B_k\sigma_\infty$ must "cover" all the ground terms t_j. To this end choose $\sigma_\infty(x_{ijk}) = t_j$ for all $i, j, k \geq 1$.

The construction of σ_∞ and the fact that T_∞ is constructed in a fair way ensures that: if B is a branch in T_∞ and $B\sigma_\infty$ is open, then $B\sigma_\infty \cup \Phi$ is a Hintikka set, and thus Φ is satisfiable. Since this would contradict the assumption of the theorem, we can conclude that there is a substitution σ (for example σ_∞) such that $T_\infty\sigma$ is closed. Because $T_\infty\sigma$ is a finitely branching tree and the distance of all the complementary literals that close branches to its root node is finite, König's Lemma[7] applies, and there has to be an $n \geq 1$ such that the finite tableau $T_n\sigma$ is closed.

The substitution σ, however, is in general not a *most general* unifier of complementary literals, and cannot be used in an MGU closure rule application to T_n. Therefore, it remains to show that σ can be suitably decomposed: $\sigma = \sigma' \circ \sigma_r \circ \sigma_{r-1} \circ \cdots \circ \sigma_1$, where σ_i is a most general closing substitution for the instantiation $B_i\sigma_1\sigma_2 \ldots \sigma_{i-1}$ of the i-th branch B_i in T_n ($1 \leq i \leq r$); σ' is the part of σ that is not actually needed to close T_n. The σ_i are inductively constructed as follows:

Let $\sigma'_1 = \sigma$. For $1 < i \leq r$, let σ_i be a most general substitution such that (1) σ'_{i-1} is a specialization of σ_i; that is, there is a substitution σ'_i such that $\sigma'_{i-1} = \sigma'_i \circ \sigma_i$; and (2) σ_i is a closing substitution for $B_i\sigma_1\sigma_2 \ldots \sigma_{i-1}$. Now σ_i is a most general *closing* substitution of $B_i\sigma_1\sigma_2 \ldots \sigma_{i-1}$. Otherwise, there is a closing substitution σ''_i being more general than σ_i. The is-more-general relation is transitive, hence σ''_i is more general than σ'_{i-1} in contradiction to σ_i being a most general substitution satisfying (1) and (2). Finally, let $\sigma' = \sigma'_r$.

To obtain a Hintikka set from a fairly constructed sequence of tableaux it suffices to apply the appropriate expansion rule exactly once to each α-, β-, or δ-formula on each branch. This has the practically relevant consequence that only to γ-formulae must a rule be applied more than once per branch.

4.3. *Anderson-Bledsoe Completeness Proof*

As an alternative to Hintikka-style completeness proofs, Anderson & Bledsoe's (1970) method for proving completeness of *resolution* can be adapted to tableaux. In contrast to the Hintikka proof, which involves induction on the formula structure, this style of proof works by induction on the size of the signature, leading to a quite different pattern: while in the Hintikka proof

[7] "A tree that is finitely branching but infinite must have an infinite branch." A proof is, for example, in (Fitting, 1996).

any complete and fair (open) tableau is shown to contain enough information to construct a model, the Anderson-Bledsoe argument directly constructs a tableau for a given unsatisfiable formula from suitable subtableaux provided by the induction hypothesis. This makes it possible to exclude certain tableaux from being acceptable as proofs by imposing additional conditions on the subproofs used in the construction. In contrast to this, at the heart of the Hintikka-style proof lies the saturation of a formula set. Whenever such a saturation is possible the corresponding calculus must be *proof confluent*, i.e., every partial tableau proof for an unsatisfiable formula can be extended to a closed tableau. This means that this proof method is not suitable for non-proof confluent tableau variants as defined in Section 6.2 below.

The proof given below is taken (slightly modified) from (Hähnle et al., 1997). We only consider tableaux for propositional formulae, but completeness of free variable tableaux can be proven using a similar technique based on enumerating instantiations as in the previous section. In the proof, we make use of the following equivalences: (1) $\neg \mathit{false} \equiv \mathit{true}$; (2) $\neg \mathit{true} \equiv \mathit{false}$; (3) $\bigwedge_{i=1}^{n} \alpha_i \equiv \bigwedge_{i=1, i\neq j}^{n} \alpha_i$ if $\alpha_j = \mathit{true}$; (4) $\bigvee_{i=1}^{n} \beta_i \equiv \bigvee_{i=1, i\neq j}^{n} \beta_i$ if $\beta_j = \mathit{false}$; (5) $\bigwedge_{i=1}^{n} \alpha_i \equiv \mathit{false}$ if $\alpha_j = \mathit{false}$; (6) $\bigvee_{i=1}^{n} \beta_i \equiv \mathit{true}$ if $\beta_j = \mathit{true}$. A formula to which these rules have been applied is called *simplified*.

