Analytic Tableaux for Higher-Order Logic with Choice

Julian Backes and Chad E. Brown

Saarland University, Saarbrücken, Germany

Abstract. While many higher-order interactive theorem provers include a choice operator, higher-order automated theorem provers currently do not. As a step towards supporting automated reasoning in the presence of a choice operator, we present a cut-free ground tableau calculus for Church's simple type theory with choice. The tableau calculus is designed with automated search in mind. In particular, the rules only operate on the top level structure of formulas. Additionally, we restrict the instantiation terms for quantifiers to a universe that depends on the current branch. At base types the universe of instantiations is finite. We prove completeness of the tableau calculus relative to Henkin models.

1 Introduction

Interactive theorem provers based on classical higher-order logic (e.g., Isabelle-HOL [16], HOL [12] and the successors of the HOL system) build in the axiom of choice by including a form of Hilbert's ε binder and appropriate rules. Church's formulation of the simple theory of types [11] included a choice operator (called ι) and an axiom of choice at each type. Henkin defined a general notion of a model of Church's type theory with choice and proved completeness [13]. A higher-order version of the TPTP has been under development the past few years [18]. In 2009 it was decided that Henkin models with choice would be the default semantics of the higher-order TPTP.

Automated theorem provers for classical higher-order logic (e.g., TPS [3] and LEO-II [7]) do not currently build in the axiom of choice. Completeness of such calculi is judged with respect to a variant of Henkin's models without choice [2, 6]. What would be involved in adding support for choice? Assume a new logical constant ε_σ of type $(\sigma \to o) \to \sigma$ at each type σ is added to the syntax. We need new rules corresponding to this constant. The fundamental property ε_σ should satisfy is expressed by the formula

$$\forall p_{\sigma\to o} x_\sigma . px \to p(\varepsilon_\sigma p)$$

Our purpose in this paper is to give a complete analytic tableau calculus for higher-order logic with choice that forms a basis for automated reasoning in the logic. Mints [15] has given a sequent calculus for relational higher-order logic with an ε-operator and proves completeness. Mints' calculus does not include arbitrary function types and the corresponding simply typed λ-terms. We adapt Mints' rules for a simply typed formulation in the style of Church. We obtain tighter restrictions on when Mints' main choice rule (the ε-rule) needs to be applied. Furthermore, we show we can omit Mints' ε-extensionality rule altogether. These results are important for automated reasoning because these two rules would be highly branching in practice. In addition to including cut-free rules for the ε-operator, we give strong restrictions on the instantiation of universal quantifiers over base types analogous to those reported in [9].

In Section 2 we give a quick presentation of the syntax and semantics of simple type theory with choice. In Section 3 we present the tableau calculus. In Section 4 we define the notion of an evident set and prove that every evident set has a Henkin model. We define a notion of abstract consistency and use it to prove completeness of the tableau calculus in Section 5. We discuss related work and conclude in Sections 6 and 7. For reasons of space several proofs are omitted. Detailed proofs are available in [5].

2 Preliminaries

We start by giving the syntax for simple type theory with a choice operator in the style of Church [11]. Types (σ, τ, μ) are given inductively by the base type o (of truth values), ι (of individuals) and $\sigma \to \tau$ (of functions from σ to τ). For brevity, we will omit the arrow and write $\sigma\tau$ for $\sigma \to \tau$. Omitted parenthesis in types associate to the right: $\sigma\tau\mu$ means $\sigma(\tau\mu)$. The results in the paper generalize to the case where there are arbitrarily many base types of individuals. We use β to range over the base types o and ι.

For each type σ we assume a countably infinite set $\mathcal{V}_\sigma$ of variables of type σ. For each type σ we have logical constants $=_\sigma$ of type $\sigma\sigma o$, $\forall_\sigma$ of type $(\sigma o)o$ and ε_σ (the choice operator) of type $(\sigma o)\sigma$. Furthermore, we have logical constants for disjunction $\vee$ of type ooo, negation $\neg$ of type oo, false $\bot$ of type o and for a default individual $*$ of type ι. (The default individual is included only to act as an instantiation when no other instantiation of type ι is allowed by our calculus.) We use x, y to range over variables and c to range over logical constants. A name is either a variable or a logical constant. We use ν to range over names. Variables x and choice operators ε_σ are called *decomposable* names. We use δ to range over decomposable names.

The family of sets Λ_σ of terms of type σ are inductively defined. If ν is a name of type σ, then $\nu \in \Lambda_\sigma$. If $t \in \Lambda_{\sigma\tau}$ and $s \in \Lambda_\sigma$, then we have an application term $ts \in \Lambda_\tau$. If $x \in \mathcal{V}_\sigma$ and $t \in \Lambda_\tau$, then we have an abstraction term $\lambda x.s \in \Lambda_{\sigma\tau}$. A *formula* is a term $s \in \Lambda_o$.

Application associates to the left, so that stu means $(st)u$, with the exception that $\neg tu$ always means $\neg(tu)$. We use infix notation and write $s =_\sigma t$ (or $s = t$) for $=_\sigma st$ and write $s \vee t$ for $\vee st$. We write $s \neq_\sigma t$ (or $s \neq t$) for $\neg(s =_\sigma t)$. We also use binder notation to write $\forall x.s$ for $\forall_\sigma \lambda x.s$ and write $\varepsilon x.s$ for $\varepsilon_\sigma \lambda x.s$.

The set $\mathcal{V}t$ of *free variables of* t is defined as usual. For a set of variables X we write Λ_σ^X for the set of all terms $t \in \Lambda_\sigma$ such that $\mathcal{V}t \subseteq X$.

An *elimination context* ($\mathcal{E}$) is a term with a hole $[]_\sigma$ defined inductively as follows. $[]_\sigma$ is an elimination context of type σ. If $\mathcal{E}$ is an elimination context of type $\tau\mu$ and $s \in \Lambda_\tau$ then $\mathcal{E}s$ is an elimination context of type μ.

Let $\mathcal{E}$ be an elimination context of type σ which has a hole of type τ. We can apply $\mathcal{E}$ to a term $t \in \Lambda_\tau$ to get a term of type σ: $[\,][t] = t$ and $(\mathcal{E}\ s)[t] = \mathcal{E}[t]\ s$.

An *accessibility context* ($\mathcal{C}$) is a term with a hole $[]_\sigma$ of the form $\mathcal{E}$, $\neg\mathcal{E}$, $\mathcal{E} \neq_\iota s$ or $s \neq_\iota \mathcal{E}$ where $\mathcal{E}$ is an elimination context. We can apply an accessibility context $\mathcal{C}$ with a hole of type σ to a term $t \in \Lambda_\sigma$ to get a term of type o in the obvious way. A term s is *accessible* in a set A of formulas iff there is an accessibility context $\mathcal{C}$ such that $\mathcal{C}[s] \in A$.

Let A be a set of formulas. A term s is *discriminating* in A iff there is a term t such that $s \neq_\iota t \in A$ or $t \neq_\iota s \in A$. A *discriminant* Δ of A is a maximal set of discriminating terms such that there is no $s, t \in \Delta$ with $s \neq_\iota t \in A$. (Discriminants first appeared in [10].)

