Saarland University
Faculty of Natural Sciences and Technology I
Computer Science Department

Master's Thesis

Tableaux for Higher-Order Logic with If-Then-Else, Description and Choice

submitted by
Julian Backes

submitted
March 30, 2010

Supervisor	Prof. Dr. Gert Smolka
Advisor	Dr. Chad E. Brown
Reviewers	Prof. Dr. Gert Smolka
	Dr. Chad E. Brown

Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, March 30, 2010

Julian Backes

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, March 30, 2010

Julian Backes

Acknowledgement

I would to like thank Prof. Dr. Gert Smolka for giving me the opportunity to do my Master's thesis at the Programming Systems Lab.

I also want to thank Dr. Chad E. Brown for being my advisor during the past twelve months. His motivations and his endless patience have always given me the feeling that he enjoyed working with me. The result is not only this thesis but also my first scientific publication which I am very proud of. The past year has really been the most interesting and beneficial time of my studies.

I thank Jan André, Bianca Grün, Aileen Leinenbach, Stefan Lorenz, Björn Kunz, Kim Pecina and all other people who I may call my friends and who have always been supporting me. I want to thank my girlfriend Sarah Cieslik for her love and for her patience with me.

Finally, I want to thank my family, and especially my parents. They made my education possible which is probably more valuable for my future than anything else.

Abstract

In 2009, Brown and Smolka published four papers about different fragments of higher-order logic and presented corresponding complete tableau systems. These fragments support different well-known logical constants, such as conjunction, disjunction, implication and quantifiers.

This thesis is focused on the integration of new logical constants, namely if-then-else, description and choice. To motivate this goal, we give several independence results: We show that if-then-else is independent of classical simply typed higher-order logic and that description is independent of classical simply typed higher-order logic including if-then-else.

We present a generic tableau system. That is, for any set of logical constants, we show how to construct a cut-free tableau system. We prove completeness with respect to general models. In addition, we investigate in which cases we can achieve completeness with respect to standard models.

Contents

1 Introduction

1.1 Tableaux

Tableau systems are proof systems based on refutation: Instead of proving the validity of a formula they show that its negation is unsatisfiable. Tableaux were introduced independently by Beth [5] and Hintikka [17] in 1955 and later elaborated by Prawitz [22], Smullyan [24] and Fitting [13]. Today, many different tableau systems exist, for example for extended first-order logic [9] or higher-order logic [18] [8]. Tableaux are described by a set of *tableau rules* which look as follows:

$$\frac{A}{A_1 \mid \dots \mid A_n}$$

A and $A_1, \dots, A_n$ (for $n \geq 1$) are sets of formulas. A is the set of *premisses* and the A_i are the *alternatives*. Such a rule can be read as "If A is satisfiable then A_i is satisfiable for at least one $1 \leq i \leq n$" or as "If A is unsatisfiable then A_i is unsatisfiable for all $1 \leq i \leq n$". As an example, consider the following rule for disjunction:

$$\frac{s \vee t}{s \mid t}$$

The meaning of this rule is "If $s \vee t$ is satisfiable then either s is satisfiable or t is satisfiable".

Tableaux rules are applied to sets of formulas which we call *branches*. A rule can only be applied if the the branch contains all premisses. The result of this application is one new branch per alternative containing all the formulas from the original branch plus the corresponding alternative. There are special rules called *closed rules* which have no alternative.

$$\frac{A}{\quad}$$

A branch A is *closed* (or *refutable*) iff either a closed rule applies to A or the application of a rule to A yields branches $A_1, \dots, A_n$ which can all be closed. Given a certain formula s, we will prove the validity of s using a tableau system by showing that $\{\neg s\}$ is refutable in this tableau system, i.e., that $\neg s$ unsatisfiable.

1.2 If-Then-Else, Description & Choice

The main topic of this thesis will be about the three logical constants *if-then-else*, *description* and *choice* which we will introduce in this section. We now present them and discuss why it is interesting to analyze them.

1.2.1 If-Then-Else

Although if-then-else is a very common control structure in almost any programming language there has not been much research going on in the area of automated higher-order proof systems with support for if-then-else. In [4], Beeson presents a unification algorithm based on if-then-else. In [1], the authors use decision trees to give a completeness result. Decision trees can be represented using nested if-then-else expressions.