THEOREM 4. *If $\Phi \subset \mathcal{L}_\Sigma$ is any unsatisfiable finite set of simplified propositional formulae, then there is a tableau proof for the unsatisfiability of Φ.*

Proof. Let Φ be as above; we proceed by induction on the number n of distinct atoms in Φ.

If $n = 0$, then $\Phi = \{\mathit{false}\}$ (as it is simplified), and there is nothing to prove; so assume the theorem holds for all formula sets with at most n distinct atoms, and let Φ be a set of formulae with $n+1$ distinct atoms.

Let Φ_p be the set of formulae produced as follows: first replace in Φ each positive occurrence of p with *false*, call the result Φ'_p. By a routine argument Φ'_p is still unsatisfiable.[8] Also, since p does now only occur negatively in Φ'_p, all remaining occurrences of p may be replaced with *true*, again preserving unsatisfiability (the latter can be seen as a non-clausal version of the Pure Rule, see (Ramesh, 1995)). Finally, simplify to obtain Φ_p.

By the induction hypothesis, there is a closed tableau T_p for Φ_p. Let T'_p be the tableau tree produced by applying each extension in T_p to the corresponding formulae in Φ. If T'_p is closed, we are done. If not, then all open branches in T'_p must result from formulae containing p.

[8] Here is a sketch: assume $\mathbf{I}$ satisfies Φ'_p, but not Φ. As p does not occur positively in Φ'_p we can safely modify $\mathbf{I}$ at p to be *false*. Then $\mathbf{I}$ still satisfies Φ'_p, but by definition of Φ'_p it must satisfy Φ as well—contradiction.

A formula that contains a positive occurrence of p is a formula of Φ in which this p was replaced with *false* to obtain the corresponding formula of Φ_p. Whether p occurred in an α- or in a β-subformula, in both cases a branch of T_p that was closed because it contained *false* is open in T'_p and contains p (because simplification cannot propagate over changing types of subformulae).

Dually, negative occurrences of p are replaced with *true* and a closed branch of T_p containing $\neg$*true* is open in T'_p and contains $\neg\neg p$. By a single α-rule application this gives p on each such branch. Hence, all open branches in T'_p contain nodes labeled p.

Similarly, by replacing negative occurrences of p by *false*, positive occurrences by *true*, followed by simplification one produces the set of formulae $\Phi_{\overline{p}}$. The induction hypothesis provides a proof $T_{\overline{p}}$ of $\Phi_{\overline{p}}$ and a corresponding proof tree $T'_{\overline{p}}$ of Φ. The leaves of all open branches of $T'_{\overline{p}}$ are labeled $\neg p$.

Finally, to obtain a closed tableau for Φ from $T'_{\overline{p}}$ and T'_p, we append $T'_{\overline{p}}$ to all open branches of T'_p and observe that any branch not closed within $T'_{\overline{p}}$ or within T'_p has nodes labeled p and labeled $\neg p$ and thus is closed.

5. Resolving the Indeterminism

In this section we discuss the difficulties that emerge if one wants to define a concrete, i.e. deterministic, (and complete) procedure that systematically looks for free variable tableau proofs.

In the case of ground tableaux this is relatively easy: their rules are *non-destructive*, thus it suffices to add systematically all ground instances of γ-formulae until a branch is closed, after which the next branch is considered. Any fair selection of ground instances together with König's lemma guarantees completeness.

DEFINITION 14. *A tableau calculus is* non-destructive *if all tableaux that can be derived from a given tableau T contain T as an initial subtree; otherwise the calculus is* destructive.

In free variable tableaux the closure rule obviously renders the calculus destructive. In the completeness proof this situation was resolved by restricting the closure rule such that it is only applied when the whole tableau can be closed. Then, a fair selection of *free variable* instances of γ-formulae suffices. Such a procedure seems impractical, because (1) it requires to store the whole tableau, and (2) after each extension step the whole tableau must be tested for

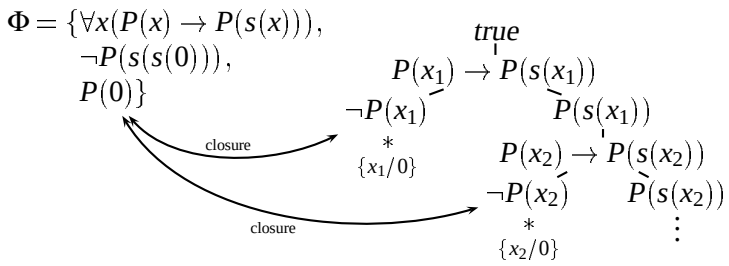


Figure 2. Incompleteness caused by unfair *select pair*.

closure. So far no techniques have been developed to deal with this problem efficiently.