We now turn to a brief description of the semantics. Our notion of an interpretation is essentially that given by Henkin [13]. A *frame* $\mathcal{D}$ is a typed family of nonempty sets such that $\mathcal{D}_o = \{0, 1\}$ and $\mathcal{D}_{\sigma\tau}$ is a set of total functions from $\mathcal{D}_\sigma$ to $\mathcal{D}_\tau$. $\mathcal{D}_o$ is the set of Booleans 0 (*false*) and 1 (*true*). An *assignment* into a frame $\mathcal{D}$ is a function $\mathcal{I}$ that maps every name ν of type σ to an element of $\mathcal{D}_\sigma$. We denote $\mathcal{I}^x_a$ to be the assignment that is like $\mathcal{I}$ but maps the variable x to a.

prop.	where	holds		for all
$\mathfrak{L}_*(a)$	$a \in \mathcal{D}_\iota$	always		
$\mathfrak{L}_\bot(a)$	$a \in \mathcal{D}_o$	when $a = 0$		
$\mathfrak{L}_\neg(n)$	$n \in \mathcal{D}_{oo}$	when $na = 1$	iff $a = 0$	$a \in \mathcal{D}_o$
$\mathfrak{L}_\vee(d)$	$d \in \mathcal{D}_{ooo}$	when $dab = 1$	iff $a = 1$ or $b = 1$	$a, b \in \mathcal{D}_o$
$\mathfrak{L}_{\forall_\sigma}(p)$	$p \in \mathcal{D}_{(\sigma o)o}$	when $pf = 1$	iff $\forall a \in \mathcal{D}_\sigma\ fa = 1$	$f \in \mathcal{D}_{\sigma o}$
$\mathfrak{L}_{=_\sigma}(q)$	$q \in \mathcal{D}_{\sigma\sigma o}$	when $qab = 1$	iff $a = b$	$a, b \in \mathcal{D}_\sigma$
$\mathfrak{L}_{\varepsilon_\sigma}(\Phi)$	$\Phi \in \mathcal{D}_{(\sigma o)\sigma}$	when $f(\Phi f) = 1$	iff $\exists a \in \mathcal{D}_\sigma\ fa = 1$	$f \in \mathcal{D}_{\sigma o}$

Table 1. Properties of values of logical constants

For each logical constant c of type σ we define a corresponding property $\mathfrak{L}_c(a)$ of elements $a \in \mathcal{D}_\sigma$ in Table 1. Essentially $\mathfrak{L}_c(a)$ holds iff a is an appropriate interpretation of c. An assignment $\mathcal{I}$ into $\mathcal{D}$ is *logical* if $\mathfrak{L}_c(\mathcal{I}c)$ holds for each logical constant c. A logical assignment $\mathcal{I}$ must map $\bot$ to 0, $\neg$ to the negation function, and so on. There is no restriction on the value of $\mathcal{I}*$ in $\mathcal{D}_\iota$. The most interesting case to consider is the choice function ε_σ. For an assignment to be logical, $\mathcal{I}\varepsilon_\sigma$ must be a function in $\mathcal{D}_{(\sigma o)\sigma}$ such that $f((\mathcal{I}\varepsilon_\sigma)f) = 1$ for every $f \in \mathcal{D}_{\sigma o}$ except when f is the constant 0 function. There may be many different elements in $\mathcal{D}_{(\sigma o)\sigma}$ satisfying this condition. (Of course, there may also be no element satisfying the condition.)

We now turn to the interpretation of all typed terms. To do this we use induction on terms to lift each assignment $\mathcal{I}$ to a partial function $\hat{\mathcal{I}}$ on terms:

$$\begin{aligned}\hat{\mathcal{I}}(\nu) &:= \mathcal{I}(\nu) \\ \hat{\mathcal{I}}(st) &:= fa \quad \text{if } \hat{\mathcal{I}}s = f \text{ and } \hat{\mathcal{I}}t = a \\ \hat{\mathcal{I}}(\lambda x.s) &:= f \quad\ \ \text{if } \lambda x.s \in \Lambda_{\sigma\tau},\ f \in \mathcal{D}_{\sigma\tau} \text{ and } \forall a \in \mathcal{D}_\sigma\colon \widehat{\mathcal{I}^x_a}s = fa\end{aligned}$$

If $\hat{\mathcal{I}}$ is a total function, then we say $\mathcal{I}$ is an *interpretation.*

A *model* $(\mathcal{D},\mathcal{I})$ is a frame $\mathcal{D}$ and a logical interpretation $\mathcal{I}$ into $\mathcal{D}$. We say that a model $(\mathcal{D},\mathcal{I})$ satisfies a formula s iff $\hat{\mathcal{I}}(s) = 1$. A formula is *satisfiable* iff there is a model $(\mathcal{D},\mathcal{I})$ such that $\hat{\mathcal{I}}(s) = 1$. We say $(\mathcal{D},\mathcal{I})$ is *a model of* a set of formulas A if $\hat{\mathcal{I}}(s) = 1$ for every $s \in A$. A set A of formulas is *satisfiable* if there is a model of A.

We assume a type preserving and total *normalization operator* $[\cdot]$ from terms to terms. A term is *normal* iff $[s] = s$. A set of terms is normal if every element of this set is normal. Instead of committing to a specific operator such as β-normalization or $\beta\eta$-normalization, we require the following properties:

N1 $[[s]] = [s]$
N2 $[[s]t] = [st]$
N3 $[\nu s_1 \dots s_n] = \nu[s_1]\dots[s_n]$ if $\nu s_1 \dots s_n \in \Lambda_\beta$ for some base type β and $n \geq 0$
N4 $\hat{\mathcal{I}}[s] = \hat{\mathcal{I}}s$ for every model $(\mathcal{D},\mathcal{I})$.
N5 $\mathcal{V}[s] \subseteq \mathcal{V}s$

Note that by N5 we know $[s] \in \Lambda^X_\sigma$ whenever $s \in \Lambda^X_\sigma$.

A *substitution* is a type preserving partial function from variables to terms. If θ is a substitution, x is a variable, and s is a term that has the same type as x, we write θ^x_s for the substitution that agrees everywhere with θ except possibly on x where it yields s. For each substitution θ we assume there is a type preserving total function $\hat{\theta}$ from terms to terms such that the following conditions hold:

S1 $\hat{\theta}x = \theta x$ for every $x \in \mathrm{Dom}\,\theta$
S2 $\hat{\theta}(st) = (\hat{\theta}s)(\hat{\theta}t)$
S3 $[(\hat{\theta}(\lambda x.s))t] = [\widehat{\theta^x_t}s]$
S4 $[\hat{\theta}s] = [s]$ if $\theta x = x$ for every $x \in \mathrm{Dom}\,\theta \cap \mathcal{V}s$
S5 $[\hat{\theta}[s]] = [\hat{\theta}s]$

The following proposition demonstrates that we can recover a form of β-reduction relative to abstract normalization and substitution. The empty set $\emptyset$ is the substitution that is undefined on every variable.

Proposition 1. $[[\lambda x.s]t] = [\widehat{\emptyset^x_t}s]$

Proof. $[[\lambda x.s]t] \overset{\text{S4}}{=} [[\hat{\emptyset}(\lambda x.s)]t] \overset{\text{N2}}{=} [(\hat{\emptyset}(\lambda x.s))t] \overset{\text{S3}}{=} [\widehat{\emptyset^x_t}s]$

For each set A of formulas and each type σ we define a nonempty universe $\mathcal{U}^A_\sigma \subseteq \Lambda_\sigma$ as follows.

- Let $\mathcal{U}^A_o = \{\bot, \neg\bot\}$.
- Let $\mathcal{U}^A_\iota$ be the set of discriminating terms in A if there is some discriminating term in A.

– Let $\mathcal{U}_\iota^A = \{*\}$ if there are no discriminating terms in A.
– Let $\mathcal{U}_{\sigma\tau}^A = \{[s] | s \in \Lambda_{\sigma\tau}, \mathcal{V}s \subseteq \mathcal{V}A\}$.

When trying to refute a set A of formulas, all our instantiations of type σ will come from the set $\mathcal{U}_\sigma^A$. When the set A is clear in context, we write $\mathcal{U}_\sigma$.