If-Then-Else is a function getting three arguments: one boolean value (true or false) and two values of the same type corresponding to the if-branch respectively to the else-branch. The following two axioms define the properties of if-then-else. We use *if* to refer to the if-then-else operator and use $\top/\bot$ to represent true/false:

$$\begin{aligned} \forall x, y.\ \mathit{if} \top xy &= x \\ \forall x, y.\ \mathit{if} \bot xy &= y \end{aligned}$$

Note that, as for other well-known logical constants like disjunction or negation, there is exactly one function satisfying these axioms. The interesting difference between if-then-else and other logical constants is that if-then-else does not necessarily return something boolean. For example $\mathit{if}\top(\lambda x.\bot)(\lambda x.\top)$ will evaluate to $\lambda x.\bot$ which is clearly a function.

1.2.2 Choice

The Axiom of Choice was first stated in 1904 by Zermelo [26]. He used it to prove that every set can be well-ordered. In 1908, Zermelo made this axiom part of his famous "Untersuchungen über die Grundlagen der Mengenlehre" [28] where he presented the first axiomatization of a set theory. Later, Zermelo's axioms were extended to form Zermelo-Fraenkel set theory with choice (ZFC) on which most of modern mathematics is based. The Axiom of Choice states that there (for any set A of sets) there is is a function which, for any nonempty set (in A). Formally (suppressing the A):

$$\exists f.\ \forall p.\ (\exists x.\ px) \rightarrow p(fp)$$

The Axiom of Choice has always been very controversial. Already after Zermelo's publication in 1904 there had been objections against it by several people (in [27], Zermelo mentions Borel and Peano). This not only led to a new proof of the well-ordering theorem which made again use of a choice operator [27] but, in addition, he spent one section of this paper on the discussion of choice and defended his decision to assume the existence of such a function.

Today, the existence of choice has widely been accepted. For instance, the interactive proof system Isabelle/HOL [21] assumes the Axiom of Choice. An interesting fact about choice is that there are several possible candidates. For example the set $\{0, 1\}$ has the subsets $\{0, 1\}$, $\{0\}$, $\{1\}$ and $\emptyset$. Already for the empty set, there are two possible elements for choice to return. The same holds for $\{0, 1\}$ which results in four possible choice operators. One important part of this thesis will be to give a specific definition of choice in corresponding situations.

1.2.3 Description

The theory of descriptions originated from the areas of philosophy and linguistics. It was introduced by Russell in 1905 [23] although Frege had his own version already in 1893 [16]. The existence of a description function can be expressed using the following axiom:

$$\exists f.\ \forall p.\ (\exists! x.\ px) \rightarrow p(fp)$$

Although this looks very similar to the Axiom of Choice there is an important difference. The behaviour of a description function is only defined on singleton sets, i.e., sets which only contain one element.

We will later show that description functions are in a sense weaker than choice functions. That's why we want to investigate whether adding support for description affects the properties of a tableau system in a different way than adding support for choice does.

1.3 Related Work

This thesis builds upon four papers by Brown and Smolka presenting tableau systems for different fragments of simply typed higher-order logic:

- *Terminating Tableaux for the Basic Fragment of Simple Type Theory* [11] presents a fragment which does not support lambda abstracts, has no equations at functional types (although it has disequations at any type) and no quantifiers. The corresponding tableau system is terminating, cut-free and complete with respect to standard models.
- *Extended First-Order Logic* [9] supports lambda abstracts and has quantifiers at base types. The tableau system is not terminating but still cut-free and complete with respect to standard models.
- *Complete Cut-Free Tableaux for Equational Simple Type Theory* [8] covers the full higher-order logic, including lambda abstracts and all well-known logical constants. The proof system is not terminating but – as the title suggests – still cut-free and complete with respect to non-standard models.
- *Analytic Tableaux for Simple Type Theory and its First-Order Fragment* [7] essentially combines the three papers from above and tries to present them in a more modular way.

We will add new logical constants to the second and third fragment while maintaining existing properties (cut-freeness, completeness) of the corresponding proof systems. As already mentioned in the last section, one of these logical constants will be choice. In 1999, Mints extended a proof system based on a sequent calculus by Takeuti [25] with choice and gave a corresponding cut-elimination proof [20]. His new rules inspired our work, although we made some changes and improvements.

1.4 Structure of the Thesis

Chapter 2 introduces all basic definitions which are needed to understand the subsequent chapters. This mainly includes syntax and semantics of Church's simply typed lambda calculus [12]. We extend it by making almost all definitions depend on a *signature* (a set of logical constants).

