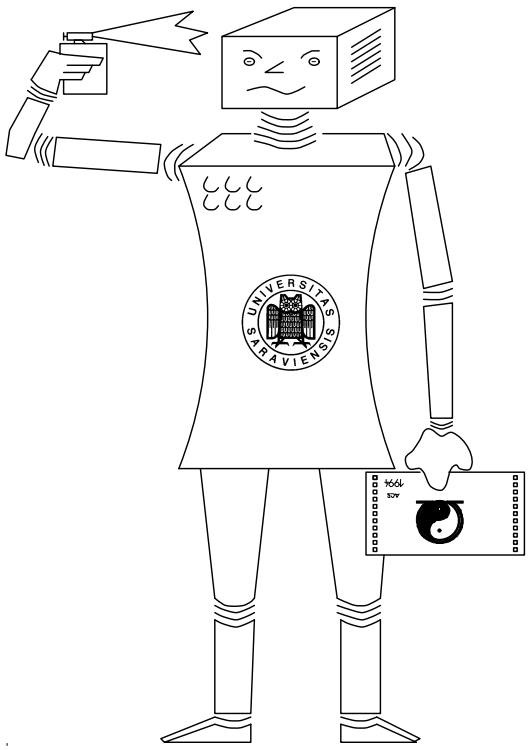
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/



Higher-Order Tableaux

Michael Kohlhase

Published as: To appear at the Tableau-Workshop 1995, Koblenz

Higher-Order Tableaux

Michael Kohlhase

Fachbereich Informatik, Universität des Saarlandes
66041 Saarbrücken, Germany
+49-681-302-4627
kohlhase@cs.uni-sb.de

Abstract. Even though higher-order calculi for automated theorem proving are rather old, tableau calculi have not been investigated yet. This paper presents two free variable tableau calculi for higher-order logic that use higher-order unification as the key inference procedure. These calculi differ in the treatment of the substitutional properties of equivalences. The first calculus is equivalent in deductive power to the machine-oriented higher-order refutation calculi known from the literature, whereas the second is complete with respect to Henkin's general models.

1 Introduction

The history of building automated theorem provers for higher-order logic is almost as old as the field of deduction systems itself. The first successful attempts to mechanize and implement higher-order logic were those of Huet [Hue72] and Jensen and Pietrzykowski [JP76]. They combine the resolution principle for higher-order logic (first studied in [And71]) with higher-order unification. The unification problem in typed λ-calculi is much more complex than that for first-order terms, since it has to take the theory of λ-equality into account. In particular the higher-order unification problem is undecidable and sets of solutions need not even always have most general elements that represent them. Thus the calculi for higher-order logic have taken special measures to circumvent the problems posed by the theoretical complexity of higher-order unification.

This paper shows how the methods developed in the context of higher-order resolution theorem proving can be carried over to the tableau framework. First-order tableaux were introduced by Beth [Bet69] and Hintikka [Hin55] and later unified by Smullyan [Smu68]. The free variable tableaux method has its origin in the work of Prawitz [Pra60] and has further been elaborated by Reeves [Ree87] and Fitting [Fit90].

Generalizing these methods, we develop a tableau calculus $\mathcal{HT}$, which is strongly related to the higher-order matings method [And89] of Andrews and Miller's expansion trees [Mil83, Pfe87]. Just like tableau calculi these methods build trees annotated with formulae by decomposing the propositional structure of the input formulae. The most significant difference is that the test of inconsistency is done by finding a "spanning mating" instead of closing all branches. In this perspective expansion trees correspond to standard tableaux whereas the higher-order mating method corresponds to free variable tableaux, and we have

adapted ideas from both for the development of $\mathcal{HT}$. Therefore it seems reasonable to expect that the methods presented in this paper can be directly carried over to a completeness proof for the higher-order matings method that is still missing in the literature.

For the completeness proofs for these calculi we need to generalize the notion of a general model to the notion of a Σ-model structure. This gives us a semantic characterization of the deductive power in addition to the proof theoretic notion of relative completeness found in the literature so far [And71, Hue72, Mil83, Pfe87]. In fact neither the calculi nor $\mathcal{HT}$ are complete with respect to Henkin models, since they fail to capture the substitutivity of equivalence. We remedy this shortcoming augmenting $\mathcal{HT}$ with a tableau construction rule that uses substitutivity in a goal-oriented way. This "extensional" tableau calculus $\mathcal{HTE}$ is complete with respect to Henkin's general model semantics [Hen50].

2 Higher-Order Logic

Definition 2.1 (Types) Let $\mathcal{BT} := \{o, \iota\}$, then the set $\mathcal{T}$ of **types** is inductively defined to be the set $\mathcal{BT}$ together with all expressions $\alpha \to \beta$, where α and β are types. Here the base type ι stands for the set of **individuals** and the type o for the **truth values**. The functional type $\alpha \to \beta$ denotes the type of functions with domain α and codomain β. The types in $\mathcal{BT} \subseteq \mathcal{T}$ are called **base types**, types of the form $\alpha \to \beta$ are called **functional types**. We use the convention of association to the right for omitting parentheses in functional types, thus $\alpha \to \beta \to \gamma$ is an abbreviation for $(\alpha \to (\beta \to \gamma))$. This way the type $\gamma := \beta_1 \to \ldots \to \beta_n \to \alpha$ denotes the type of n-ary functions, that take n arguments of the types $\beta_1, \ldots, \beta_n$ and have values of type α. To conserve even more space we use a kind of vector notation and abbreviate γ by $\overline{\beta_n} \to \alpha$.

Definition 2.2 (Typed Collection) A collection $\mathcal{D} := \mathcal{D}_{\mathcal{T}} := \{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$ of sets $\mathcal{D}_\alpha$, indexed by the set $\mathcal{T}$ of types, is called a **typed collection (of sets)**. A typed collection $\mathcal{I} := \{\mathcal{I}^\alpha \in \mathcal{F}_p(\mathcal{D}_\alpha; \mathcal{E}_\alpha) \mid \alpha \in \mathcal{T}\}$ of functions is called a **typed function** $\mathcal{I} : \mathcal{D}_{\mathcal{T}} \longrightarrow \mathcal{E}_{\mathcal{T}}$.

We will write finite functions like substitutions or variable assignments as sets of pairs $\varphi := [a^1/X^1], \ldots, [a^n/X^n]$ with the intended meaning that $\varphi(X^i) = a^i$. Furthermore we use the convention that $\psi := \varphi, [a/X]$ assigns a to X and coincides with φ everywhere else. We use the asterisk for union with singletons, so that $\Phi * \mathbf{A}$ stands for $\Phi \cup \{\mathbf{A}\}$.

For the definition of well-formed formulae we fix a **signature**, i.e. a typed collection $\Sigma_{\mathcal{T}}$ of symbols and a countably infinite set $\mathcal{V}$ of variables. While the **constants** in Σ are a priori typed by definition ($\tau(c) = \alpha$, iff $c \in \Sigma_\alpha$) the variables are more volatile objects and obtain their type by a **variable context** Γ, i.e. a partial function from variables to types that we write as a sequence $\Gamma = [X^1 : \alpha^1], \ldots, [X^n : \alpha^n]$.

Definition 2.3 (Well-Formed Formulae) For each $\alpha \in \mathcal{T}$ we define the set $\mathit{wff}_\alpha(\Sigma, \Gamma)$ of **well-formed formulae of type** α inductively by

1. $\Sigma_\alpha \subseteq \mathit{wff}_\alpha(\Sigma, \Gamma)$
2. If $X \in \mathcal{V}$ and $\Gamma(X) = \alpha$, then $X \in \mathit{wff}_\alpha(\Sigma, \Gamma)$.
3. If $\mathbf{A} \in \mathit{wff}_{\beta\to\alpha}(\Sigma, \Gamma)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma, \Gamma)$, then $\mathbf{AB} \in \mathit{wff}_\alpha(\Sigma, \Gamma)$.
4. If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \Gamma, [X{:}\beta])$, then $(\lambda X_\beta \mathbf{.A}) \in \mathit{wff}_{\beta\to\alpha}(\Sigma, \Gamma)$.