3 Tableau Calculus

A *branch* is a finite set of normal formulas. A *step* is an n+1-tuple $\langle A, A_1, \ldots, A_n\rangle$ of branches where $n \geq 1$, $\bot \notin A$ and $A \subset A_i$ for each $i \in \{1, \ldots, n\}$. The branch A is the *head* of the step $\langle A, A_1, \ldots, A_n\rangle$ and each A_i is an *alternative.* A *rule* is a set of steps, and is usually indicated by a schema. For example, the schema

$$\mathcal{T}_{\text{BE}} \quad \frac{s \neq_o t}{s\,,\, \neg t \;\mid\; \neg s\,,\, t}$$

indicates the set of steps $\langle A, A_1, A_2\rangle$ where $s \neq_o t$ is in A, $\bot \notin A$, $\{s, \neg t\} \not\subseteq A_1$, $\{\neg s, t\} \not\subseteq A_1$, $A_1 = A \cup \{s, \neg t\}$ and $A_2 = A \cup \{\neg s, t\}$. We say a rule *applies to* a branch A if some step in the rule has A as its head. A tableau calculus is also a set of steps. Our tableau calculus $\mathcal{T}$ is given as the union of the rules in Figure 1.

$$\mathcal{T}_\neg \ \frac{s,\, \neg s}{\bot} \qquad \mathcal{T}_{\neq} \ \frac{s \neq_\iota s}{\bot} \qquad \mathcal{T}_{\neg\neg} \ \frac{\neg\neg s}{s} \qquad \mathcal{T}_\vee \ \frac{s \vee t}{s \mid t} \qquad \mathcal{T}_{\neg\vee} \ \frac{\neg(s \vee t)}{\neg s, \neg t}$$

$$\mathcal{T}_\forall \ \frac{\forall_\sigma s}{[st]} \ t \in \mathcal{U}_\sigma \qquad\qquad \mathcal{T}_{\neg\forall} \ \frac{\neg\forall_\sigma s}{\neg[sx]} \ x \in \mathcal{V}_\sigma \text{ fresh}$$

$$\mathcal{T}_{\text{MAT}} \ \frac{\delta s_1 \ldots s_n\,,\, \neg\delta t_1 \ldots t_n}{s_1 \neq t_1 \mid \cdots \mid s_n \neq t_n} \ n \geq 1 \qquad \mathcal{T}_{\text{DEC}} \ \frac{\delta s_1 \ldots s_n \neq_\iota \delta t_1 \ldots t_n}{s_1 \neq t_1 \mid \cdots \mid s_n \neq t_n} \ n \geq 1$$

$$\mathcal{T}_{\text{CON}} \ \frac{s =_\iota t\,,\, u \neq_\iota v}{s \neq u\,,\, t \neq u \mid s \neq v\,,\, t \neq v} \qquad \mathcal{T}_{\text{BE}} \ \frac{s \neq_o t}{s\,,\, \neg t \;\mid\; \neg s\,,\, t} \qquad \mathcal{T}_{\text{BQ}} \ \frac{s =_o t}{s\,,\, t \;\mid\; \neg s\,,\, \neg t}$$

$$\mathcal{T}_{\text{FE}} \ \frac{s \neq_{\sigma\tau} t}{\neg[\forall x.sx = tx]} \ x \notin \mathcal{V}s \cup \mathcal{V}t \qquad \mathcal{T}_{\text{FQ}} \ \frac{s =_{\sigma\tau} t}{[\forall x.sx = tx]} \ x \notin \mathcal{V}s \cup \mathcal{V}t$$

$$\mathcal{T}_\varepsilon \ \frac{}{[\forall x.\neg(sx)] \;\mid\; [s(\varepsilon s)]} \ \varepsilon s \text{ accessible, } x \notin \mathcal{V}s$$

Fig. 1. Tableau rules

In the rules $\mathcal{T}_{\text{MAT}}$ (the mating rule) and $\mathcal{T}_{\text{DEC}}$ (the decomposition rule) δ ranges over decomposable names (variables and choice operators). In the rule $\mathcal{T}_\forall$ the instantiation term t must belong to the set $\mathcal{U}_\sigma^A$ where A is the head of the step. In the rule $\mathcal{T}_{\neg\forall}$ the variable x must be *fresh* in the sense that it is not in $\mathcal{V}A$ where A is the head of the step. We restrict the $\mathcal{T}_{\neg\forall}$ to apply only in the case where there is no variable $y \in \mathcal{V}_\sigma$ such that $\neg[sy]$ is in the head A. In the context of an automated prover, this restriction implies there is no need to apply the $\mathcal{T}_{\neg\forall}$ rule to a formula $\neg\forall s$ more than once.

We explain the choice rule $\mathcal{T}_\varepsilon$. Whenever we must consider εs, either s corresponds to the empty set and hence $\forall x.\neg(sx)$ holds, or s represents a set containing at least one element and $s(\varepsilon s)$ holds. Note that we obtain a complete calculus even though we only apply the choice rule when εs occurs on the branch in the form $\mathcal{C}[\varepsilon s]$ for some accessibility context $\mathcal{C}$. That is, the choice rule only applies using εs when the branch contains a formula of the form $\varepsilon s t_1 \cdots t_n$, $\neg(\varepsilon s t_1 \cdots t_n)$, $(\varepsilon s t_1 \cdots t_n) \neq_\iota u$ or $u \neq_\iota (\varepsilon s t_1 \cdots t_n)$. This is a tighter restriction than the one given for the choice rule in [15].

The set of *refutable* branches is defined inductively as follows. If $\bot \in A$, then A is refutable. If $\langle A, A_1, \ldots, A_n\rangle$ is a step in $\mathcal{T}$ and every alternative A_i is refutable, then A is refutable.

Proposition 2 (Soundness). *If A is refutable, then A is unsatisfiable.*

Proof. It is enough to check for each step $\langle A, A_1, \cdots, A_n\rangle$ in $\mathcal{T}$ that if A is satisfiable, then A_i is satisfiable for some $i \in \{1, \ldots, n\}$. Each case is easy. For the steps involving the normalization operator, property N4 is used.

Example 1. Let $p \in \mathcal{V}_{\iota o}$. For this example, assume p and $\lambda x.\neg px$ are normal. We refute the set $\{p(\varepsilon x.\neg px), \neg p(\varepsilon p)\}$ using the rules $\mathcal{T}_{\text{MAT}}$, $\mathcal{T}_\varepsilon$, $\mathcal{T}_\forall$ and $\mathcal{T}_\neg$.

$$\begin{array}{c}
p(\varepsilon x.\neg px) \\
\neg p(\varepsilon p) \\
(\varepsilon x.\neg px) \neq \varepsilon p \\
\hline
\begin{array}{c|c}
\begin{array}{c} \forall x.\neg px \\ \neg p(\varepsilon x.\neg px) \\ \bot \end{array} & \begin{array}{c} p(\varepsilon p) \\ \bot \end{array}
\end{array}
\end{array}$$

4 Evident Sets and Model Existence

Let E be a set of normal formulas. We say E is *evident* if it satisfies the conditions in Figure 2. The conditions $\mathcal{E}_{\text{FE}}$, $\mathcal{E}_{\text{FQ}}$ and $\mathcal{E}_\varepsilon$ are formulated in a slightly different way than the corresponding tableau rules $\mathcal{T}_{\text{FE}}$, $\mathcal{T}_{\text{FQ}}$ and $\mathcal{T}_\varepsilon$. The tableau rules are formulated in a way that makes proof search more practical while the evidence conditions are formulated in a way that will ease the model construction. The next proposition demonstrates that these three evidence conditions could also be formulated differently. Later we will use the proposition to help prove certain sets are evident. We omit the proof.