Chapter 3 will motivate why it makes sense to add support for if-then-else, description and choice to a tableau system. We will develop the notion of a *closure* of a signature. This allows to show that certain logical constants are not implied by other logical constants (for a certain definition of "implies").

Chapter 4 and Chapter 5 are the most important parts of this thesis. We will present a general tableau system and give a general completeness result. That is, for any possible signature, we show how to construct a tableau system which is complete relative to this signature. We also show under which conditions we can achieve completeness with respect to standard models.

Chapter 6 contains several small examples. They show how the tableau rules for the new logical constants work in practice.

In Chapter 7, we will show how the tableau systems can further be improved, for example by adding even more logical constants or by restricting certain tableau rules. This chapter does not always give proofs for the claims we make. In addition, many questions are left open. Consequently, Chapter 7 can be seen as a collection of ideas for future work.

Chaper 8 concludes this thesis. First, we will give a short recap of the main results of this work. Together with a reference to Chapter 7, we will give an outlook on how our results can serve as an approach to new ideas.

2 Basic Definitions

In this chapter, we will define the basic notations and notions which we will need in the subsequent chapters. Most definitions here are taken from [11], [9] and [8].

2.1 Types & Terms

We assume a countable set of base types (β). Types (σ, τ, μ) are defined inductively: every base type is a type; if σ and τ are types then $\sigma\tau$ is a type. Although all results in this thesis generalize to arbitrarily many base types, we fix this set to be $\{o, \iota\}$. Purely propositional types are types consisting only of o: o is a purely propositional type. If σ and τ is a purely propositional type then $\sigma\tau$ is a purely propositional type. Omitted parenthesis in types associate to the right. For example $\iota\iota\iota$ corresponds to $\iota(\iota\iota)$.

We assume a countably infinite set $\mathcal{N}$ of *names* (ν). Every name comes with a unique type and for every type, there are infinitely many names. We write $n : \sigma$ to denote the name n of type σ and omit the type when possible. We partition $\mathcal{N}$ into two countably infinite subsets $\mathcal{V} \subsetneq \mathcal{N}$, the *variables* ($x$), and $\mathcal{LC} \subsetneq \mathcal{N}$, the *logical constants* (c).

$\mathcal{LC}$ is defined by the constants in Figure 2.1. Note that this set contains infinitely many elements because we have, for instance, one constant $\exists_\sigma$ for each type σ.

Given a set $S \subseteq \mathcal{LC}$ (*signature*), *typed $\mathcal{S}$-terms* (s, t, u, v, w) are defined inductively: Every variable is an $\mathcal{S}$-term and every logical constant in $\mathcal{S}$ is an $\mathcal{S}$-term; if s is an $\mathcal{S}$-term of type $\sigma\tau$ and t is an $\mathcal{S}$-term of type σ, st is an $\mathcal{S}$-term of type τ; if x is a variable of type σ and s is an $\mathcal{S}$-term of type τ then $\lambda x.\ s$ is an $\mathcal{S}$-term of type $\sigma\tau$. Sometimes, we write $\lambda x : \sigma.\ s$ to make the type of x clear, if necessary. We write $s : \sigma$ to say that s is a term of type σ. We write $\Lambda_\sigma^{\mathcal{S}}$ for the set of all $\mathcal{S}$-terms of type σ. An $\mathcal{S}$-term of type o is called *$\mathcal{S}$-formula*. We use *wff*(S) to denote the set of all $\mathcal{S}$-formulas. Applications associate to the left: stu means $(st)u$. We have the exception that $\neg st$ always means $\neg(st)$.

We fix the special signature $\mathcal{LC}_{hol} = \{\top, \bot, \neg, \vee, \exists_\sigma, =_\sigma\}$, that is the signature which defines *full simply typed higher-order logic*.

To simplify matters, we will combine multiple lambdas, i.e., we will write $\lambda x, y.\ s$ instead of $\lambda x.\ \lambda y.\ s$. For most logical constants, we will use infix notation. For example, we will write $s = t$ instead of $= st$. Moreover, we will write $\exists x.\ s$ instead of $\exists(\lambda x.\ s)$. We use $s \neq t$ as a shorthand for $\neg(s = t)$. Terms of this kind are called *disequations*.