We call formulae of the form **AB applications**, and formulae of the form $\lambda X_\alpha\mathbf{.A}$ **λ-abstractions**. We will also denote the fact that a formula $\mathbf{A}$ has the type α with the judgment $\Gamma \vdash_\Sigma \mathbf{A}{:}\alpha$ and often write the type as a subscript $\mathbf{A}_\alpha$, if it is not irrelevant or clear from the context.

Note that this definition gives a dynamic account of variables, only requiring to specify the free variables of a formula in the context. For instance the variable X is discharged from the context Γ, when it is bound by the abstraction.

We adopt the usual definition of **free** and **bound** (all occurrences of the variable X in $\lambda X_\alpha\mathbf{.A}$ are called bound), variables and call a formula **closed**, iff it does not contain free variables. As in first-order logic the names of bound variables have no meaning at all, thus we consider alphabetic variants as identical and use a notion of substitution that systematically renames bound variables in order to avoid variable capture. We refer to formulae of type o as **propositions** and as **sentences** if they are closed.

We denote the constants by lower case letters and the variables by upper case letters and use bold upper case letters $\mathbf{A}_\alpha$, $\mathbf{B}_{\alpha\to\beta}$, $\mathbf{C}_\gamma \ldots$ as syntactical variables for well-formed formulae. In order to make the notation of well-formed formulae more legible, we use the convention that the group brackets (and) associate to the left and that the square dot . denotes a left bracket, whose mate is as far right as consistent with the brackets already present. Additionally, we combine successive λ-abstractions, so that the well-formed formulae $(\lambda X^1\mathbf{.}\lambda X^2\mathbf{.}\ldots\lambda X^n\mathbf{.AE}^1\ldots\mathbf{E}^m)$ and $\lambda X^1\ldots X^n\mathbf{.AE}^1\ldots\mathbf{E}^m$ and $\lambda\overline{X^n}\mathbf{.A}\overline{\mathbf{E}^m}$ stand for $(\lambda X^1(\lambda X^2\ldots(\lambda X^n(\mathbf{AE}^1)\mathbf{E}^2\ldots\mathbf{E}^m)\cdots)$. Finally we will abbreviate a formula $\Pi^\alpha(\lambda X_\alpha\mathbf{.A})$ with $\forall X_\alpha\mathbf{.A}$ in order to re-obtain a more traditional appearance of quantified formulae.

Definition 2.4 (λ-Reduction) Let $\lambda \in \{\beta, \beta\eta, \eta\}$. We say that a well-formed formula $\mathbf{B}$ is obtained from a well-formed formula $\mathbf{A}$ by a **one-step λ-reduction** ($\mathbf{A} \to_\lambda \mathbf{B}$), if it is obtained by applying one of the following rules to a well-formed part (which we call a **λ-redex**) of $\mathbf{A}$.

β-Reduction $(\lambda X\mathbf{.C})\mathbf{D} \to_\beta [\mathbf{D}/X]\mathbf{C}$.
η-Reduction If X is not free in $\mathbf{C}$, then $(\lambda X\mathbf{.C}X) \to_\eta \mathbf{C}$.

As usual we denote the transitive closure of a reduction relation $\to_\lambda$ with $\to_\lambda^*$. These rules induce equivalence relations $=_\beta, =_\eta$, and $=_{\beta\eta}$ on $\mathit{wff}(\Sigma)$, which we call the **λ-equality** relations. A formula that does not contain a λ-redex, and thus cannot be reduced by λ-reduction, is called a **λ-normal form**.

The β-, η-, and $\beta\eta$-reduction relations are terminating and confluent, as the reader can convince himself by looking at the proofs in [Bar80] or [HS86]. Thus for any formula $\mathbf{A}$ there is a sequence of β-reductions $\mathbf{A} \to_\beta^* \mathbf{B}$ such that $\mathbf{B}$ is a **β-normal form**.

3 Semantics

We will now develop the semantical notions needed for characterizing the deductive power of higher-order calculi. The basis of these is the algebraic notion of a Σ-structure, which in turn rests on the following technical notion.

Definition 3.1 (Pre-Σ-Structure) Let $\mathcal{D}_\mathcal{T}$ be a typed collection of sets, @ a family $\{@^{\alpha\beta}: \mathcal{D}_{\alpha\to\beta} \times \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta \mid \alpha, \beta \in \mathcal{T}\}$ of mappings, and let $\mathcal{I}: \Sigma \longrightarrow \mathcal{D}$, then we call the triple $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ a **pre-Σ-structure**. The collection $\mathcal{D}$ is called the **carrier set**, the set $\mathcal{D}_\alpha$ the **universe of type** α, the function @ the **application operator**, and the function $\mathcal{I}$ the **interpretation of constants**. Let $\mathcal{B} = (\mathcal{E}, @^\mathcal{B}, \mathcal{J})$ be a pre-Σ-structures then a typed function $\kappa: \mathcal{D} \longrightarrow \mathcal{E}$ is called a **Σ-homomorphism**, iff $\kappa \circ \mathcal{I} = \mathcal{J}$ and $\kappa(f)@^\mathcal{B}\kappa(g) = \kappa(f@^\mathcal{A}g)$ for all $f \in \mathcal{D}_{\alpha\to\beta}$ and $g \in \mathcal{D}_\alpha$. $\mathcal{A}$ is called **functional**, iff the following statement holds for all $f, g \in \mathcal{D}_{\alpha\to\beta}$: $f = g$, if for all $a \in \mathcal{D}_\alpha$ $f@a = g@a$. Note that functionality only poses a restriction on the function universes.

Note that the typed collections of well-formed formulae are pre-Σ-structures. We will now use the concept of Σ-homomorphisms to give values to well-formed formulae.

Definition 3.2 (Σ-Structure) Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a functional pre-Σ-structure, and Γ a variable context, then we call a function $\varphi: \mathbf{Dom}(\Gamma) \longrightarrow \mathcal{D}$ a **Γ-assignment**, iff $\Gamma(X) = \alpha$ implies $\varphi(X) \in \mathcal{D}_\alpha$. The **homomorphic extension $\mathcal{I}_\varphi$ of φ to** $\mathit{wff}(\Sigma, \Gamma)$ is inductively defined to be a typed partial function $\mathcal{I}_\varphi: \mathit{wff}(\Sigma, \Gamma) \longrightarrow \mathcal{D}$ such that

1. $\mathcal{I}_\varphi(X) = \varphi(X)$, if X is a variable,
2. $\mathcal{I}_\varphi(c) = \mathcal{I}(c)$, if c is a constant,
3. $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B})$,
4. $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{B}_\beta)$ is the function $f \in \mathcal{D}_{\alpha\to\beta}$ such that $f@z := \mathcal{I}_{\varphi,[z/X]}(\mathbf{B})$, if such an element f exists. Otherwise $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{B})$ is undefined. Note that this function is unique, since we have assumed $\mathcal{A}$ to be functional.

We call $\mathcal{I}_\varphi(\mathbf{A}_\alpha) \in \mathcal{D}_\alpha$ the **value** or **denotation of $\mathbf{A}_\alpha$ in $\mathcal{A}$ for** φ. $\mathcal{A}$ is called **comprehension-closed**, iff for each Γ-assignment φ into $\mathcal{A}$ the homomorphic extension $\mathcal{I}_\varphi$ is defined everywhere on $\mathit{wff}(\Sigma, \Gamma)$. In this case we call $\mathcal{A}$ a **Σ-structure**. This closure conditions for the carrier set $\mathcal{D}$ of $\mathcal{A}$ assures that the universes of functions $\mathcal{D}_{\alpha\to\beta}$ are rich enough to contain a value for all $\mathbf{A} \in \mathit{wff}_{\alpha\to\beta}(\Sigma, \Gamma)$.

Definition 3.3 (Σ-Valuation) Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a Σ-structure, then a surjective total function $v: \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ such that

1. $v(\mathcal{I}(\neg)@a) = \mathtt{T}$, iff $v(a) = \mathtt{F}$,
2. $v(\mathcal{I}(\vee)@a@b) = \mathtt{T}$, iff $v(a) = \mathtt{T}$ or $v(b) = \mathtt{T}$,
3. $v(\mathcal{I}(\Pi^\alpha)@f) = \mathtt{T}$, iff $v(f@a) = \mathtt{T}$ for each $a \in \mathcal{D}_\alpha$

is called a Σ-**valuation for** $\mathcal{A}$.