Proposition 3. *Let E be a set of normal formulas satisfying $\mathcal{E}_\forall$ and $\mathcal{E}_{\neg\forall}$.*

$\mathcal{E}_\bot$	$\bot$ is not in E.
$\mathcal{E}_\neg$	If $\neg s$ is in E, then s is not in E.
$\mathcal{E}_{\neq}$	$s \neq_\iota s$ is not in E.
$\mathcal{E}_{\neg\neg}$	If $\neg\neg s$ is in E, then s is in E.
$\mathcal{E}_\lor$	If $s \lor t$ is in E, then either s or t is in E.
$\mathcal{E}_{\neg\lor}$	If $\neg(s \lor t)$ is in E, then $\neg s$ and $\neg t$ are in E.
$\mathcal{E}_\forall$	If $\forall_\sigma s$ is in E, then $[st]$ is in E for every $t \in \mathcal{U}^E_\sigma$.
$\mathcal{E}_{\neg\forall}$	If $\neg\forall_\sigma s$ is in E, then $\neg[sx]$ is in E for some $x \in \mathcal{V}_\sigma$.
$\mathcal{E}_{\text{MAT}}$	If $\delta s_1 \dots s_n$ and $\neg\delta t_1 \dots t_n$ are in E where $n \geq 1$, then $s_i \neq t_i$ is in E for some $i \in \{1, \dots, n\}$.
$\mathcal{E}_{\text{DEC}}$	If $\delta s_1 \dots s_n \neq_\iota \delta t_1 \dots t_n$ is in E where $n \geq 1$, then $s_i \neq t_i$ is in E for some $i \in \{1, \dots, n\}$.
$\mathcal{E}_{\text{CON}}$	If $s =_\iota t$ and $u \neq_\iota v$ are in E, then either $s \neq u$ and $t \neq u$ are in E or $s \neq v$ and $t \neq v$ are in E.
$\mathcal{E}_{\text{BE}}$	If $s \neq_o t$ is in E, then either s and $\neg t$ are in E or $\neg s$ and t are in E.
$\mathcal{E}_{\text{BQ}}$	If $s =_o t$ is in E, then either s and t are in E or $\neg s$ and $\neg t$ are in E.
$\mathcal{E}_{\text{FE}}$	If $s \neq_{\sigma\tau} t$ is in E, then $[sx] \neq [tx]$ is in E for some $x \in \mathcal{V}_\sigma$.
$\mathcal{E}_{\text{FQ}}$	If $s =_{\sigma\tau} t$ is in E, then $[su] = [tu]$ is in E for every $u \in \mathcal{U}^E_\sigma$.
$\mathcal{E}_\varepsilon$	If $\varepsilon_\sigma s$ is accessible in E, then either $[s(\varepsilon s)]$ is in E or $\neg[st]$ is in E for every $t \in \mathcal{U}^E_\sigma$.

Fig. 2. Evidence conditions

1. *For $s, t \in \Lambda_{\sigma\tau}$ and $x \in \mathcal{V}_\sigma \setminus (\mathcal{V}s \cup \mathcal{V}t)$, if $\neg[\forall x.sx =_\tau tx]$ is in E, then $[sy] \neq [ty]$ is in E for some $y \in \mathcal{V}_\sigma$.*
2. *For $s, t \in \Lambda_{\sigma\tau}$ and $x \in \mathcal{V}_\sigma \setminus (\mathcal{V}s \cup \mathcal{V}t)$, if $[\forall x.sx =_\tau tx]$ is in E, then $[su] = [tu]$ is in E for every $u \in \mathcal{U}^E_\sigma$.*
3. *For $s \in \Lambda_{\sigma\tau}$ and $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$, if $[\forall x.\neg sx]$ is in E, then $\neg[st]$ is in E for every $t \in \mathcal{U}^E_\sigma$.*

Let E be an evident set. In the rest of this section we will construct a model of E. The construction is similar to the ones in [8, 9] except for some complications arising from the instantiation restrictions.

Let X be the set $\mathcal{V}E$ of free variables in E. We begin by defining a binary relation $\triangleright_\sigma$ by induction on types. For each σ, let $\mathcal{D}_\sigma$ be the range of $\triangleright_\sigma$, i.e., set of all a such that there is some $s \in \Lambda^X_\sigma$ such that $s \triangleright_\sigma a$.

- $s \triangleright_o 0$ if $s \in \Lambda^X_o$ and $[s] \notin E$.
- $s \triangleright_o 1$ if $s \in \Lambda^X_o$ and $\neg[s] \notin E$.
- $s \triangleright_\iota \Delta$ if $s \in \Lambda^X_\iota$, Δ is a discriminant (of E), and either $[s]$ is not a discriminating term or $[s] \in \Delta$.
- $s \triangleright_{\sigma\tau} f$ if $s \in \Lambda^X_{\sigma\tau}$, $f : \mathcal{D}_\sigma \to \mathcal{D}_\tau$ and $st \triangleright_\tau fa$ whenever $t \triangleright_\sigma a$.

Clearly we have $\triangleright_\sigma \subseteq \Lambda^X_\sigma \times \mathcal{D}_\sigma$. Also, by definition of $\mathcal{D}$ we have that for every $a \in \mathcal{D}_\sigma$ there is some $s \in \Lambda^X_\sigma$ such that $s \triangleright_\sigma a$. For any set $T \subseteq \Lambda^X_\sigma$ we write $T \triangleright a$ if $s \triangleright a$ for every $s \in T$.

Lemma 1. *For all types σ, terms $s \in \Lambda^X_\sigma$ and values $a \in \mathcal{D}_\sigma$, $s \triangleright a$ iff $[s] \triangleright a$.*

Proof. This follows by an easy induction on types σ using N1, N2 and N5. The proof is essentially the same as that of Lemma 3.3 in [8].

The next proposition records a number of useful facts about $\triangleright$ and $\mathcal{D}$. In particular, $\mathcal{D}$ is a frame and for every value $a \in \mathcal{D}_\sigma$ there is some $t \in \mathcal{U}^E_\sigma$ such that $t \triangleright a$. We omit the proof.

Proposition 4.

1. *$\bot \triangleright 0$ and $\neg\bot \triangleright 1$. In particular, $\mathcal{D}_o = \{0,1\}$.*
2. *For every discriminant Δ, there is a term $t \in \mathcal{U}^E_\iota$ such that $t \triangleright \Delta$. In particular, $\mathcal{D}_\iota$ is the set of all discriminants.*
3. *For all types σ and $a \in \mathcal{D}_\sigma$ there is a term $t \in \mathcal{U}^E_\sigma$ such that $t \triangleright a$.*
4. *If $t \triangleright_\mu b$ and $x \in \mathcal{V}_\tau \setminus \mathcal{V}t$, then $\lambda x.t \triangleright K_b$ where $K_b : \mathcal{D}_\tau \to \mathcal{D}_\mu$ is the constant b function.*
5. *For all types σ, $\mathcal{D}_\sigma$ is nonempty.*
6. *$\mathcal{D}$ is a frame.*

We now turn to a notion of compatibility of terms. For $s,t \in \Lambda^X_\sigma$ we say $s\sharp t$ holds if either $s \neq t$ or $t \neq s$ is in E.