An *$\mathcal{S}$-context* (C) is an $\mathcal{S}$-term with a hole. $\mathcal{S}$-contexts are defined inductively: $[]_\sigma$ is an $\mathcal{S}$-context of type σ; if s is an $\mathcal{S}$-term of type $\sigma\tau$ and C an $\mathcal{S}$-context of type σ then sC is an $\mathcal{S}$-context of type τ; if C is an $\mathcal{S}$-context of type $\sigma\tau$ and s is an $\mathcal{S}$-term of type σ then Cs is an $\mathcal{S}$-context of type τ; if x is a name of type σ and C is an $\mathcal{S}$-context of type τ then $\lambda x.\ C$ is an $\mathcal{S}$-context of type σ.

Constant c	Predicate $\mathcal{P}_c$
$\bot : o$	$\mathcal{P}_\bot(v) = (v = 0)$
$\top : o$	$\mathcal{P}_\top(v) = (v = 1)$
$\neg : oo$	$\mathcal{P}_\neg(f) = (f1 = 0 \wedge f0 = 1)$
$\vee : ooo$	$\mathcal{P}_\vee(f) = \forall a, b \in \mathcal{D}o.$ if $a = 0$ and $b = 0$ then $fab = 0$ else $fab = 1$
$\exists_\sigma : (\sigma o)o$	$\mathcal{P}_{\exists_\sigma}(f) = \forall g \in \mathcal{D}(\sigma o).$ if $g \neq (\lambda a \in \mathcal{D}\sigma.\ 0)$ then $fg = 1$ else $fg = 0$
$=_\sigma: \sigma\sigma o$	$\mathcal{P}_{=_\sigma}(f) = \forall a, b \in \mathcal{D}\sigma.$ if $a = b$ then $fab = 1$ else $fab = 0$
if$_\sigma : o\sigma\sigma\sigma$	$\mathcal{P}_{\mathit{if}}(f) = \forall a \in \mathcal{D}o, b, c \in \mathcal{D}\sigma.$ if $a = 1$ then $fabc = b$ else $fabc = c$
$\iota_\sigma : (\sigma o)\sigma$	$\mathcal{P}_\iota(f) = \forall g \in \mathcal{D}(\sigma o).\ (\exists! a \in \mathcal{D}\sigma.\ ga) \rightarrow g(fg)$
$\varepsilon_\sigma : (\sigma o)\sigma$	$\mathcal{P}_\varepsilon(f) = \forall g \in \mathcal{D}(\sigma o).\ (\exists a \in \mathcal{D}\sigma.\ ga) \rightarrow g(fg)$

Figure 2.1: A list of all logical constants defining $\mathcal{LC}$ together with their properties

Let C' be an $\mathcal{S}$-context of type σ which has a hole of type τ. We can apply C' to $\mathcal{S}$-terms t of type τ to get an $\mathcal{S}$-term of type σ:

$$\begin{aligned}
[\,][t] &:= t \\
(C\ s)[t] &:= C[t]\ s \\
(s\ C)[t] &:= s\ C[t] \\
(\lambda x.\ C)[t] &:= \lambda x.\ C[t]
\end{aligned}$$

Let t be an $\mathcal{S}$-term. The *free variables* of t, written $\mathcal{V}t$, are defined by induction on $\mathcal{S}$-terms:

$$\begin{aligned}
\mathcal{V}c &:= \emptyset \\
\mathcal{V}x &:= \{x\} \\
\mathcal{V}(uv) &:= \mathcal{V}u \cup \mathcal{V}v \\
\mathcal{V}(\lambda x.\ u) &:= \mathcal{V}u \setminus \{x\}
\end{aligned}$$

An $\mathcal{S}$-term t is *closed* iff $\mathcal{V}t = \emptyset$.

2.2 Frames, Interpretations & Satisfiability

In the last section, we defined the syntax for types and terms. In this section, we want to give meaning to these things: A frame $\mathcal{D}$ is a function mapping types to nonempty sets such that

- $\mathcal{D}o \subseteq \{0, 1\}$
- $\mathcal{D}(\sigma\tau)$ is a set of total functions from $\mathcal{D}\sigma$ to $\mathcal{D}\tau$

$\mathcal{D}o$ represents a nonempty set of truth values containing 0 (*false*) or 1 (*true*) or both. A frame is *standard* if $\mathcal{D}(\sigma\tau)$ is the set of all total functions from $\mathcal{D}\sigma$ to $\mathcal{D}\tau$ and if $\mathcal{D}o = \{0, 1\}$.

We say that a frame *realizes* a logical constant $c : \sigma$ iff there is an object $v \in \mathcal{D}\sigma$ such that $\mathcal{P}_c(v)$ is true. Given a signature $\mathcal{S}$, an $\mathcal{S}$-frame is a frame which realizes all logical constants in $\mathcal{S}$. Note that a standard frame is trivially an $\mathcal{S}$-frame for any signature $\mathcal{S}$.