Up to now we have specified a general simply typed λ-calculus, which we will now turn into a higher-order logic $\mathcal{HOL}$ by giving a special meaning to the type o of **truth values**. We will assume that Σ contains at least the **logical** constants $\neg$, $\vee$ and Π^α of types $o \to o$, $o \to o \to o$ and $(\alpha \to o) \to o$. All other constants in $\Sigma_{\mathcal{T}}$ are called **parameters**.

Definition 3.4 (Σ-Model Structure) If $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ is a Σ-structure and υ is a Σ-valuation for $\mathcal{A}$, then we call the quadruple $\mathcal{M} := (\mathcal{D}, @, \mathcal{I}, \upsilon)$ a Σ-**model structure**. Let Γ be a variable context, and φ be a Γ-assignment into $\mathcal{A}$, then we call the function $\mathcal{V}_\varphi = \upsilon \circ \mathcal{I}_\varphi \colon \mathit{wff}_o(\Sigma) \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ the **value function for** $\mathcal{M}$ **and** φ.

We now present two special classes of Σ-model structures, which model the intended understanding of $\mathcal{HOL}$. The class of standard Σ-models is in some way the most natural notion of semantics for $\mathcal{HOL}$, however, with the notion of completeness induced with this semantics there cannot be complete calculi [Göd31], a fact that makes it virtually useless for our purposes. The class of general Σ-models allows complete calculi and, in fact, we will exhibit one later in section 6).

Definition 3.5 (General Σ-Model)
Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon)$ be a Σ-model structure, then $\mathcal{M}$ is called a **general** Σ-**model**, iff $\mathcal{D}_o$ is the set $\{\mathtt{T}, \mathtt{F}\}$ of truth values. Note that with this definition υ must be the identity function on $\mathcal{D}$. We are striving for a general notion of algebraic model, so we only require $\mathcal{M}$ to be comprehension-closed ($\mathcal{A}$ is a Σ-structure) and do not require the carrier sets of $\mathcal{M}$ to contain all possible functions, in this special case we call $\mathcal{M}$ a **standard** Σ-**model**.

4 Tableaux

Now that we have specified the semantics we can turn to the exposition of our tableau calculus. The case of standard tableaux for higher-order logic is a simple extension of first-order tableau methods to the higher-order language. The only significant difference is that $\beta\eta$-equality has to be built in by keeping formulae in normal form. Therefore we will only concern ourselves with free variable tableaux. Here there are two main differences to the first-order case.

- Higher-order unification is undecidable, therefore we cannot simply use it as a sub-procedure that is invoked for closing branches in tableaux. The solution for this problem is treat the unification problem as a constraint and residuate using an explicit $\mathcal{HT}(\mathit{cut})$ tableau rule.
- Naive Skolemization is not sound for higher-order logic. Indeed it necessary to instantiate a predicate variable occurring negatively by a subformula $\mathbf{A}$. Clearly it is not sound to Skolemize $\mathbf{A}$ before instantiation, since P occurs under a negative prefix. But $\mathbf{A}$ is a subformula of a clause $\mathbf{C}$ and must

therefore be Skolemized before refutation. Huet [Hue72] solves this problem by a splitting rule that systematically generates instances for P. Even then Skolemization is not sound for $\mathcal{HOL}$: is possible to prove an instance of the axiom of choice that is known to be independent of $\mathcal{HOL}$ [And73] using naive Skolemization. Dale Miller has investigated this problem in depth [Mil83]. We will use a variant of his technique that the author has developed for higher-order resolution [Koh94].

Skolemization in first-order logic is a syntactic trick that allows to use the "occurs-in-check" in unification to keep track of the dependencies that the sequencing of quantifiers induces on variables. Variables Y that were existentially quantified in the scope of a universal quantifier $\forall X$ may not be free in any formula instantiated for X. These dependencies have to be respected in order to obtain a sound calculus. Thus our tableaux will be built up from a variable context, a variable condition, and a conventional tableau, i.e. a tree labeled with formulae. To make this formal, we have to generalize our notion of variable context by marking the variables with labels $+$, $-$, and 0, in order to distinguish between variables for which we may ($+$, coming from universal variables) or may not ($-$ coming from existential variables) substitute, and those that come from variables that used to be locally bound (0).

Definition 4.1 (Annotated Variable Contexts) Let Γ be a typed partial function on $\mathcal{V}_\mathcal{T}$ that associates with each variable a type α and an **annotation** $\pm \in \{+, 0, -\}$, then we call Γ an **annotated variable context**. As in the case of usual variable contexts we write Γ as a set of **annotated variable declarations** of the form $[X^\pm : \alpha]$, if $\Gamma(X) = (\alpha, \pm)$, and call $\pm$ the **annotation of** X **in** Γ. If the annotation of X is $+$ ($-$) in Γ, then we call it **positive** (**negative**), otherwise (0) **locally bound**, and we indicate this by annotating X with $\pm$ as in $X^\pm$.

Obviously any annotated variable context can be made into a variable context by projection on the first component, so we can use all of the machinery developed so far. Furthermore, we can obtain a variable context Γ^+ (Γ^-, Γ^0) from Γ by restricting Γ to the positive (negative, locally bound) variables.

Definition 4.2 (Variable Condition, $\mathcal{R}_\Gamma$-Substitution) Let Γ be an annotated variable context and $\mathcal{R} \subseteq \mathbf{Dom}(\Gamma^+) \times \mathbf{Dom}(\Gamma^-)$, then $\mathcal{R}$ is called a **variable condition**.

A Σ-substitution σ with $\mathbf{Dom}(\sigma) \subseteq \mathbf{Dom}(\Gamma^+ \cup \mathbf{Dom}(\Delta)$ is called an $\mathcal{R}_\Gamma$-**substitution** assuming Γ, iff $Y \notin \mathbf{Free}(\sigma(X))$ for all $(X, Y) \in \mathcal{R}$. Thus the intuitive meaning of a pair (X^+, Y^-) in a variable condition $\mathcal{R}$ for Γ is that no formula containing Y^- as a free variable can be legally substituted for X^+.

For a variable condition $\mathcal{R}$ and an annotated variable context Δ we define a judgment $\Delta \vdash \overline{\mathcal{R}}(X^+, \mathbf{A})$, called the **associated substitution condition** $\overline{\mathcal{R}}$ to hold on X^+ and $\mathbf{A}$, iff $\mathbf{A} \in \mathit{wff}_{\Gamma^+(X)}(\Sigma, \Delta \cup \Gamma)$, $X \notin \mathbf{Free}(\mathbf{A})$, and no variable $Y \in \mathbf{Free}(\mathbf{A})$ is an $\mathcal{R}$-image of X^+, ($\{X^+\} \times \mathbf{Free}(\mathbf{A}) \cap \mathcal{R} = \emptyset$). Thus we can rephrase the condition on $\mathcal{R}_\Gamma$-substitutions as $\Delta \vdash_\Sigma \overline{\mathcal{R}}(X, \sigma(X))$ for all $X \in \mathbf{Dom}(\sigma)$. For a given variable condition $\mathcal{R}$ for Γ and an $\mathcal{R}_\Gamma$-substitution

$[\mathbf{A}/X^+]$ we will often need the following variable condition for Γ.

$$\mathcal{R}(\mathbf{A}/X) := \{(Z,W) \in \mathcal{R} \mid Z \neq X\} \cup \{(Z,W) \mid Z \in \mathbf{Free}(\mathbf{A}), \mathcal{R}(X,W)\}$$

Definition 4.3 (Labeled Formula) We call a pair $\mathbf{A}^v$ a **labeled formula**, iff $\mathbf{A} \in \mathit{wff}_o(\Sigma,\Gamma)$ and $v \in \{\mathtt{T},\mathtt{F}\}$. A labeled formula $\mathbf{A}^v$ is called **literal**, if $\mathbf{head}(\mathbf{A})$ is a parameter or variable. For the definition of higher-order tableaux we will need a special kind of literals of the form $(\mathbf{A}_\alpha =^? \mathbf{B}_\alpha)^{\mathtt{F}}$. We call these literals **pairs**, since they serve the same purpose as pairs in unification problems, and we often write them as $\mathbf{A} \neq^? \mathbf{B}$ to conserve space. If $\mathbf{A}$ is a positive variable X^+ that is not free in $\mathbf{B}$ and furthermore $\Gamma \vdash_\Sigma \overline{\mathcal{R}}(X,\mathbf{A})$, then $X \neq^? \mathbf{B}$ is called **solved** for Γ and $\mathcal{R}$. $\mathbf{A} \neq^? \mathbf{B}$ is called **flex/flex**, iff the heads of $\mathbf{A}$ and $\mathbf{B}$ are positive variables.