Definition 1. *For each type σ we define when two terms $s,t \in \Lambda^X_\sigma$ are* compatible *(written $s \parallel t$) by induction on types.*

$\sigma = o$: *$s \parallel t$ if $\{[s], \neg[t]\} \not\subseteq E$ and $\{\neg[s],[t]\} \not\subseteq E$.*
$\sigma = \iota$: *$s \parallel t$ if $[s]\sharp[t]$ does not hold.*
$\sigma = \tau\mu$: *$s \parallel t$ if for all $u,v \in \Lambda^X_\tau$ $u \parallel v$ implies $su \parallel tv$.*

We say a set $T \subseteq \Lambda^X_\sigma$ is compatible *if $s \parallel t$ for all $s,t \in T$.*

The next lemma provides relationships between compatibility of terms and the presence of disequations in E. Note that part (2) of the lemma implies $\varepsilon_\sigma \parallel \varepsilon_\sigma$ for every type σ and $x \parallel x$ for every variable $x \in X$.

Lemma 2. *For all types σ we have the following:*

1. *For all $s,t \in \Lambda^X_\sigma$, if $s \parallel t$, then $[s]\sharp[t]$ does not hold.*
2. *For all $\delta s_1 \cdots s_n, \delta t_1, \cdots t_n \in \Lambda^X_\sigma$ where $n \geq 0$ and δ is a decomposable name, either $\delta s_1 \cdots s_n \parallel \delta t_1, \cdots t_n$ or there is some $i \in \{1,\ldots,n\}$ such that $[s_i]\sharp[t_i]$.*

Proof. By mutual induction on σ. The only complicated case is proving (1) when σ is $\tau\mu$. Assume $s \parallel t$ holds and $[s]\sharp[t]$ holds. By $\mathcal{E}_{\text{FE}}$ $[[s]x]\sharp[[t]x]$ for some variable x. If $x \in X$, then $x \parallel_\tau x$ by inductive hypothesis (2) and so $sx \parallel_\mu tx$, contradicting inductive hypothesis (1) and N2. Assume $x \notin X$. In particular, $x \notin \mathcal{V}s \cup \mathcal{V}t \cup \mathcal{V}[sx] \cup \mathcal{V}[tx]$. In this case we can prove $[sx]$ is $[s(\varepsilon x.\bot)]$ and $[tx]$ is $[t(\varepsilon x.\bot)]$. Using the inductive hypothesis (2) we can prove $\varepsilon x.\bot \parallel \varepsilon x.\bot$ and derive a contradiction.

The next lemma relates compatibility to $\triangleright$ and can be proven by an easy induction on types.

Lemma 3. *For all sets $T \subseteq \Lambda^X_\sigma$, T is compatible iff there exists a value $a \in \mathcal{D}_\sigma$ such that $T \triangleright a$.*

We now turn to the interpretation of the choice operators. We use a construction similar to that of Mints [15] adapted to our setting.

Let $f \in \mathcal{D}_{\sigma o}$ be a function and εs be a term in Λ_σ^X. We write $f \propto \varepsilon s$ (read f *chooses* εs) iff $s \triangleright f$ and $\varepsilon[s]$ is accessible in E. Let $f^0 := \{\varepsilon s \in \Lambda_\sigma^X | f \propto \varepsilon s\}$.

Lemma 4. *Let E be an evident set and let $f \in \mathcal{D}_{\sigma o}$ be a function. Then, there is some $a \in \mathcal{D}_\sigma$ such that $f^0 \triangleright a$.*

Proof. We show that f^0 is compatible. Lemma 3 gives us the claim. Let $\varepsilon s, \varepsilon t \in f^0$. By definition of $\propto$, $s, t \triangleright f$ and hence, by Lemma 3, $s \parallel t$. By Lemma 2(2) $\varepsilon \parallel \varepsilon$. Thus $\varepsilon s \parallel \varepsilon t$.

For any type σ, we define $\Phi_\sigma \in \mathcal{D}_{\sigma o} \to \mathcal{D}_\sigma$ as follows:

$$\Phi_\sigma f = \begin{cases} \text{some } b & \text{such that } fb = 1 \text{ if } f^0 \text{ is empty and such a } b \text{ exists.} \\ \text{some } a & \text{such that } f^0 \triangleright a. \end{cases}$$

The existence of an a in the second case follows from Lemma 4. Note that the second case includes the case in which f is the constant 0 function. In particular, if f is the constant 0 function and f^0 is empty, then $\Phi_\sigma f$ can be any $a \in \mathcal{D}_\sigma$.

Lemma 5. *Let E be an evident branch, ε be a choice operator, $\varepsilon t_1 \dots t_n \in \Lambda_\sigma^X$ and $a \in \mathcal{D}_\sigma$. If $\varepsilon t_1 \dots t_n \not\triangleright a$, then $\varepsilon[t_1] \dots [t_n]$ is accessible in E.*

Proof. The proof is by an easy induction on σ.

Lemma 6. *For any type σ we have $\varepsilon_\sigma \triangleright \Phi_\sigma$.*

Proof. Assume $\varepsilon \not\triangleright \Phi$. Then, there are s, f such that $s \triangleright f$ but $\varepsilon s \not\triangleright \Phi f$. By Lemma 5 $\varepsilon[s]$ is accessible in E. Hence $\varepsilon s \in f^0$. There is some a such that $\Phi f = a$ and $f^0 \triangleright a$. Thus $\varepsilon s \triangleright a$, a contradiction.

Lemma 7. *$\mathfrak{L}_{\varepsilon_\sigma}(\Phi_\sigma)$ holds. That is, Φ as defined above is a choice function.*

Proof. Let $f \in \mathcal{D}_{\sigma o}$ be a function and $b \in \mathcal{D}_\sigma$ be such that $fb = 1$. Suppose $f(\Phi f) = 0$. Then f^0 must be nonempty (by the definition of Φf). Choose some $\varepsilon s \in f^0$. We will show a contradiction. By $\mathcal{E}_\varepsilon$ there are two possibilities:

1. $[s(\varepsilon s)] \in E$: In this case $s(\varepsilon s) \not\triangleright 0$. On the other hand, $s \triangleright f$ and $\varepsilon \triangleright \Phi$ (by Lemma 6) and so $s(\varepsilon s) \triangleright f(\Phi f)$. This contradicts our assumption that $f(\Phi f) = 0$.
2. $\neg[st] \in E$ for every $t \in \mathcal{U}_\sigma^E$. By Proposition 4(3) there is some term $t \in \mathcal{U}_\sigma^E$ such that $t \triangleright b$. Hence $\neg[st] \in E$. By definition of $\triangleright_o$, $st \not\triangleright 1$. On the other hand, we know $st \triangleright fb$ since $s \triangleright f$ and $t \triangleright b$, contradicting the assumption that $fb = 1$.

Lemma 8. *If $s \triangleright_\sigma a$, $t \triangleright_\sigma b$ and $s = t$ is in E, then $a = b$.*

Lemma 9. *For each $c \in \Lambda_\sigma$ there is some $a \in \mathcal{D}_\sigma$ such that $\mathfrak{L}_c(a)$ and $c \triangleright a$.*

Proof. If c is a choice operator ε_σ, then we know $\varepsilon_\sigma \triangleright \Phi_\sigma$ and $\mathfrak{L}_{\varepsilon_\sigma}(\Phi_\sigma)$ by Lemmas 6 and 7. Checking for the other logical constants is tedious, but not difficult. Lemma 8 is used in the case where c is $=_\sigma$.

We say an assignment $\mathcal{I}$ into $\mathcal{D}$ is *admissible* if $c \triangleright \mathcal{I}c$ for all logical constants c. The following lemma can be proven by induction on terms.

Lemma 10. *Let s be a term, θ be a substitution and $\mathcal{I}$ be an admissible assignment into $\mathcal{D}$. Suppose for every $x \in \mathcal{V}s$, $x \in \operatorname{Dom}\theta$ and $\theta x \triangleright \mathcal{I}x$. Then $s \in \operatorname{Dom}\hat{\mathcal{I}}$ and $\hat{\theta}s \triangleright \hat{\mathcal{I}}s$.*

Now we can prove the model existence theorem for evident sets.