An $\mathcal{S}$*-assignment* into an $\mathcal{S}$-frame $\mathcal{D}$ is a function $\mathcal{I}$ that maps every variable $x : \sigma$ to an element of $\mathcal{D}\sigma$ and every logical constant $c : \sigma$ in $\mathcal{S}$ to an element $v \in \mathcal{D}\sigma$ such that $\mathcal{P}_c(v)$ is true. A *standard assignment* is an assignment into a standard frame. We write $\mathcal{I}^x_a$ to denote the assignment that is like $\mathcal{I}$ but maps x to a. Given an interpretation $\mathcal{I}$, we define the evaluation function $\hat{\mathcal{I}}$ inductively on $\mathcal{S}$-terms:

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

An *interpretation* is an assignment whose evaluation function is defined on all $\mathcal{S}$-terms. A frame is *combinatory* iff there is an interpretation into that frame. Since it is not interesting to consider non-combinatory frames, we will only consider combinatory frames from now on. We omit the word *combinatory* and only mention it if it has a special relevance. We use $\mathcal{F}(\mathcal{S})$ to refer to the set of all (combinatory) $\mathcal{S}$-frames.

We say that an $\mathcal{S}$-interpretation $\mathcal{I}$ *satisfies* an $\mathcal{S}$-formula s iff $\hat{\mathcal{I}}(s) = 1$. An $\mathcal{S}$-formula is $\mathcal{S}$*-satisfiable* iff there is an $\mathcal{S}$-interpretation $\mathcal{I}$ such that $\hat{\mathcal{I}}(s) = 1$. An $\mathcal{S}$-formula is $\mathcal{S}$*-valid* iff it is satisfied by all $\mathcal{S}$-interpretations.

We are also interested in satisfiability/validity with respect to standard frames: An $\mathcal{S}$-formula is $\mathcal{S}$-satisfiable with respect to standard frames iff there is a standard $\mathcal{S}$-interpretation $\mathcal{I}$ that satisfies s. An $\mathcal{S}$-formula is $\mathcal{S}$-valid with respect to standard frames iff it is satisfied by all standard $\mathcal{S}$-interpretations.

A set of $\mathcal{S}$-formulas is satisfiable (with respect to standard frames) iff there is a (standard) interpretation which satisifes all formulas in this set.

2.3 Normalization & Substitution

Given a signature $\mathcal{S}$, we assume a type preserving and total *normalization operator* $[\cdot]$ from $\mathcal{S}$-terms to $\mathcal{S}$-terms. A term is *normal* iff $[s] = s$. Instead of committing to a specific operator such as β or long-β, we require the following properties:

N1 $[[s]] = [s]$
N2 $[[s]t] = [st]$
N3 $[\nu s_1 \dots s_m] = \nu[s_1] \dots [s_m]$ if $ns_1 \dots s_m : \beta$ for some base type β and $m \geq 0$
N4 $\hat{\mathcal{I}}[s] = \hat{\mathcal{I}}s$

$\mathcal{S}$-substitutions (θ) are partial functions from names to $\mathcal{S}$-terms of the same type such that $c \notin \operatorname{Dom}\theta$ for all $c \in \mathcal{S}$. We write θ^x_s to denote the substitution that is like θ but maps x to s. $\emptyset$ is the substitution that is undefined on all names. We can lift a substitution θ to a function $\hat{\theta}$ from $\mathcal{S}$-terms to $\mathcal{S}$-terms. Again, we do not give a concrete definition but require the following properties:

S1 $\hat{\theta}\nu =$ if $\nu \in \operatorname{Dom}\theta$ then $\theta\nu$ else ν
S2 $\hat{\theta}(st) = (\hat{\theta}s)(\hat{\theta}t)$
S3 $[(\hat{\theta}(\lambda x.\ s))t] = [\widehat{\theta^x_t}s]$
S4 $[\hat{\emptyset}s] = [s]$

Note that we will sometimes omit the signature and talk for example only about terms, formulas or satisfiability. In such cases, the signature should be clear from the context or unimportant.

3 Independence Results

3.1 Closure

In this thesis, we want to analyze whether certain properties of existing tableau systems can be preserved when adding support for new logical constants. First, we need to answer one important question: Why do we want to add new logical constants? Can we express more things? What does it mean to *express something*?