The higher-order unification problem can be reduced to the problem of finding most general formulae of a given type and a given head symbol. Indeed this is essentially all we need for our treatment of higher-order tableaux.

Definition 4.4 (General Binding) Let $\alpha = (\overline{\beta_l} \to \gamma)$, and h be a constant or variable of type $(\overline{\delta_m} \to \gamma)$ in Γ, then $\mathbf{G} := \lambda\overline{X^l_{\alpha_l}}.h\overline{\mathbf{V}^m}$ is called a **general binding of type α and head h**, if $\mathbf{V}^i = H^i\overline{X^l_{\alpha_l}}$. The new variables H^i obtain their types from the context $\mathcal{C} := [H^1{:}\overline{\beta_l} \to \delta^1], \ldots, [H^m{:}\overline{\beta_l} \to \delta^m]$, which is called the **context of variables introduced for $\mathbf{G}$**. It is easy to show that general bindings indeed have the type and head claimed in the name and are most general in the class of all such terms. Moreover they are unique up to the choice of variable names in $\mathcal{C}$.

General bindings, where the head is a bound variable $X^j_{\beta_j}$ are called **projection bindings** (we write them as $\mathcal{G}^j_\alpha(\Sigma,\Gamma,\mathcal{C})$) and **imitation bindings** (written $\mathcal{G}^h_\alpha(\Sigma,\Gamma,\mathcal{C})$) else. Since we need both imitation and projection bindings for higher-order unification, we collect them in the set of **approximating bindings for h and α** $\mathcal{A}^h_\alpha(\Sigma,\Gamma,\mathcal{C}) := \{\mathcal{G}^h_\alpha(\Sigma,\Gamma,\mathcal{C})\} \cup \{\mathcal{G}^j_\alpha(\Sigma,\Gamma,\mathcal{C}) \mid j \leq l\}$

Definition 4.5 (Higher-Order Tableau) Let Γ be an annotated variable context and $\mathcal{R}$ a variable condition for Γ. We will call a triple $\langle\Gamma{:}\mathcal{R}\rangle.\mathcal{T}$, where $\mathcal{T}$ is a tree labeled with labeled formulae a **higher-order tableau**, iff it can be constructed by the following **tableau construction rules**.

These rules come in four categories, the structural rules

$$\frac{(\mathbf{A} \vee \mathbf{B})^{\mathtt{T}}}{\mathbf{A}^{\mathtt{T}} \mid \mathbf{B}^{\mathtt{T}}}\mathcal{HT}(\wedge) \qquad \frac{(\mathbf{A} \vee \mathbf{B})^{\mathtt{F}}}{\begin{array}{c}\mathbf{A}^{\mathtt{F}}\\ \mathbf{B}^{\mathtt{F}}\end{array}}\mathcal{HT}(\vee) \qquad \frac{(\neg\mathbf{A})^{\mathtt{T}}}{\mathbf{A}^{\mathtt{F}}}\mathcal{HT}(\neg^{\mathtt{F}}) \qquad \frac{(\neg\mathbf{A})^{\mathtt{F}}}{\mathbf{A}^{\mathtt{T}}}\mathcal{HT}(\neg^{\mathtt{T}})$$

for the propositional connectives and the structural rules for the Π^α quantifier

$$\frac{(\Pi^\alpha\mathbf{A})^{\mathtt{T}}}{(\mathbf{A}X^+_\alpha)^{\mathtt{T}}}\mathcal{HT}(all) \qquad \frac{(\Pi^\alpha\mathbf{A})^{\mathtt{F}}}{(\mathbf{A}X^-_\alpha)^{\mathtt{F}}}\mathcal{HT}(ex)$$

These recursively build up the tableau tree by decomposing the logical structure of proper labeled formulae and adding new nodes and branches. Now we still have to specify the action of these rules on the context and variable condition. Let $\langle \Gamma' : \mathcal{R}' \rangle . \mathcal{T}'$ is obtained from $\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$ by these rules, then the first set of rules leave the Γ and $\mathcal{R}$ unchanged. For $\mathcal{HT}(all)$ we set $\Gamma' = \Gamma, [X^+ : \alpha]$ and $\mathcal{R}' = \mathcal{R}$ and for $\mathcal{HT}(ex)$ we have $\Gamma' = \Gamma, [X^- : \alpha]$ and $\mathcal{R}' = \mathcal{R} \cup (\mathbf{Free}(\mathbf{A}) \times \{X^-\})$.

Since higher-order unification is undecidable, we need an explicit rule for cutting complementary formulae in branches of $\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$: If $\mathbf{A}^v$ and $\mathbf{B}^w$ occur in a branch $\mathcal{B}$ of $\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$ where $v \neq w$, then the rule $\mathcal{HT}(cut)$ introduces the constraint $\mathbf{A} \neq^? \mathbf{B}$ as a leaf of $\mathcal{B}$. Γ and Δ are left unchanged by $\mathcal{HT}(cut)$. This new constraint pair has to be processed before the branch can be closed in order to conserve soundness.

Furthermore there are the **tableau substitution rules** $\mathcal{HT}(subst)$ and $\mathcal{HT}(prim)$ that allow to instantiate the whole tableau $\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$ with an elementary substitution $[\mathbf{A}/X^+]$, iff

some path Θ of $\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$ ends in a solved pair $X^+ \neq^? \mathbf{A}$ ($\mathcal{HT}(subst)$)
$\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$ contains a labeled formula $X\overline{\mathbf{U}^n}^v$, such that $\mathbf{A} \in \mathcal{A}^k_{\Gamma(X)}(\Sigma, \Gamma, \mathcal{C})$ is a general binding that approximates some logical constant $k \in \{\vee, \neg, \Pi^\beta \mid \beta \in \mathcal{T}\}$ ($\mathcal{HT}(prim)$).

In both rules the resulting tableau is of the form $\langle \Gamma : \mathcal{R}[\mathbf{A}/X] \rangle . [\mathbf{A}/X]\mathcal{T}$. The action of the tableau substitution rules on the variable conditions propagates the variable dependencies and thus ensures soundness.

The last group of rules solve the unification constraints

$$\frac{(\lambda X_\alpha . \mathbf{A}) \neq^? (\lambda Y_\alpha . \mathbf{B}) \quad Z \notin \mathbf{Dom}(\Gamma)}{[Z/X]\mathbf{A} \neq^? [Z/Y]\mathbf{B}} \mathcal{HT}(\alpha)$$

$$\frac{(\lambda X_\alpha . \mathbf{A}) \neq^? \mathbf{B} \quad Z \notin \mathbf{Dom}(\Gamma)}{[Z/X]\mathbf{A} \neq^? (\mathbf{B}Z)} \mathcal{HT}(\eta)$$

$$\frac{h\overline{\mathbf{U}^n} \neq^? h\overline{\mathbf{V}^n} \quad h \in \Sigma \cup \mathbf{Dom}(\Gamma^0) \cup \mathbf{Dom}(\Gamma^-)}{\mathbf{U}^1 \neq^? \mathbf{V}^1 \;\Big|\; \ldots \;\Big|\; \mathbf{U}^n \neq^? \mathbf{V}^n} \mathcal{HT}(dec)$$

$$\frac{F\overline{\mathbf{U}} \neq^? h\overline{\mathbf{V}} \quad \Gamma^+(F) = \alpha \quad \Gamma \vdash_\Sigma \overline{\mathcal{R}}(F^+, \mathbf{G})}{F \neq^? \mathbf{G} \vee F\overline{\mathbf{U}} \neq^? h\overline{\mathbf{V}}} \mathcal{HT}(flex/rig)$$

Here $\mathbf{G} \in \mathcal{A}^h(\Sigma, \Gamma, \mathcal{C})$ be a general binding of type α that approximates the head h. In the first two rules $\Gamma' = \Gamma, [Z^0 : \alpha]$ and $\mathcal{R}' = \mathcal{R}[Z/X]$, whereas in $\mathcal{HT}(flex/rig)$ we have $\Gamma' = \Gamma \cup \mathcal{C}$ and $\mathcal{R}' = \mathcal{R}$. All of these rules are used with

the understanding that all formulae are reduced w-normal form after each rule application.