In the remainder of the section, we discuss variants of free variable tableaux, where the closure rule is unrestricted in the sense that closure of the whole tableau is not required, rather, closure of at least one branch suffices to trigger its application. With unrestricted substitution, the free variable tableau calculus is *proof confluent* (although it is destructive). This is trivial by the fact that a closed tableau (which is guaranteed to exist by completeness) can always be appended to each open branch.

A much more difficult problem is to explicitly specify a deterministic construction rule for destructive free variable tableaux that is complete. The problem is that different possibilities to close a certain branch can be mutually exclusive. When the wrong choice is made and, thus, the wrong substitution is applied to the tableau, it may become impossible to use the next (and possibly more usuful) branch closure immediately afterwards. Instead, it may become necessary to repeat the sequence of expansion rule applications that lead to the situtation in which the wrong choice was made; moreover the original situation may have to be reconstructed on each branch that has been generated in the meantime and that cannot be closed because of the bad choice. A practically convincing solution has so far proved elusive, but see (Billon, 1996) for a promising suggestion.

With a few examples, we illustrate incompleteness phenomena arising from unfair selection strategies for the various kinds of choice points. Needless to say, these can also interact in a complex way. The examples are more naturally formulated with the implication connective; for the tableau rules, recall that $\phi \to \psi$ abbreviates $\neg\phi \vee \psi$.

In Figure 2, the literal $P(0)$ is preferred in closures resulting in appending the same instance of $\neg P(x_i)$ time and again. In Figure 3 the γ-formula is preferred for rule application thus delaying expansion of the inconsistent second formula indefinitely. Finally, in Figure 4, a branch is closed as early as possible. Independently of which branch is closed first, the variable x gets

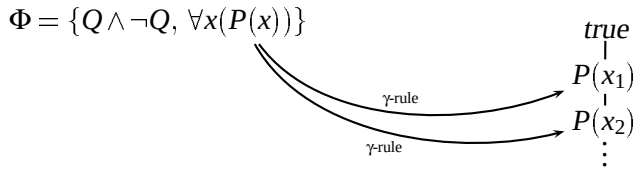


Figure 3. Incompleteness caused by unfair *select formula*.

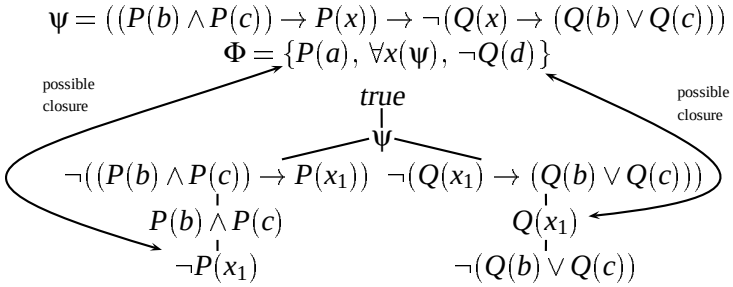


Figure 4. Incompleteness caused by unfair *select mode*.

"used up" by a substitution that blocks closure of the other branch. Of course, a second free variable instance of the γ-formula may be created, but then the same happens one level below etc. The example highlights the problems of *destructive* free variable tableaux.

We discuss remaining alternatives for free variable tableau proof search. It will be useful to visualize the AND-OR search tree spanned by the non-deterministic tableau procedure in Section 2: each non-deterministic action *select branch*, *select formula*, *select pair*, and *select mode* creates an OR node with as many successors as there are alternatives; the recursive call of the procedure on each new branch creates an AND node. Each search tree is finitely branching (provided the closure rule suitably restricts the choice of a closing substitution, as does, for instance, the MGU closure rule); branches are either finite and end with a *select pair* action or they have infinite length (if a γ-formula is accessible). In Figure 5 the start of a search tree is displayed.

No OR nodes are created for *select branch* alternatives. This is because all branches of a tableau have to be closed, so different branch selection strategies merely correspond to different traversals of the search tree. Less obvious, but simple enough is the observation that OR nodes arising from *select formula* alternatives can be eliminated as well provided that in the remaining branches of the search tree each free variable instance of each γ-formula does occur. This can be achieved with a fair selection strategy.

Possibly, further branches in the remaining search tree can be removed.

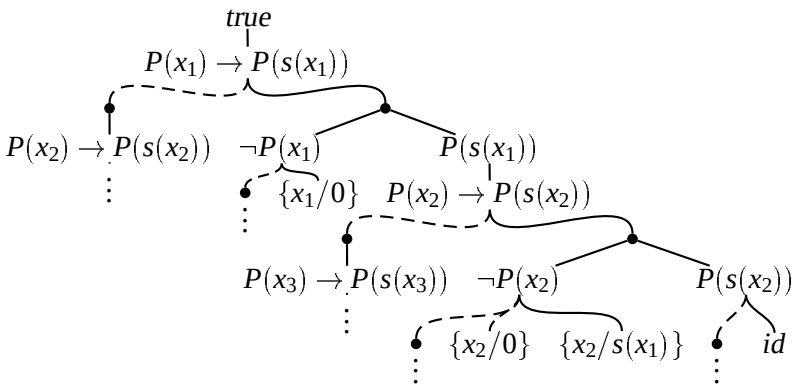


Figure 5. Start of AND-OR search tree for finding a tableau proof for the set $\{p(0), \forall x(p(x) \to p(s(x))), \neg p(s(s(0)))\}$ of sentences. OR arcs are indicated by curly braces, AND arcs are straight lines. The dashed parts of braces constitute alternatives that were not selected in the actual proof. The solid parts of braces represent a successful proof. Mode and pair selection are combined in one OR node.