Theorem 1 (Model Existence). *Every evident set is satisfiable.*

Proof. Let E be an evident set. Take $\triangleright$ and $\mathcal{D}$ as defined in this section. We define an assignment $\mathcal{I}$ as follows. For each logical constant c we can choose $\mathcal{I}c$ such that $c \triangleright \mathcal{I}c$ and $\mathfrak{L}_c(\mathcal{I}c)$ by Lemma 9. This ensures we will have a logical, admissible assignment. For each variable $x \in X$ we know $x \parallel x$ by Lemma 2(2) and we can choose $\mathcal{I}x$ such that $x \triangleright \mathcal{I}x$ by Lemma 3. For each variable $x \in \mathcal{V}_\sigma$ not in X we take $\mathcal{I}x = \Phi_\sigma K_0$ where K_0 is the constant 0 function. Let θ be the substitution mapping each $x \in X$ to x and each variable $x \in \mathcal{V}_\sigma$ not in X to $\varepsilon_\sigma y.\bot \in \Lambda_\sigma^X$. By Lemma 6 and Proposition 4(4) we know that $\varepsilon_\sigma y.\bot \triangleright \Phi K_0$ and hence $\theta x \triangleright \mathcal{I}x$ for every variable x. By Lemma 10 we know every $s \in \operatorname{Dom}\hat{\mathcal{I}}$ and $\hat{\theta}s \triangleright \hat{\mathcal{I}}s$ for every term s. In particular, $\mathcal{I}$ is an interpretation. It remains to prove $\hat{\mathcal{I}}s = 1$ for all $s \in E$. Let $s \in E$ be given. By S4 we know $[\hat{\theta}s] = [s]$. Using this and Lemma 1 we know $s \triangleright \hat{\mathcal{I}}s$. Since $s \not\triangleright 0$ and $s \triangleright \hat{\mathcal{I}}s$, we must have $\hat{\mathcal{I}}s = 1$ as desired.

We can now prove that if the tableau calculus $\mathcal{T}$ cannot make progress on a branch, then this branch is satisfiable and in fact has a model with finitely many individuals.

Corollary 1. *Let A be a branch. Suppose $\bot \notin A$ and A is not the head of any step in the calculus $\mathcal{T}$. Then A is evident and there is a model $(\mathcal{D}, \mathcal{I})$ of A where $\mathcal{D}_\iota$ is finite.*

Proof. Once we know A is evident, we know we have a model $(\mathcal{D}, \mathcal{I})$ of A where $\mathcal{D}_\iota$ is the set of discriminants of A. Since A is finite, there are only finitely many discriminating terms of A and hence only finitely many discriminants.

The evidence condition $\mathcal{E}_\bot$ follows from the assumption that $\bot \notin A$. The conditions $\mathcal{E}_\neg$ and $\mathcal{E}_{\neq}$ follows from $\bot \notin A$ and the assumption that the rules $\mathcal{T}_\neg$ and $\mathcal{T}_{\neq}$ do not apply to A. Except for $\mathcal{E}_{\text{FE}}$, $\mathcal{E}_{\text{FQ}}$ and $\mathcal{E}_\varepsilon$, the remaining evidence conditions follow immediately from the assumption that the corresponding rule does not apply. After we know $\mathcal{E}_\forall$ and $\mathcal{E}_{\neg\forall}$ hold for A, we can conclude that $\mathcal{E}_{\text{FE}}$, $\mathcal{E}_{\text{FQ}}$ and $\mathcal{E}_\varepsilon$ hold for A using Proposition 3 and the assumption that the corresponding rule does not apply.

$\mathcal{C}_\perp$	$\perp$ is not in A.
$\mathcal{C}_\neg$	If $\neg s$ is in A, then s is not in A.
$\mathcal{C}_{\neq}$	$s \neq_\iota s$ is not in A.
$\mathcal{C}_{\neg\neg}$	If $\neg\neg s$ is in A, then $A \cup \{s\}$ is in Γ.
$\mathcal{C}_\vee$	If $s \vee t$ is in A, then $A \cup \{s\}$ or $A \cup \{t\}$ is in Γ.
$\mathcal{C}_{\neg\vee}$	If $\neg(s \vee t)$ is in A, then $A \cup \{\neg s, \neg t\}$ is in Γ.
$\mathcal{C}_\forall$	If $\forall_\sigma s$ is in A, then $A \cup \{[st]\}$ is in Γ for every $t \in \mathcal{U}_\sigma^A$.
$\mathcal{C}_{\neg\forall}$	If $\neg\forall_\sigma s$ is in A, then $A \cup \{\neg[sx]\}$ is in Γ for some variable x.
$\mathcal{C}_{\text{MAT}}$	If $xs_1 \ldots s_n$ is in A and $\neg xt_1 \ldots t_n$ is in A, then $n \geq 1$ and $A \cup \{s_i \neq t_i\}$ is in Γ for some $i \in \{1, \ldots, n\}$.
$\mathcal{C}_{\text{DEC}}$	If $xs_1 \ldots s_n \neq_\iota xt_1 \ldots t_n$ is in A, then $n \geq 1$ and $A \cup \{s_i \neq t_i\}$ is in Γ for some $i \in \{1, \ldots, n\}$.
$\mathcal{C}_{\text{CON}}$	If $s =_\iota t$ and $u \neq_\iota v$ are in A, then either $A \cup \{s \neq u, t \neq u\}$ or $A \cup \{s \neq v, t \neq v\}$ is in Γ.
$\mathcal{C}_{\text{BE}}$	If $s \neq_o t$ is in A, then either $A \cup \{s, \neg t\}$ or $A \cup \{\neg s, t\}$ is in Γ.
$\mathcal{C}_{\text{BQ}}$	If $s =_o t$ is in A, then either $A \cup \{s, t\}$ or $A \cup \{\neg s, \neg t\}$ is in Γ.
$\mathcal{C}_{\text{FE}}$	If $s \neq_{\sigma\tau} t$ is in A, then $A \cup \{\neg[\forall x.sx =_\tau tx]\}$ is in Γ for some $x \in \mathcal{V}_\sigma \setminus (\mathcal{V}s \cup \mathcal{V}t)$.
$\mathcal{C}_{\text{FQ}}$	If $s =_{\sigma\tau} t$ is in A, then $A \cup \{[\forall x.sx =_\tau tx]\}$ is in Γ for some $x \in \mathcal{V}_\sigma \setminus (\mathcal{V}s \cup \mathcal{V}t)$.
$\mathcal{C}_\varepsilon$	If $\varepsilon_\sigma s$ is accessible in A, then either $A \cup \{[s(\varepsilon s)]\}$ is in Γ or there is some $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$ such that $A \cup \{[\forall x.\neg(sx)]\}$ is in Γ.

Fig. 3. Abstract consistency conditions (must hold for every $A \in \Gamma$)

Example 2. Let $p \in \mathcal{V}_{\iota o}$ and $q \in \mathcal{V}_o$. For this example assume $[s] = s$ for all $\beta\eta$-normal forms s. We prove $\forall_o q.\varepsilon_{\iota o} p \neq \varepsilon_{\iota o} x.q$ is satisfiable. Applying tableau rules we can construct a branch with the following formulas:

$$\begin{array}{cccccc} \forall_o q.\varepsilon p \neq \varepsilon x.q, & \varepsilon p \neq \varepsilon x.\perp, & \varepsilon p \neq \varepsilon x.\neg\perp, & p \neq \lambda x.\perp, & p \neq \lambda x.\neg\perp, & \neg\forall x.px = \perp, \\ \neg\forall x.px = \neg\perp, & px \neq \perp, & py \neq \neg\perp, & px, & \neg\perp, & \neg py, \\ x \neq y, & p(\varepsilon p), & \varepsilon p \neq y, & \forall x.\neg\perp & & \end{array}$$

5 Abstract Consistency and Completeness

We now lift the model existence theorem for evident sets to a model existence theorem for abstractly consistent sets. This will allow us to prove completeness of the tableau calculus $\mathcal{T}$. The use of abstract consistency to prove completeness was first used by Smullyan [17] and later used by several authors in various higher-order settings [1, 14, 6, 8].