Note that this last set of rules directly corresponds to the rules of higher-order pre-unification as they can be found in [Koh94], which generalize Huet's pre-unification transformations (see for instance [Sny91]) for variable conditions. With these rules we use the tableau mechanism to construct Huet's unification tree [Hue76].

We call a branch Θ in a higher-order tableau $\mathcal{T}$ **closed**, iff Θ ends in a flex/flex pair or a trivial pair $\mathbf{A} \neq^? \mathbf{A}$. Note that the $\mathcal{HT}(subst)$ rule immediately closes the branch Θ that ends in a solved pair. A tableau $\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$ is called **closed**, iff each branch of $\langle \Gamma : \mathcal{R} \rangle . \mathcal{T}$ is closed. Note that closed branches need not stay closed, since the heads of flex/flex pairs can be instantiated making them rigid. In this case unification has to resume on the particular branch. For a proposition $\mathbf{A} \in wff_o(\Sigma, \Gamma)$ we write $\vdash_{\mathcal{HT}} \mathbf{A}^v$, if there is a variable context $\Delta \supseteq \Gamma$ and a closed higher-order tableau $\langle \Delta : \mathcal{Q} \rangle . \mathcal{T}$ with $\mathbf{A}^v$ at the root. In this case we call $\langle \Delta : \mathcal{Q} \rangle . \mathcal{T}$ a closed tableau for $\mathbf{A}^v$. We call $\mathbf{B}$ a $\mathcal{HT}$**-theorem** ($\vdash_{\mathcal{HT}} \mathbf{B}$), iff $\vdash_{\mathcal{HT}} \mathbf{A}^{\mathtt{F}}$.

We will now exemplify $\mathcal{HT}$ by proving a variant of Cantor's theorem.

Example 4.6 (Cantor's Theorem) We show that there cannot be a surjective mapping from the natural numbers to infinite sequences of natural numbers. In order to simplify the problem we take the type ι to be the set of natural numbers, thus Cantor's theorem has the form

$$\neg \exists F_{\iota \to \iota \to \iota} . \forall G_{\iota \to \iota} . \exists J_\iota . FJ = G$$

To be able to prove the theorem we need a formalization of the natural numbers (instead of the Peano Axioms we add the fact that $\forall X_\iota . \neg X = sX$ that the successor function has no fixed point) and an extensionality axiom.

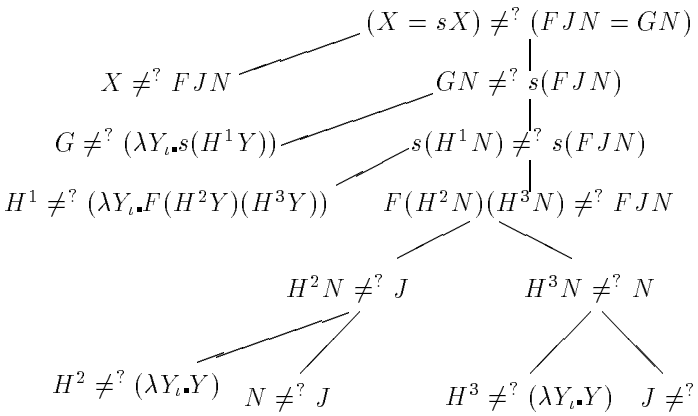
$$\begin{array}{c}
(\neg \exists F_{\iota \to \iota \to \iota} . \forall G_{\iota \to \iota} . (\exists J_\iota . FJ = G))^{\mathtt{F}} \\
(\forall H_{\iota \to \iota} . \forall K_{\iota \to \iota} . H = K \Rightarrow . \forall N_\iota . HN = KN)^{\mathtt{T}} \\
(\forall X_\iota . \neg X = sX)^{\mathtt{T}} \\
\forall G_{\iota \to \iota} . (\exists J_\iota . FJ = G)^{\mathtt{T}} \\
(\exists J_\iota . FJ = G)^{\mathtt{T}} \\
FJ = G^{\mathtt{T}} \\
(H^+ = K^+ \Rightarrow . \forall N_\iota . H^+ N = K^+ N)^{\mathtt{T}} \\
\begin{array}{c|c|c}
H = K^{\mathtt{F}} & & HN = KN^{\mathtt{T}} \\
H = K \neq^? FJ = G & & FJN = GN^{\mathtt{T}} \\
H \neq^? FJ & K \neq^? G & (X = sX)^{\mathtt{F}}
\end{array}
\end{array}$$

At this stage the variable context is

$$\Gamma = [F^- : \iota \to \iota \to \iota], [G^+ : \iota \to \iota], [J^- : \iota], [H : \iota \to \iota], [K : \iota \to \iota]$$

and the variable condition $\mathcal{R}$ consists of the pair (G^+, J^-). Both paths in the left subtableau can now be closed by instantiating the tableau with the substitution

$[FJ/H], [G/K]$, which is obviously a $\mathcal{R}_\Gamma$-substitution. The last two labeled formulae on the rightmost path are complementary, so they can be cut by $\mathcal{HT}(cut)$ yielding a subtableau of the following form.



Note that this tableau tree exactly corresponds to a sequence of higher-order unification transformations to the initial pair. Note that at the end of the leftmost branch, the variable condition has become $\{(H^i, J^-) \mid i = 1, 2, 3\}$, which bars the imitation binding $H^2 \neq^? (\lambda Y_\iota . J^-)$, which would have been possible without, but which would have violated the variable condition imposed by the quantifier prefix of the theorem.

Even though we have not needed it for the example above (it was sufficient to treat equality as an arbitrary binary relation). $\mathcal{HT}$ can be used for equational reasoning in higher-order logic, since $\mathcal{HOL}$ is powerful enough to define equality.

Definition 4.7 (Leibniz' Formulation for Equality) We define the **Leibniz formula** for equality by

$$\mathbf{Q}^\alpha := (\lambda X_\alpha Y_\alpha . \forall P_{\alpha \to o} . PX \Rightarrow PY)$$

With this definition the formula $(\mathbf{A} = \mathbf{B}) = \mathbf{Q}^\alpha \mathbf{A} \mathbf{B}$ β-reduces to $\forall P_{\alpha \to o} . (P\mathbf{A}) \Rightarrow (P\mathbf{B})$, which can be read as: formulae $\mathbf{A}$ and $\mathbf{B}$ are not equal, iff there exists a discerning property P. In other words, $\mathbf{A}$ and $\mathbf{B}$ are equal, if they are indiscernible. We semantically justify this definition by 4.8.

Lemma 4.8 *Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, v)$ be a Σ-model structure, and let $\mathbf{Q}^\alpha$ be defined as in 4.7, then $\mathcal{V}_\varphi(\mathbf{Q}^\alpha \mathbf{A} \mathbf{B}) = \mathtt{T}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$.*

5 Soundness and Completeness

We now proceed to give a definition of validity for labeled formulae that are the basis of the soundness considerations. This notion of validity takes positive variables implicitly, universally quantified, and uses the notion of $\mathcal{R}_\Gamma$-correspondences as a semantic counterpart of variable conditions that specify the dependencies of variables recorded during the clause normal form transformation.