Assume, for example, there are substitutions σ and later τ occurring during a tableau proof search whose supports (the *support* of a substitution σ is the set of variables on which σ is *not* the identity) have an empty intersection. Then it is unnecessary to consider the part of the search tree, where the sequence of applying σ and τ is reversed, because $\tau\sigma = \sigma\tau$. Redundancies of this kind are hard to detect efficiently, though.

We return to the problem of finding a successful proof in our AND-OR search tree, a common AI search problem. Unrestricted *depth first* search is excluded because of the difficulties discussed above to find a selection strategy that ensures completeness, leaving *breadth first* and *depth first iterative deepening* search.

For both the concept of a *completion mode* is useful: this is a monotone function m from $\mathbb{N}$ to sets of tableaux such that $\bigcup_{i\in\mathbb{N}} m(i)$ includes all possible tableaux. Let $M(i)$ be the part of the tableau search tree that contains all tableaux in $m(i)$, but not the ones in $m(j)$ for $j < i$. As search trees have infinite depth, breadth first search has to consider $M(i)$ for some i, which is guessed. As breadth first search is space expensive and for all practical completion modes $|m(i)|$ grows exponentially in i, it has been suggested by Stickel (1988) to use depth first iterative deepening (DFID) search (Korf, 1985): successively search $\bigcup_{j<i} M(j)$ for $i = 0, 1, 2, \ldots$ causing only polynomial overhead as compared to a breadth first search at "the right level."

A fundamental advantage of DFID over breadth first search is that it can be implemented efficiently via bounded depth first search and backtracking

as in (Beckert and Posegga, 1995). Although this leads to acceptable performance of tableau-based automated theorem provers, it should be stressed that DFID search is only a compromise while a complete selection strategy without backtracking (making full use of the proof confluency of analytic tableaux) is not yet available.

6. Optimizations

6.1. *A Classification of Optimizations*

Below, the main types of optimizations of analytic tableaux are described. Most known variants of tableaux belong to one of these classes (although the classes are not completely disjoint).

1. Restrictions that forbid certain rule applications to avoid parts of the search space that (a) are symmetrical to or subsumed by other parts, or that (b) for some reason are known not to contain a proof; typical examples are for (a) the *regularity* condition (Section 6.3) and for (b) the *connectedness* condition (Section 6.2).

2. Changes to the tableau rules or the introduction of additional rules that strengthen the calculus, i.e., allow to derive additional tableau proofs; this only makes sense if the additional proofs that can be found are shorter and replace (subsume) *several* other proofs; an example is the universal formula technique (Section 6.4), allowing to use more general closing substitutions.

3. Optimizations making use of knowledge accumulated during proof search (a) for restricting and/or rearranging the search space (for example *pruning* of redundant branches, Section 6.5), or (b) for reusing parts of the already constructed proof (like local lemmata).

6.2. *Links*

One of the first crucial advances in resolution-based theorem proving was the introduction of the set-of-support (SOS) strategy (Wos et al., 1965). It has the effect of preventing deduction steps that are unrelated to previous ones.

A similar effect can be achieved with tableaux. The basic idea is that a formula used for extension should lead to the closure of at least one branch. When all formulae are clauses this amounts to saying that the clause used for extension and the branch on which it is used must contain a complementary pair of literals. Several calculi based on this idea are discussed in detail in Chapters I.1.2 and I.1.3 of this volume. In the non-clausal case a little more effort must be spent.

Recall that skolemization (Lemma 2) provides the option of eliminating all δ-subformulae from a formula before any other rules are applied and yet preserves tableau semantics, i.e., satisfiability under canonical models. Similarly, we define a *free variable instance* of a formula by (a) replacing each positive subformula occurrence of some $\gamma(x)$ by $\gamma_1(x')$, and (b) replacing each negative occurrence of $\overline{\gamma(x)}$ by $\overline{\gamma_1(x')}$; where the x' are new variables.