A set Γ of branches is an *abstract consistency class* if it satisfies the conditions in Figure 3. In Lemma 12 we will prove that every member of an abstract consistency class can be extended to an evident set. In order to verify the $\mathcal{E}_\forall$ condition we will need the following lemma relating universes for different sets of formulas.

Lemma 11. *Let $\mathcal{A}$ be a nonempty subset of Γ and let E be $\bigcup \mathcal{A}$. Suppose for every branch $B \subseteq E$ there is a branch $A \in \mathcal{A}$ such that $B \subseteq A$. Then for every $t \in \mathcal{U}_\sigma^E$ there is some $A \in \mathcal{A}$ such that $t \in \mathcal{U}_\sigma^A$.*

We can now prove the desired extension lemma.

Lemma 12 (Extension Lemma). *Let Γ be an abstract consistency class. For every $A \in \Gamma$ there is an evident set E such that $A \subseteq E$.*

Proof. Let $u^0, u^1, \ldots$ be an enumeration of all normal formulas. We will construct a sequence $A_0 \subseteq A_1 \subseteq A_2 \subseteq \cdots$ of branches such that every $A_n \in \Gamma$. Let $A_0 := A$. We define A_{n+1} by cases. If there is no $B \in \Gamma$ such that $A_n \cup \{u_n\} \subseteq B$, then let $A_{n+1} := A_n$. Otherwise, choose some $B \in \Gamma$ such that $A_n \cup \{u_n\} \subseteq B$. We consider five subcases.

1. If u_n is of the form $\neg\forall_\sigma s$, then choose A_{n+1} to be $B \cup \{\neg[sx]\} \in \Gamma$ for some $x \in \mathcal{V}_\sigma$. This is possible since Γ satisfies $\mathcal{C}_{\neg\forall}$.
2. If u_n is of the form $s \neq_{\sigma\tau} t$, then choose A_{n+1} to be $B \cup \{\neg[\forall x.sx =_\tau tx]\} \in \Gamma$ for some $x \in \mathcal{V}_\sigma \setminus ([s] \cup [t])$. This is possible by $\mathcal{C}_{\text{FE}}$.
3. If u_n is of the form $s =_{\sigma\tau} t$, then choose A_{n+1} to be $B \cup \{[\forall x.sx =_\tau tx]\} \in \Gamma$ for some $x \in \mathcal{V}_\sigma \setminus ([s] \cup [t])$. This is possible by $\mathcal{C}_{\text{FQ}}$.
4. Suppose u_n is of the form $C[\varepsilon_\sigma s]$ where C is an accessibility context. (Note that if u_n is of this form, then it cannot be of one of the previous forms by the definition of an accessibility context.) By $\mathcal{C}_\varepsilon$ there either $B \cup \{[s(\varepsilon s)]\}$ is in Γ or there is some $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$ such that $B \cup \{[\forall x.\neg(sx)]\}$ is in Γ. If $B \cup \{[s(\varepsilon s)]\}$ is in Γ, then let A_{n+1} be $B \cup \{[s(\varepsilon s)]\}$. Otherwise, choose A_{n+1} to be $B \cup [\forall x.\neg(sx)] \in \Gamma$ for some $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$.
5. If no previous case applies, then let A_{n+1} be B.

Let $E := \bigcup_{n\in\mathbb{N}} A_n$. We must prove E satisfies the evidence conditions. We check only $\mathcal{E}_\forall$ and $\mathcal{E}_\varepsilon$ in detail, leaving the others to the reader. Proposition 3 is helpful for verifying $\mathcal{E}_{\text{FE}}$ and $\mathcal{E}_{\text{FQ}}$ just as it is helpful verifying $\mathcal{E}_\varepsilon$ below.

$\mathcal{E}_\forall$ Assume $\forall_\sigma s$ is in E. Let $t \in \mathcal{U}_\sigma^E$ be a normal term. Let n be such that $u_n = [st]$. By Lemma 11 (taking $\mathcal{A}$ to be $\{A_r | r \geq n \text{ and } \forall_\sigma s \in A_r\}$) there is some $r \geq n$ such that $t \in \mathcal{U}_\sigma^{A_r}$ and $\forall_\sigma s$ is in A_r. By $\mathcal{C}_\forall$ $A_r \cup \{[st]\}$ is in Γ. Since $A_n \cup \{u_n\} \subseteq A_r \cup \{[st]\}$, we have $[st] = u_n \in A_{n+1} \subseteq E$.

$\mathcal{E}_\varepsilon$ Assume $\varepsilon_\sigma s$ is accessible in E. Then there is some accessibility context C such that $C[\varepsilon_\sigma s]$ is in E. Let n be such that u_n is $C[\varepsilon_\sigma s]$. Let $r \geq n$ be such that u_n is in A_r. By the definition of A_{n+1} either $[s(\varepsilon s)]$ is in A_{n+1} or $[\forall x.\neg(sx)]$ is in A_{n+1} for some $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$. In the first case we are done. In the second case let $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$ be such that $[\forall x.\neg(sx)]$ is in E. Let $t \in \mathcal{U}_\sigma^E$ be given. By Proposition 3(3) we know $\neg[st]$ is in E.

Using the extension lemma we can lift the model existence theorem for evident sets to a model existence theorem for abstract consistency classes.

Theorem 2 (Model Existence). *Let Γ be an abstract consistency class. Every $A \in \Gamma$ is satisfiable.*

Proof. Let $A \in \Gamma$ be given. By Lemma 12 there is an evident set E such that $A \subseteq E$. By Theorem 1 E is satisfiable.

We can now prove completeness of the tableau calculus $\mathcal{T}$. Let $\Gamma_\mathcal{T}$ be the set of all branches A which are not refutable. We will first prove $\Gamma_\mathcal{T}$ is an abstract consistency class and then use Model Existence to prove completeness.

Lemma 13. *$\Gamma_\mathcal{T}$ is an abstract consistency class.*

Proof. It is easy to check each condition in Figure 3 using the corresponding tableau rule in $\mathcal{T}$. For example, we check $\mathcal{C}_\varepsilon$. Suppose $\varepsilon_\sigma s$ is accessible in A, $A \cup \{[s(\varepsilon s)]\}$ is not in $\Gamma_\mathcal{T}$ and $A \cup \{[\forall x.\neg(sx)]\}$ is not in $\Gamma_\mathcal{T}$ for every $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$. Choose some $x \in \mathcal{V}_\sigma \setminus \mathcal{V}s$. We know $A \cup \{[s(\varepsilon s)]\}$ and $A \cup \{[\forall x.\neg(sx)]\}$ are refutable. Hence A is refutable using $\mathcal{C}_\varepsilon$, contradicting $A \in \Gamma_\mathcal{T}$.

Completeness now follows directly from Lemma 13 and Theorem 2.