Definition 5.1 (Tableau Satisfiability) Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, v)$ be a model structure, Γ an annotated variable context, and $\mathcal{R}$ a variable condition for Γ. If $Y^- \in \mathbf{Dom}(\Gamma^-)$, $\{X_1^+, \ldots, X_n^+\} = \mathcal{R}^{-1}(Y)$, and $\Gamma(X_i) = \alpha_i$, then a total function $f_Y : \mathcal{D}_{\alpha_1} \times \cdots \times \mathcal{D}_{\alpha_n} \longrightarrow \mathcal{D}_{\Gamma(Y)}$ is called an $\mathcal{R}_\Gamma$-**function** for Y in $\mathcal{M}$. We call a complete set $\{f_Y \mid Y \in \mathbf{Dom}(\Gamma^-)\}$ of $\mathcal{R}_\Gamma$-functions an $\mathcal{R}_\Gamma$-**correspondence** for $\mathcal{M}$. Note that in the case, where $n = 0$, the variable $Y^- \in \mathbf{Dom}(\Gamma^-)$ is not in $\mathbf{Im}(\mathcal{R})$, but we still need an $f_Y \in \mathcal{D}_{\Gamma(Y)}$ in $\mathcal{F}$.

If $\mathcal{F}$ is an $\mathcal{R}_\Gamma$-correspondence for $\mathcal{M}$ and φ is a Γ-assignment into $\mathcal{M}$, then we define the Γ-assignment $\varphi_\mathcal{F}$ by

$$\varphi_\mathcal{F}(Y) := \begin{cases} \varphi(Y), & \text{if } Y \notin \mathbf{Dom}(\Gamma^-) \\ f_Y @ \varphi(X_1) @ \cdots @ \varphi(X_n), & \text{if } Y \in \mathbf{Dom}(\Gamma^-) \text{ and} \\ & \{X_1, \ldots, X_n\} = \mathcal{R}^{-1}(Y) \end{cases}$$

We say that a labeled formula $\mathbf{A}^v$ is **satisfiable in a model structure** $\mathcal{M} = (\mathcal{D}, \mathcal{I}, @, v)$, iff there is an $\mathcal{R}_\Gamma$-correspondence $\mathcal{F}$ for $\mathcal{M}$, such that for all Γ-assignments φ we have $v(\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{A})) = v$, analogously for a pair $\mathbf{A} \neq^? \mathbf{B}$, iff $\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{A}) \neq \mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{B})$. We call a tableau $\mathcal{T}$ satisfiable, iff all of its formulae are.

A consequence of this definition, which regards positive variables as implicitly, universally quantified, is that the names of these do not carry any semantic meaning. Thus we consider the declaration $\langle \Gamma : \mathcal{R} \rangle.$ in a tableau as a binder for all variables in $\mathbf{Dom}(\Gamma)$, and we keep α-conversion for tableaux implicit, renaming them in the tableau substitution rules whenever needed to prevent variable capture.

For sentences the new notion of validity coincides with the classical notion. Furthermore, if Γ is an annotated variable context, $\mathcal{R}$ is a variable condition for Γ and $\mathbf{C} =_{\beta\eta} \mathbf{D}$, then $\mathcal{M} \models \mathbf{C}^v$, iff $\mathcal{M} \models \mathbf{D}^v$ for any model structure $\mathcal{M}$.

Lemma 5.2 *Let $\langle \Gamma' : \mathcal{Q} \rangle.\mathcal{T}'$ be a tableau obtained from a tableau $\langle \Gamma : \mathcal{R} \rangle.\mathcal{T}$ by a tableau construction rule and $\mathcal{M}$ be a model structure, then $\mathcal{M} \models \langle \Gamma : \mathcal{R} \rangle.\mathcal{T}$, iff $\mathcal{M} \models \langle \Delta : \mathcal{Q} \rangle.\mathcal{T}'$.*

Proof: We only present the proof for the case where $\langle \Gamma : \mathcal{R} \rangle.\mathcal{T} \vdash_{\mathcal{HT}(ex)} \langle \Gamma' : \mathcal{R}' \rangle.\mathcal{T}'$, since all others are unproblematic, because the variable condition is not altered by the transformation. In this case $\Gamma' = \Gamma, [X^- : \alpha]$ and $\mathcal{R}' := \mathcal{R} \cup (\mathbf{Free}(\mathbf{A}) \times \{X^-\})$. If $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, v) \models \langle \Gamma : \mathcal{R} \rangle.\mathcal{T}$, then there is an $\mathcal{R}_\Gamma$-correspondence $\mathcal{F}$ for $\mathcal{M}$ such that for all Γ-assignments φ there is a labeled proposition or pair in $\langle \Gamma : \mathcal{R} \rangle.\mathcal{T}$ that is satisfied by φ in $\mathcal{M}$. Let $\Pi^\alpha \mathbf{A}^\mathtt{F}$ be the formula in $\langle \Gamma : \mathcal{R} \rangle.\mathcal{T}$ that the construction rule acts on. Since $\mathcal{M} \models \mathbf{A}^\mathtt{F}$ we have $v(\mathcal{I}_{\varphi_\mathcal{F}}(\Pi^\alpha \mathbf{A})) = \mathtt{F}$, thus there is an $a \in \mathcal{D}_\alpha$ such that $\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{A}) @ a = \mathtt{F}$, and finally $\mathcal{I}_\psi(\mathbf{A} X^-) = \mathtt{F}$, where $\psi := \varphi_\mathcal{F}, [a/X^-]$. Since for any ψ' that agrees with ψ on $\mathbf{Free}(\mathbf{A}) = \{X_1, \ldots, X_n\}$, we have $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_{\psi'}(\mathbf{A})$, this a only depends on $\psi\big|_{\mathbf{Free}(\mathbf{A})} = \varphi_\mathcal{F}\big|_{\mathbf{Free}(\mathbf{A})}$. Since we have made no assumptions on φ, we can construct a function F_X that maps each tuple $(\psi(X_1), \ldots, \psi(X_n))$, where φ is a Γ-assignment to the set of elements $a \in \mathcal{D}_\alpha$, such that $v(\mathcal{I}_{\varphi_\mathcal{F}}(\mathbf{A}) @ a) = \mathtt{F}$. These sets are always

nonempty, so by the axiom of choice (not the one on the object level, which we have not included into $\mathcal{HOL}$ but the one on the meta level we are reasoning on) there is a total function f_X mapping each tuple $(\psi(X_1), \ldots, \psi(X_n))$ to some element of $F_X(\psi(X_1), \ldots, \psi(X_n))$. Thus $\mathcal{F}' := \mathcal{F} * f_X$ to an $\mathcal{R}'$-correspondence. Furthermore, we have $\psi = \varphi_{\mathcal{F}}, [a/X^-] = \varphi_{\mathcal{F}'}$, so $\mathcal{I}_{\varphi_{\mathcal{F}'}}(\mathbf{A}X^-) = \mathtt{F}$ for all Γ-assignments φ into $\mathcal{M}$ and thus $\mathcal{M} \models \mathcal{D}$ by definition.

For the converse direction let $\mathcal{M} \models \mathcal{D}$. We assume the existence of an $\mathcal{R}'_{\Gamma,[X^-:\alpha]}$-correspondence $\mathcal{F}'$ for $\mathcal{M}$ such that for all $\Gamma, [X^-:\alpha]$-assignments φ we have $v(\mathcal{I}_{\varphi_{\mathcal{F}'}}(\mathbf{A}X^-)) = \mathtt{F}$. Since $X^- \in \mathbf{Dom}(\Gamma^-, [X^-:\alpha])$ there must be a function $f_X : \mathcal{D}_{\Gamma(X_1)} \times \cdots \times \mathcal{D}_{\Gamma(X_n)} \longrightarrow \mathcal{D}_{\Gamma(X^-)}$ in $\mathcal{F}'$. Let $\mathcal{F} := \mathcal{F}' \setminus \{f_X\}$, then $\mathcal{F}$ is an $\mathcal{R}$-correspondence and $\varphi_{\mathcal{F}'} = \varphi_{\mathcal{F}}, [f_X @ \varphi(X_1) @ \cdots @ \varphi(X_n)/X^-]$. Thus $\mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{A}X) = \mathcal{I}_{\varphi_{\mathcal{F}}}(\mathbf{A}) @ (f_X @ \varphi(X_1) @ \cdots @ \varphi(X_n))$, and therefore $v(\mathcal{I}_{\varphi_{\mathcal{F}}}(\Pi^\alpha \mathbf{A})) = \mathtt{F}$, since v is a Σ-valuation. Since we have taken φ to be an arbitrary Γ-valuation, we have $\mathcal{M} \models \langle \Gamma : \mathcal{R} \rangle . \mathcal{I}$. □

We now introduce an important tool for proving completeness results in higher-order logic. The importance of model existence theorems lies in the fact that they abstract over the model theoretic part of various completeness proofs. With the help of a model existence theorem the completeness proof for a given logical system $\mathcal{C}$ is reduced to the (purely proof-theoretic) demonstration that the class of $\mathcal{C}$-consistent sets is an abstract consistency class. This proof technique (called "unifying principle" there) was first introduced by Smullyan in [Smu63, Smu68], based on work by Hintikka and Beth and later generalized to higher-order logic by Andrews in [And71]. Since there is no simple Herbrand theorem in higher-order logic, Andrews "unifying principle for type theory" from [And71] has become the standard method for completeness proofs in higher-order logic.