DEFINITION 15. *Given (not necessarily closed) formulae $\phi_1, \phi_2 \in \mathcal{L}_{\Sigma^*}$, the formula ϕ_1 has a* link *into ϕ_2 (is linked to ϕ_2) with MGU σ iff free variable instances ϕ_1' and ϕ_2' of their skolemizations contain literals ρ_1 and ρ_2, respectively, with different polarity (i.e., one literal occurrence is positive and one is negative), and ρ_1, ρ_2 are unifiable with MGU σ. If one of ϕ_1, ϕ_2 is a set of formulae it is treated as the conjunction of its elements.*

A formula $\phi \in \mathcal{L}_{\Sigma^*}$ has a link into itself *(is linked to itself) with MGU σ iff there is an α-subformula α of ϕ such that two immediate tableau subformulae α_i and α_j, $i \neq j$, of α are linked with MGU σ.*

EXAMPLE 4. *The formula $P(x)$ has a link into the formula $\neg(q \vee P(a))$ with MGU $\{x \leftarrow a\}$. The α-formula $\neg(p \vee \neg p)$ has a link into itself; whereas the β-formula $p \vee \neg p$ is* not *linked to itself.*

DEFINITION 16. *A tableau T for Φ is* weakly connected *iff for all expansion rule applications used in its construction the following holds: if the rule has been applied to a formula ϕ extending a branch B, then the instance ϕ' of ϕ that occurs in Φ or on the (sub-)branch B' of T (which is an instance of B) has a link into $B' \cup \Phi$ or is linked to itself.*

It is connected *iff the link is always from ϕ' into (a) a formula of B' that appears below the node on B' that corresponds to the last branching node of B, or (b) into Φ if there is no branching node in B.*

EXAMPLE 5. *The formulae to which an expansion rule is applied to construct the tableau that is shown in Figure 2 are $\phi_0 = \forall x(P(x) \rightarrow P(s(x)))$ and $\phi_i = P(x_i) \rightarrow P(s(x_i))$ for $i \geq 1$. The formula ϕ_0 is identical to its instances, which have a link into Φ (namely to the atoms of Φ); all instances of the formulae ϕ_i are linked to ϕ_0 and so have a link into Φ. Thus, the tableau is weakly connected. It is, however,* not *connected, as the instance $\phi_2' = P(0) \rightarrow P(s(0))$ of ϕ_2 is not linked to the atom $P(s(0))$, which is the only formula below the last branching point of the branch expanded by applying the β-rule to ϕ_2'.*

This shows that the unfair choice of complementary pairs exemplified in Figure 2 cannot be avoided by weak connectedness. If, however, the connectedness condition is observed, at least this type of unfair choice is avoided (other types of wrong or unfair choice are, of course, still possible).

The tableau shown in Figure 3 is not *weakly connected (hence, not connected) because the formula $\forall x(P(x))$ used for expansion does not have a link to any formula in Φ or on the branch.*

Observe that the destructive closure rule of free variable tableaux creates a serious implementation challenge as its application may injure weak connectedness at any point. In (Pape, 1996) an implementation using term constraints is suggested. Alternatively, one applies the MGU corresponding to a connection immediately and admits backtracking over extensions. In this latter version, connected tableaux, the connection method (Bibel, 1982), and matings (Andrews, 1981) can be considered to be notational variants of each other. Restricted to CNF, connected tableaux are also closely related to model elimination (Loveland, 1969). Variants of connected CNF tableaux are discussed in great detail in Chapters I.1.2, I.1.3, and I.1.5 of this volume.

Both notions of connectivity can be refined further by *regularity* (see Section 6.3) and weakly connected tableaux can in addition be refined with literal orderings (Hähnle and Klingenbeck, 1996).

Connected tableaux are not proof confluent—even on the propositional level—, as the simple example $\Phi = \{(p \wedge \neg p) \vee q, r \wedge \neg r\}$ shows: if the first formula (that has a link into itself) is used for extension, there is no way to obtain a connected closed tableau from there. It is necessary to take the extending formulae from a minimally unsatisfiable subset (MUS) of Φ. Completeness of refinements of this kind can be proven with the Anderson-Bledsoe technique, which is compatible with considering an MUS. On the other hand, proof confluent refinements are best tackled with a saturation-based method. Below, we show paradigmatically how the basic techniques for proving completeness from Section 4 are revamped to deal with more advanced calculi.

THEOREM 5. *If Φ is any unsatisfiable finite set of simplified propositional formulae, then there exists a closed connected tableau for it.*

Proof. We proceed as in the proof of Theorem 4, but make two modifications: first, one restricts attention to a minimally unsatisfiable subset of Φ; second, one notes that the proof still goes through if the induction hypothesis is strengthened as follows:

For all minimally unsatisfiable sets of simplified formulae Φ with at most n distinct atoms and any non-literal $\phi \in \Phi$, there exists a closed connected tableau in which the first rule is applied to ϕ.