Theorem 3 (Completeness). *Let A be a branch. If A is not refutable, then A is satisfiable.*

6 Related Work

This work is an extension of two lines of research. First, we have extended the tableau calculus of Brown and Smolka [8] to support a choice operator at every type. We have done this by modifying sequent rules given by Mints [15] to be tableau rules and adapting the relevant parts of his cut-elimination proof. Second, we have obtained tighter restrictions on the instantiations of quantifiers than were available before.

In [9] Brown and Smolka give a complete tableau calculus for a first-order subsystem (EFO) of higher-order logic. Quantifiers are only allowed at type ι there and the instantiations are restricted to discriminating terms. We have maintained this restriction on instantiations for quantifiers at type ι. In addition we have proven that it is enough to instantiate quantifiers at type o with the two terms $\bot$ and $\neg\bot$. As for quantifiers at function types, we have proven that these instantiations need not consider variables that do not already occur free on the branch.

The choice rule given in this paper is similar to a ε-rule for a sequent calculus given by Mints [15]. We briefly sketch a comparison between our rules and the rules of Mints.

Translating into our language, Mints' rule could be represented as

$$(\textsc{Mints' } \varepsilon)\ \frac{}{[\neg(st)] \mid [s(\varepsilon s)]}\ \varepsilon s \text{ occurs on the branch}$$

By εs occurs on the branch we simply mean that εs appears as any subterm where none of the free variables of s are captured by a λ-binder. Note that this rule could apply more often than our $\mathcal{T}_\varepsilon$ rule. Our $\mathcal{T}_\varepsilon$ rule cannot be applied until εs appears on the branch in one of the forms $\varepsilon st_1 \cdots t_n$, $\neg(\varepsilon st_1 \cdots t_n)$, $(\varepsilon st_1 \cdots t_n) \neq_\iota u$ or $u \neq_\iota (\varepsilon st_1 \cdots t_n)$. Furthermore, in Mints' system the ε-rule would need to be applied for each new instantiation term t. In practice this could lead to the need to refute branches with $[s(\varepsilon s)]$ multiple times. We have avoided this by using the quantified formula $[\forall x.\neg(sx)]$ on the left branch.

Mints also includes an ε-extensionality rule in [15]. In our context, his rule could be realized as

$$(\textsc{Mints' ext } \varepsilon)\ \frac{}{s \neq t \ \mid\ (\varepsilon s) = (\varepsilon t)}\ \varepsilon_\sigma s \text{ and } \varepsilon_\sigma t \text{ occur on the branch}$$

In words, whenever $\varepsilon_\sigma s$ and $\varepsilon_\sigma t$ both occur on the branch, we must consider the case where s and t are different, and the case where εs and εt are the same. This rule could be highly branching in practice. When n different terms of the form εs occur on the branch, then the rule must be applied $\frac{n^2-n}{2}$ times. Furthermore, it has the disadvantage that it adds a positive equation to the branch. If σ is a function type, this will lead to the need to perform instantiations. We were able to omit such a rule entirely from our system and still prove completeness. It seems that Mints needed such a rule because the extensionality in [15] is not liberal enough. Translated into our context, the extensionality rule in [15] includes the rule

$$(\textsc{Special Case of Mints' extensionality})\ \frac{\varepsilon s s_1 \ldots s_n\ ,\ \neg\varepsilon s t_1 \ldots t_n}{s_1 \neq t_1 \mid \cdots \mid s_n \neq t_n}\ \ n \geq 1$$

This corresponds to our mating rule, except that we have liberalized the rule to include the case when the corresponding first arguments of ε are different.

$$(\textsc{Special Case of } \mathcal{T}_{\textsc{mat}})\ \frac{\varepsilon s_1 \ldots s_n\ ,\ \neg\varepsilon t_1 \ldots t_n}{s_1 \neq t_1 \mid \cdots \mid s_n \neq t_n}\ \ n \geq 1$$

7 Conclusion

We have presented a cut-free tableau calculus for Church's simple type theory with a choice operator. The calculus is designed with automated proof search in mind. In particular, only accessible terms on the branch need to be considered in order to apply a rule. Furthermore, instantiation terms are restricted according to the type and the formulas on the branch. At type o only instantiations corresponding to true and false are considered. At the base type ι only discriminating terms on the branch need to be considered (except when there are no discriminating terms in which case a default element can be used). Note that this means only finitely many instantiations at type ι need to be considered at each stage of the search. At function types, the set of instantiations is infinite, but we have at least proven that we do not need to consider instantiations with free variables that do not occur on the current branch.

The first author has extended this work by also considering description operators and if-then-else operators in addition to choice operators. This work has been reported in his Master's thesis [4]. The same style of rules (restricted to accessible terms) and model construction (using discriminants and possible values) can be used to incorporate description and if-then-else. Interpreting if-then-else is straightforward. Interpreting description is analogous to the interpretation of choice given here.

The second author has implemented a new higher-order automated theorem prover, Satallax, based on the ground calculus in this paper. Early results show the implementation to be competitive with the automated theorem provers TPS [3] and LEO-II [7] as well as the automated features of Isabelle [16].

References

1. Peter B. Andrews. Resolution in type theory. *J. Symb. Log.*, 36:414–432, 1971.
2. Peter B. Andrews. General models and extensionality. *J. Symb. Log.*, 37:395–397, 1972.
3. Peter B. Andrews and Chad E. Brown. TPS: A hybrid automatic-interactive system for developing proofs. *Journal of Applied Logic*, 4(4):367–395, 2006.
4. Julian Backes. Tableaux for higher-order logic with if-then-else, description and choice. Master's thesis, Universität des Saarlandes, 2010.
5. Julian Backes and Chad E. Brown. Analytic tableaux for higher-order logic with choice. Technical report, Programming Systems Lab, Saarland University, Saarbrücken, Germany, Jan 2010.
6. Christoph Benzmüller, Chad E. Brown, and Michael Kohlhase. Higher-order semantics and extensionality. *J. Symb. Log.*, 69:1027–1088, 2004.
7. Christoph Benzmüller, Frank Theiss, Larry Paulson, and Arnaud Fietzke. LEO-II — A cooperative automatic theorem prover for higher-order logic. In *Fourth International Joint Conference on Automated Reasoning (IJCAR'08)*, volume 5195 of *LNAI*. Springer, 2008.
8. Chad E. Brown and Gert Smolka. Analytic tableaux for simple type theory and its first-order fragment. Technical report, Programming Systems Lab, Saarland University, Saarbrücken, Germany, Dec 2009. accepted for publication by Logical Methods in Computer Science.
9. Chad E. Brown and Gert Smolka. Extended first-order logic. In Tobias Nipkow and Christian Urban, editor, *TPHOLs 2009*, volume 5674 of *LNCS*. Springer LNCS 5674, August 2009.
10. Chad E. Brown and Gert Smolka. Terminating tableaux for the basic fragment of simple type theory. In M. Giese and A. Waaler, editors, *TABLEAUX 2009*, volume 5607 of *LNCS (LNAI)*, pages 138–151. Springer, 2009.
11. Alonzo Church. A formulation of the simple theory of types. *J. Symb. Log.*, 5:56–68, 1940.
12. M.J. Gordon and T.F. Melham. *Introduction to HOL: A Theorem-Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.
13. Leon Henkin. Completeness in the theory of types. *J. Symb. Log.*, 15:81–91, 1950.
14. Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic*. PhD thesis, Case Western Reserve University, 1972.
15. G. Mints. Cut-elimination for simple type theory with an axiom of choice. *J. Symb. Log.*, 64(2):479–485, 1999.
16. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.
17. Raymond M. Smullyan. *First-Order Logic*. Springer, 1968.
18. Geoff Sutcliffe, Christoph Benzmüller, Chad E. Brown, and Frank Theiss. Progress in the development of automated theorem proving for higher-order logic. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2009.