Definition 5.3 (Abstract Consistency Class) Let Γ be a negative annotated variable context, and let $\nabla_\Sigma(\Gamma)$ be a class of sets of propositions, then $\nabla_\Sigma := \{\nabla_\Sigma(\Gamma)\}$ is called an **abstract consistency class**, iff each $\nabla_\Sigma(\Gamma)$ is closed under subsets, and for all sets $\Phi \in \nabla_\Sigma(\Gamma)$ the following conditions hold:

1. If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.
2. If $\mathbf{A} \in \Phi$ and if $\mathbf{B}$ is the $\beta\eta$-normal form of $\mathbf{A}$, then $\mathbf{B} * \Phi \in \nabla_\Sigma(\Gamma)$.
3. If $\neg\neg\mathbf{A} \in \Phi$, then $\mathbf{A} * \Phi \in \nabla_\Sigma(\Gamma)$.
4. If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \nabla_\Sigma(\Gamma)$ or $\Phi * \mathbf{B} \in \nabla_\Sigma(\Gamma)$.
5. If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \in \nabla_\Sigma(\Gamma)$.
6. If $\Pi^\alpha \mathbf{A} \in \Phi$, then $\Phi * \mathbf{AB} \in \nabla_\Sigma(\Gamma)$ for each $\mathbf{B} \in \mathit{wff}_\alpha(\Sigma)$.
7. If $\neg\Pi^\alpha \mathbf{A} \in \Phi$, then $\Phi * \neg(\mathbf{A}X) \in \nabla_\Sigma(\Gamma, [X^-:\alpha])$.

We call an abstract consistency class **saturated**, iff for all $\Phi \in \nabla_\Sigma(\Gamma)$ and all atomic propositions $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ we have $\Phi * \mathbf{A} \in \nabla_\Sigma(\Gamma)$ or $\Phi * \neg\mathbf{A} \in \nabla_\Sigma(\Gamma)$.

The significance of abstract consistency classes lies in the following theorem, which we cite from [Koh94].

Theorem 5.4 (Existence of Σ-Model Structures)
Let $H \in \nabla_\Sigma(\Gamma)$ and ∇_Σ be a saturated abstract consistency class, then there is a Σ-model structure $\mathcal{M}$ with $\mathcal{M} \models H$.

Theorem 5.5 (Tableau Lifting) *Let $\Phi \subseteq \mathit{wff}_o(\Sigma,\Gamma)$ and σ a substitution, then Φ has closed $\mathcal{HT}$-tableau provided $\theta(\Phi)$ has one.*

Proof sketch: Let $\langle \Gamma : \mathcal{R}\rangle\!.\mathcal{T}_\theta$ be a closed tableau for $\theta(\Phi)$, the claim is proven by an induction on the construction of $\langle \Gamma : \mathcal{R}\rangle\!.\mathcal{T}_\theta$ constructing a tableau $\langle \Delta : \mathcal{Q}\rangle\!.\mathcal{T}$ for Φ that is isomorphic to $\langle \Gamma : \mathcal{R}\rangle\!.\mathcal{T}$.

For this task it is crucial to maintain a tight correspondence $\omega : \mathcal{T} \longrightarrow \mathcal{T}_\theta$ between $\langle \Gamma : \mathcal{R}\rangle\!.\mathcal{T}_\theta$ and $\langle \Delta : \mathcal{Q}\rangle\!.\mathcal{T}$ that respects labels and is compatible with θ, i.e. for any node $\mathcal{N}$ in $\mathcal{T}$ with labeled formula $\mathbf{A}^v$ we have $\omega_{\mathcal{N}}(\mathbf{A}) = \theta(\mathbf{A})$. The main difficulty with lifting properties in higher-order logic is the fact that due to the existence of predicate variables at the head of formulae, the propositional structure of formulae can change during instantiation. For instance if $\theta(F) = \lambda X_\alpha\!.GX \vee p$, and $\mathbf{A}^{\mathsf{T}} = Fa^{\mathsf{T}}$ is the formula of $\mathcal{N}$, then $\mathcal{HT}(\vee)$ is applicable in $\mathcal{T}_\theta$ but not in the fragment of $\mathcal{T}$ already constructed. The solution of this problem is to apply $\mathcal{HT}(\mathit{prim})$ with a suitable general binding $\mathcal{G}^{\vee}_{\alpha\to o} = \lambda X_\alpha\!.(H^1X)\vee(H^2X)$ and obtain a node $\mathcal{N}'$ with formula $(H^1a \vee H^2a)^{\mathsf{T}}$, to which $\mathcal{HT}(\vee)$ can be applied. Since $\mathcal{G}^{\vee}_{\alpha\to o}$ is more general than $\theta(F)$ there is a substitution ρ, such that $\theta(F) = \rho(\mathcal{G}^{\vee}_{\alpha\to o})$, therefore $\omega_{\mathcal{N}'}((H^1a \vee H^2a)^{\mathsf{T}}) = \theta'((H^1a \vee H^2a)^{\mathsf{T}})$ where $\theta' = \theta \cup \rho$. □

Theorem 5.6 (Completeness) *$\mathcal{HT}$ is refutation complete, i.e. if Φ is a valid set of sentences, then there is a closed higher-order tableau for $\langle \emptyset : \emptyset\rangle\!.\Phi^{\mathsf{F}}$.*

Proof: Completeness of $\mathcal{HT}$ can be proven using the model existence theorem by verifying that the class ∇_Σ defined by

$$\nabla_\Sigma(\Gamma) := \{\Phi \subseteq \mathit{wff}_o(\Sigma,\Gamma) \mid \nvdash_{\mathcal{HT}} \Phi^{\mathsf{T}}\}$$

is a saturated abstract consistency class. This can be achieved with the usual techniques. We have treated the only significant difference to the proofs given in e.g. [Fit90] in the tableau lifting theorem above. □

In contrast to most completeness theorems for higher-order machine-oriented refutation calculi [Hue72, Mil83, And89] in the literature, which only state completeness with respect to a certain Hilbert calculus, we have given a model-theoretic completeness theorem for $\mathcal{HT}$. Since the other calculi are also sound and complete with respect to our semantics of Σ-model structures (see [Koh94]), the theorem above immediately gives rise to a relative completeness theorem in that style.

6 Extensional Tableaux

In the previous section we have proven completeness of $\mathcal{HT}$ with respect to Σ-model-structures, which characterizes traditional higher-order refutation calculi, but is not a very intuitive notion of semantics. Indeed $\mathcal{HT}$ is not complete with respect to general models. Consider for instance the signature formulae $\mathbf{A} := (cb)$,

$\mathbf{B} := c(\neg\neg b)$, and $\mathbf{C} := \neg\mathbf{A} \vee \mathbf{B}$ for the signature $\Sigma := \{[c\!:\!o \rightarrow o], [b\!:\!o]\}$. As we see from the following tableau

$$\begin{array}{c}(\neg(cb) \vee c(\neg\neg b))^{\mathtt{F}}\\ \neg(cb)^{\mathtt{F}}\\ c(\neg\neg b)^{\mathtt{F}}\\ cb^{\mathtt{T}}\\ cb \neq^{?} c(\neg\neg b)\\ b \neq^{?} \neg\neg b\end{array}$$

which cannot be closed, since no $\mathcal{HT}$ construction rule applies to the last pair. This lack of completeness is unfortunate, since class of general Σ-models is the most intuitive one that admits complete calculi. In particular, our mathematical intuition which conforms to general Σ-models would make us believe that $\mathbf{C}$ should be refutable, because $\neg\neg b$ is provably equivalent to b. This example shows us that in extensional calculi we have to deal with propositions that appear in the arguments of parameters. The simplest approach to build a calculus that can refute $\mathbf{C}$ is to add the equational theory $b = \neg\neg b$ to higher-order unification. Even though this approach is intuitive, it does not solve the general problem of incorporating extensionality into higher-order tableaux. In fact, we can generalize the formula $\mathbf{C} := (cb) \vee \neg c(\neg\neg b)$ to $\mathbf{C}' := (c\mathbf{A}) \vee \neg(c\mathbf{B})$, where $\mathbf{A}$ and $\mathbf{B}$ are arbitrary propositions. Now $\mathbf{C}'$ is valid in the class of general Σ-models, iff $\mathbf{A} \Leftrightarrow \mathbf{B}$ is valid. So the approach of enhancing the unification would require augmenting the unification procedure by the theory of logical equivalence, which would enable the unification procedure to prove any theorem by unifying it with $\top_o$.