As before, $n = 0$ is trivial and so is $n = 1$ when there are only literals in Φ. Thus, for the induction, take any atom p such that there are formulae $\phi \in \Phi$ containing a positive occurrence of p and $\phi' \in \Phi$ containing a negative occurrence of p. Then construct Φ_p and $\Phi_{\overline{p}}$ as before, but instead of these

sets themselves use any minimally unsatisfiable subsets that still contain ϕ_p resp. $\phi'_{\overline{p}}$ (the proof for the existence of these minimally unsatisfiable subsets is not hard, but it requires some technical definitions, see (Hähnle et al., 1997)).

This time the induction hypothesis gives closed connected tableaux T_p for Φ_p and $T_{\overline{p}}$ for $\Phi_{\overline{p}}$ in which the first rule has been applied to ϕ_p resp. to $\phi'_{\overline{p}}$. As before, from these one obtains tableaux T'_p, $T'_{\overline{p}}$ in which all open branches contain p resp. $\neg p$. Observe that the branches of $T'_{\overline{p}}$ containing p do not exist in T_p and thus are never extended therein; therefore p occurs *after the last branching point* on these branches.

Finally, the fact that T'_p, $T'_{\overline{p}}$ start with rule applications to formulae containing p resp. $\neg p$ implies that appending $T'_{\overline{p}}$ to the open branches of T'_p gives a connected tableau for Φ (as ϕ' and p are linked by definition) starting with a rule application to ϕ.

6.3. *Regularity*

Regularity, another well known refinement from clausal tableaux (see Chapter I.1.2) is also defined in the non-clausal case (Hähnle and Klingenbeck, 1996; Hähnle et al., 1997).

The following definition of the (ir-)regularity of a formula ϕ takes only the *immediate* tableau subformulae of ϕ into concern. It is possible to give a more elaborate definition that takes all tableau subformulae of ϕ into concern (Hähnle and Klingenbeck, 1996).

DEFINITION 17. *A formula $\phi \in \mathcal{L}_{\Sigma^*}$ is* regular *w.r.t. a branch B of a tableau for $\Phi \subset \mathcal{L}_\Sigma$ iff (1) ϕ is an α- or δ-formula and all immediate tableau subformulae of ϕ are in $B \cup \Phi$, or (2) ϕ is a β-formula and some $\beta_i \in B \cup \Phi$.*

A tableau T for $\Phi \subset \mathcal{L}_\Sigma$ is regular *iff, for each expansion rule applications used in its construction, the following holds, where the rule has been applied to a formula ϕ extending a branch B: the instance ϕ' of ϕ on the (sub-)branch B' of T (which is an instance of B) is regular w.r.t. B'.*

A formula that is regular w.r.t. a branch may become irregular through the application of a substitution (take, for instance, a branch containing the formulae $P(x) \vee q$ and $P(y)$, and the substitution $\{x/a, y/a\}$). This is a serious implementation problem in free variable tableaux; see (Letz et al., 1992) for a possible solution. Contrary to the clausal case, neither a formula occurring more than once on a certain branch, nor a tableau branch that is a subset of another branch implies irregularity. Take, for example, a closed tableau for the formula $\neg p \wedge (p \vee (p \wedge p \wedge q))$: one of its branches is a proper subset of the other branch, and the latter contains two occurrences of p.

THEOREM 6. *If the set $\Phi \subset \mathcal{L}_\Sigma$ of sentences is unsatisfiable, then there is a* regular *tableau proof for the unsatisfiability of* Φ.

Proof. The completeness proof for free variable tableaux without the regularity condition can easily be adapted. The only difference is that a tableau T'_∞ is used instead of T_∞: all expansion rule applications that are part of constructing T_∞ and that violate the regularity condition are left out; the result is T'_∞. Because of the definition of irregular formulae, the set $B'_\infty\sigma_\infty$ is still a Hintikka set (where B'_∞ is the branch of T'_∞ that corresponds to, and is a subset of, the branch B_∞ of T_∞).

A similar argument can be used to prove completeness of many refinements of free variable tableaux (Hähnle and Klingenbeck, 1996); one shows that any open branch of an infinite tableau T_∞ constructed in a fair way is still a (subset of) a Hintikka set, even if the saturation conditions of Def. 13 do not apply to all formulae of the branch.

6.4. *Universal Formulae*

A formula is often needed in several instances in order to close a branch (or a subtableau) with different substitutions for the free variables occurring in it. In free variable tableaux the mechanism to do so is to apply the γ-rule multiply to generate several instances of ϕ with different free variables. Free variables in tableaux are *not* implicitly universally quantified (as it is, for instance, the case with variables in clauses when using a resolution calculus), but are *rigid*: a substitution must be applied to all occurrences of a free variable in a tableau.