The semantic problem with completeness behind this example is maybe best illustrated by the following lemma.

Lemma 6.1 *The axiom* $\mathbf{Ext}^o = \forall F_o.\forall G_o.(F \Leftrightarrow G) \Leftrightarrow F = G$ *is valid in the class of general* Σ*-models, but not in that of* Σ*-model structures.*

In particular the failure of equivalence to entail equality means that substitutivity of equivalence does not hold for Σ-model structures (remember the Leibniz definition of equality) and thus "buried occurences" propositions cannot be substituted for equivalent ones. Thus the remedy for the incompleteness of $\mathcal{HT}$ is a tableau construction rule that relates equality in unification constraints with equivalence.

Definition 6.2 (Extensional Tableau Calculus $\mathcal{HTE}$)
The tableau construction rules for $\mathcal{HTE}$ consist of those for $\mathcal{HT}$ together with the following one that takes care of unification constraints of type o

$$\frac{\mathbf{A}_o \neq^{?} \mathbf{B}_o}{\begin{array}{c|c}\mathbf{A}^{\mathtt{F}} & \mathbf{A}^{\mathtt{T}}\\ \mathbf{B}^{\mathtt{T}} & \mathbf{B}^{\mathtt{F}}\end{array}}\,\mathcal{HTE}(ext)$$

In particular we can continue our example above with the subtableau

$$\begin{array}{c|c} b \neq^? \neg\neg b & \\ \neg\neg b^{\mathtt{F}} & \neg\neg b^{\mathtt{T}} \\ b^{\mathtt{T}} & b^{\mathtt{F}} \\ b^{\mathtt{F}} & b^{\mathtt{T}} \\ b \neq^? b & b \neq^? b \end{array}$$

Note that $\mathcal{HTE}$ completely blurs the distinction between propositional reasoning by decomposition of formulae and substitution construction by unification constraint solving. In particular we can indeed prove a sentence $\mathbf{A}$ by unifying it with a simple tautology, such as $\mathbf{B} \vee \neg\mathbf{A}$.

Theorem 6.3 (Soundness and Completeness) *$\mathcal{HTE}$ is sound and refutation complete, i.e. a proposition $\mathbf{A} \in wff_o(\Sigma, \Gamma)$ is valid in all general Σ-models, iff there is a closed higher-order extensional tableau for $\langle \Gamma : \emptyset \rangle \mathbf{.A}^{\mathtt{F}}$.*

Proof sketch: Soundness of $\mathcal{HTE}(ext)$ is immediate: If $\mathbf{A}$ and $\mathbf{B}$ have different truth values, then one must be $\mathtt{T}$, while the other must be $\mathtt{F}$. Completeness is a consequence of the model existence theorem for general Σ models below, which is proven in [Koh94]. □

Theorem 6.4 (Model Existence for General Σ-Models) *If ∇_Σ is a saturated abstract consistency class (cf. 5.3), such that the following additional conditions hold for all sets $\Phi \in \nabla_\Sigma(\Gamma)$:*

8. If $\neg(\mathbf{A} =^{\alpha\to\beta} \mathbf{B}) \in \Phi$, then $\Phi * (\neg\mathbf{A}X = \mathbf{B}X) \in \nabla_\Sigma(\Gamma, [X^- : \alpha])$.
9. If $\{\mathbf{A}, \mathbf{B}\} \subseteq \Phi$, then $\Phi * (\mathbf{A} = \mathbf{B}) \in \nabla_\Sigma(\Gamma)$.
10. If $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \Phi$, then $\Phi * (\mathbf{A} = \mathbf{B}) \in \nabla_\Sigma(\Gamma)$.

then H has a countable general Σ-model.

7 Conclusion

We have presented two tableau calculi for higher-order logic and proven them sound and complete. The first variant $\mathcal{HT}$ is complete relative to a non-standard semantics that also characterizes completeness of known higher-order refutation calculi like Huet's constrained resolution or Andrews' higher-order matings method. All of these calculi are not complete in the presence of buried occurences of propositions, since substitutivity of equivalence is not admissible in them and thus they are not complete with respect to Henkin's more intuitive general model semantics. We have remedied this situation by introducing a special rule that trades propositional unification pairs for equivalences and gives a complete calculus. To the best of the authors knowledge $\mathcal{HTE}$ is the only machine-oriented calculus that is complete with respect to general Σ-models.

References

[And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 3(36):414–432, 1971.

[And73] Peter B. Andrews, 1973. letter to Roger Hindley dated January 22, 1973.

[And89] Peter B. Andrews. On connections and higher order logic. *Journal of Automated Reasoning*, 5:257–291, 1989.

[Bar80] Hendrik P. Barendregt. *The Lambda-Calculus: Its Syntax and Semantics*. North-Holland, 1980.

[Bet55] E. W. Beth. Semantic entailmentand formla derivability. *Medelingen von de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde*, 18(13):309–342, 1955.

[Bet69] E. W. Beth. Semantic entailmentand formla derivability. In J. Hintikka, editor, *The Philosophy of Mathematics*, pages 9–49. Oxford University Press, 1969. First published in [Bet55].

[Fit90] Melvin Fitting. *First-Order Logic and Automated Threorem Proving*. Springer Verlag, 1990.

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte der Mathematischen Physik*, 38:173–198, 1931. English Version in [vH67].

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

[Hin55] K. J. J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:7–55, 1955.

[HS86] J. Hindely and J. Seldin. *Introduction to Combinators and Lambda Calculus*. Cambridge University Press, 1986.

[Hue72] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic*. PhD thesis, Case Western Reserve University, 1972.

[Hue76] Gérard P. Huet. *Résolution d'Équations dans des Langages d'ordre 1,2,...,w*. Thèse d'État, Université de Paris VII, 1976.

[JP76] D. C. Jensen and T. Pietrzykowski. Mechanizing ω-order type theory through unification. *Theoretical Computer Science*, 3:123–171, 1976.

[Koh94] Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. PhD thesis, Universität des Saarlandes, 1994.

[Mil83] Dale Miller. *Proofs in Higher-Order Logic*. PhD thesis, Carnegie-Mellon University, 1983.

[Pfe87] F. Pfenning. *Proof Transformations in Higher-Order Logic*. PhD thesis, Carnegie-Mellon University, Pittsburgh Pa., 1987.

[Pra60] Dag Prawitz. An improved proof procedure. *Theoria*, 26:102–139, 1960.

[Ree87] S. Reeves. Semantic tableaux as a framework for automated theorem-proving. In J. Hallam and C. Mellish, editors, *Advances in Artificial Intelligence, AISB-87*, pages 125–139. Wiley, 1987.

[Smu63] Raymond M. Smullyan. A unifying principle for quantification theory. *Proc. Nat. Acad Sciences*, 49:828–832, 1963.

[Smu68] Raymond M. Smullyan. *First-Order Logic*. Springer Verlag, 1968.

[Sny91] Wayne Snyder. *A Proof Theory for General Unification*. Progress in Computer Science and Applied Logic. Birkhäuser, 1991.

[vH67] Jean van Heijenoort, editor. *From Frege to Göel A Soruce Book in Mathematical Logic, 1879-1931*. Source Books in the History of the Sciences. Harvard University Press, 1967.