Suppose we have a branch B with a formula $\phi(x)$ on it; assume further that the expansion of the tableau then proceeds with creating new branches. Some of these branches contain occurrences of x; for closing the generated branches, the same substitution for x has to be used on all of them. For example, we might have a tableau for $\Phi = \{\neg P(a) \vee \neg P(b), \forall x(P(x))\}$ that consists of two branches, one containing $P(x)$ and $\neg P(a)$, and the other containing $P(x)$ and $\neg P(b)$. This tableau cannot be closed immediately as no single substitution closes both branches. To find a proof, the γ-rule has to be applied again to create another instance of $P(x)$. In the example, as a logical consequence of Φ and the formulae already on the tableau (in a sense made precise in Def. 18), $\forall x(\phi(x))$ can be added to B. In such cases, different substitutions for x can be used without destroying soundness of the calculus. The tableau above then closes immediately. Recognizing such situations and exploiting them allows to use more general closing substitutions, yields shorter tableau proofs, and in most cases reduces the search space.

DEFINITION 18. *Suppose ϕ is a formula on a branch B of a tableau T for $\Phi \subset \mathcal{L}_\Sigma$. Let T' result from adding $\forall x \phi$ to B for some $x \in$ Var. Then, ϕ is* universal *on B with respect to x if $T \models T'$, where T and T' are identified with the disjunctions of their branches, which in turn are the conjunctions of their labels.* $\mathrm{UVar}(\phi)$ *is the set of all variables w.r.t. which ϕ is universal.*

Instead of designing a closure rule that takes universal formulae into account (replacing rule (3) in Def. 6), we generalize the concept of unifier:

DEFINITION 19. *A substitution σ is a* unifier *of formulae ϕ, ϕ' on a branch of a tableau T if it is the restriction of a substitution τ with the property $(\phi\pi)\tau = (\phi'\pi)\tau$ to $\mathrm{Var} \setminus U$, where $U = \mathrm{UVar}(\phi) \cap \mathrm{UVar}(\phi')$ and π is a renaming of the variables in U with variables new to T.*

With the closure rule based on this modified concept of unification, a tableau proof with less applications of expansion rules than in the standard free variable tableau calculus may be found; the calculus is strengthened.

Recognizing universal formulae is undecidable in general, however, an important class can be recognized easily (and this can already shorten tableau proofs exponentially): a formula ϕ on a branch B of a tableau T is universal w.r.t. x if all branches B' of T containing an occurrence of x that is not on B as well are closed; this holds in particular if the branch B contains all occurrences of x in T. In any sequence of tableau rule applications with a variable x introduced by γ-rule application and not distributed over different branches by β-rule application, the above criterion is obviously satisfied and all formulae generated in this sequence are universal w.r.t. x, formally:

LEMMA 5. *A formula ϕ on a branch B of a tableau T is universal w.r.t. x on B if in the construction of T the formula ϕ was added to B by applying (1) a γ-rule and x is the free variable introduced; (2) an α-, γ-, or δ-rule to a formula that is universal on B w.r.t. x; or (3) a β-rule to a formula β that is universal on B w.r.t. x, and x does not occur in any $\beta_i \neq \phi$.*

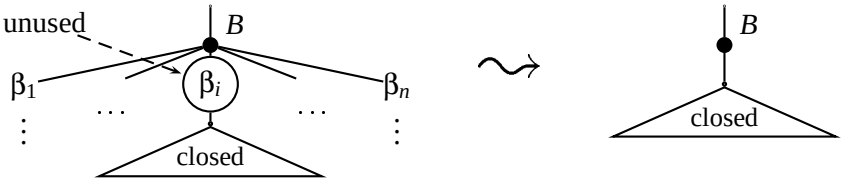
The soundness proof of free variable tableaux (Section 4.1) can accomodate the universal formula technique.

An adaption of the universal formula technique to clausal tableaux is discussed in Section 5.1 of the following chapter. Bibel (1982) proposed a technique for reducing the size of proofs in the connection method, called *splitting by need*; like universal formulae it is based on the idea to avoid copying a universally quantified formula in cases where it is sound to use a single copy with different variable instantiations.

6.5. *Pruning*

Pruning, which is closely related to the *condensing* technique described in (Oppacher and Suen, 1988), allows the reduction of both the size of the search space and the size of generated tableau proofs.

Suppose a branch B of a tableau was extended by a β-rule application and one of the extensions β_i was *not* used to close the subtableau T_i below β_i, then T_i is still closed when appended to any of the other extensions β_j, $j \neq i$, or even immediately to B (the extension β_i is *used* if β_i itself or any of its tableau subformulae is a literal used in an application of the closure rule). To make use of this situation, either the closure rule is changed such that all branches in the tableau containing B as a subbranch are considered to be closed, or—similarly—all branches containing one of the β_j are *pruned*, i.e., the effects of the β-rule application are removed:



ACKNOWLEDGEMENTS

We thank Uwe Egly, Reinhold Letz, Don Loveland, Neil Murray, and Benjamin Shults for comments on earlier versions of this chapter.

REFERENCES

Anderson, R. and W. Bledsoe: 1970, 'A Linear Format for Resolution with Merging and a New Technique for Establishing Completeness'. *JACM* **17**, 525–534.

Andrews, P. B.: 1981, 'Theorem Proving through General Matings'. *JACM* **28**, 193–214.

Baaz, M. and C. Fermüller: 1995, 'Non-elementary Speedups between Different Versions of Tableaux'. In: *Proc. 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, St. Goar, Germany*. pp. 217–230.

Beckert, B., R. Hähnle, and P. H. Schmitt: 1993, 'The Even More Liberalized δ-Rule in Free Variable Semantic Tableaux'. In: G. Gottlob, A. Leitsch, and D. Mundici (eds.): *Proc. 3rd Kurt Gödel Colloquium, Brno, Czech Republic*. pp. 108–119.

Beckert, B. and J. Posegga: 1995, 'leanTAP: Lean Tableau-based Deduction'. *J. of Automated Reasoning* **15**(3), 339–358.

Bibel, W.: 1982, *Automated Theorem Proving*. Vieweg, Braunschweig.

Billon, J.-P.: 1996, 'The disconnection method: a confluent integration of unification in the analytic framework'. In: *Proceedings, 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Terrasini, Italy*. pp. 110–126.

Broda, K.: 1980, 'The Relationship between Semantic Tableaux and Resolution Theorem Proving'. In: *Proceedings, Workshop on Logic, Debrecen, Hungary*. Also as technical report, Imperial College, Department of Computing, London, UK.

Brown, F. M.: 1978, 'Towards the Automation of Set Theory and its Logic'. *Artificial Intelligence* **10**, 281–316.

D'Agostino, M., D. Gabbay, R. Hähnle, and J. Posegga (eds.): 1998, *Handbook of Tableau Methods*. Kluwer. To appear.

Fitting, M. C.: 1996, *First-Order Logic and Automated Theorem Proving*. Springer, 2nd edition.

Hähnle, R. and S. Klingenbeck: 1996, 'A-Ordered Tableaux'. *J. of Logic and Computation* **6**(6), 819–834.

Hähnle, R., N. Murray, and E. Rosenthal: 1997, 'Completeness for Linear Regular Negation Normal Form Inference Systems'. In: Z. Ras (ed.): *Proc. Int. Symposium on Methodologies for Intelligent Systems, Charlotte/NC, USA*. pp. 590–599.

Korf, R. E.: 1985, 'Depth-First Iterative Deepening: An Optimal Admissible Tree Search'. *Artificial Intelligence* **27**, 97–109.

Letz, R., J. Schumann, S. Bayerl, and W. Bibel: 1992, 'SETHEO: A High-Performance Theorem Prover'. *J. of Automated Reasoning* **8**(2), 183–212.

Loveland, D. W.: 1969, 'A Simplified Format for the Model Elimination Procedure'. *JACM* **16**(3), 233–248.

Oppacher, F. and E. Suen: 1988, 'HARP: A Tableau-Based Theorem Prover'. *J. of Automated Reasoning* **4**, 69–100.

Pape, C.: 1996, 'Vergleich und Analyse von Ordnungseinschränkungen für freie Variablen Tableau'. TR 30/96, Universität Karlsruhe, Fakultät für Informatik.

Prawitz, D.: 1960, 'An Improved Proof Procedure'. *Theoria* **26**, 102–139. Reprinted in (Siekmann and Wrightson, 1983, vol. 1, pp. 162–199).

Ramesh, A. G.: 1995, 'Some Applications of Non-Clausal Deduction'. Ph.D. thesis, Dept. of Computer Science, SUNY at Albany.

Reeves, S.: 1987, 'Semantic Tableaux as a Framework for Automated Theorem-Proving'. In: C. S. Mellish and J. Hallam (eds.): *Advances in Artificial Intelligence (Proceedings of AISB-87)*. pp. 125–139.

Siekmann, J. and G. Wrightson (eds.): 1983, *Automation of Reasoning: Classical Papers in Computational Logic*. Springer.

Smullyan, R. M.: 1968, *First-Order Logic*. Springer, Heidelberg. 2nd corrected edition published in 1995 by Dover Publications, New York.

Stickel, M. E.: 1988, 'A Prolog Technology Theorem Prover'. In: E. Lusk and R. Overbeek (eds.): *Proc. 9th Int. Conf. on Automated Deduction*. pp. 752–753.

Wang, H.: 1960, 'Toward Mechanical Mathematics'. *IBM J. of Research and Development* **4**(1). Reprinted in (Siekmann and Wrightson, 1983, vol. 1, pp. 244–264).

Wos, L., G. A. Robinson, and D. F. Carson: 1965, 'Efficiency and Completeness of the Set of Support Strategy in Theorem Proving'. *JACM* **12**(4), 536–541.