Assume we are in the context of simply typed lambda calculus as defined in the last chapter, together with a signature $\mathcal{S}_1$ that contains $=_o$ (equivalence) and $\bot$ (bottom). Does it make sense to add $\neg$ (negation) to $\mathcal{S}_1$ (yielding a signature $\mathcal{S}_2$) in order to get more expressivity?

Intuitively, it should not make sense because we can replace any occurence of $\neg$ by $\lambda x.\ x =_o \bot$, we can *express* negation. What happens when we add $\neg$? Given the definitions from the last chapter, mainly one thing changes: We explicitly force the frames which we are considering to contain the negation function. Does this really affect the frames in this case?

The answer is "no". Any $\mathcal{S}_1$-interpretation will have to interpret the lambda abstract $\lambda x.\ x =_o \bot$ as the negation function because there are no free variables and the interpretation of each logical constant is fixed. As a consequence, *all* $\mathcal{S}_1$-frames will also contain the negation function because we only consider combinatory frames. Explicitly adding negation has no impact because the frames already realized it before. This leads to the definition of a *closure* on signatures: Let $\mathcal{S}$ be a signature. The *closure of* $\mathcal{S}$ is the set

$$\overline{\mathcal{S}} = \{c \in \mathcal{LC} \mid \forall \mathcal{D} \in \mathcal{F}(\mathcal{S}).\ \mathcal{D} \text{ realizes } c\}$$

The closure of a signature contains all logical constants which are realized by all corresponding frames. We say that a signature $\mathcal{S}$ *implies* a logical constant c iff $c \in \overline{\mathcal{S}}$. We use this notion ambiguously and say that a logical constant c_1 implies another logical constant c_2 relative to a signature $\mathcal{S}$ iff $c_2 \in \overline{\mathcal{S} \cup \{c_1\}}$

Note that there are many other possible ways to define "implies". For example, we could consider signatures $\mathcal{S}$ and $\mathcal{S} \cup \{c\}$ and compare $\mathcal{F}(\mathcal{S})$ and $\mathcal{F}(\mathcal{S} \cup \{c\})$. Alternatively, we could compare certain sets of valid formulas induced by two signatures because the set of frames (and hence the set of interpretations) possibly gets smaller when adding new logical constants. However, since all these ideas cause other problems and do not give us any advantage in the context of this thesis, we decided to define "implies" in terms of the closure of signatures.

3.2 Ordering Logical Constants

From now on and for the rest of this chapter, we assume full simply typed higher-order logic, that is the signature $\mathcal{LC}_{\mathit{hol}}$. In this section, we will start to bring order to if-then-else, description and choice in terms of implications among them. One obvious observation is: ε_σ implies ι_σ, i.e., given a choice operator at type σ, we also have a description operator at that type. The corresponding term for description is ε_σ. Also, ι_σ implies if_σ. This fact is not that easy to see. The following term does the job (we write $s \wedge t$ here as an abbreviation for $\neg(\neg s \vee \neg t)$:

$$\lambda x : o, y : \sigma, z : \sigma.\ \iota_\sigma(\lambda u : \sigma.(x \wedge y = u) \vee (\neg x \wedge z = u))$$

These two facts yield one possible order: "Choice implies description implies if-then-else". For this order to be useful, the other direction should not hold, i.e., choice should not be implied by description and description should not be implied by if-then-else. In addition, simply typed higher-order logic should not imply if-then-else.

Already in 1972, Andrews solved one of these problems [2]: He proved that ι_ι does not imply ε_ι, i.e., that choice is really more powerful than description. He used a technique which was introduced by Fraenkel in 1922 [15] [14] and later refined and extended by Lindenbaum and Mostowski in 1938 [19].

In the next two sections, we will solve the two remaining problems and show that the order proposed above indeed holds. The idea is to construct nonstandard combinatory frames which realize for example all logical constants in $\mathcal{LC}_{\mathit{hol}}$ but not *if*. As a consequence, the related closure will not contain *if* either.

These constructions will be based on so-called *logical relation frames*: Given definitions for all base types, and given some specific relations on them, the relations and the definition for the functional types follow by an inductive definition. The definitions for the base types together with the definitions for the functional types then form a frame. Very detailed information about such frame constructions can be found in [6]; we will only present the most important lemmas here and refer to this book for the corresponding proofs.

As already said, we want to construct nonstandard combinatory frames using logical relations. In this case, we express these relations as follows: Given a frame $\mathcal{D}$, we define an index set A of size i and represent an i-ary tuple on some $\mathcal{D}\sigma$ as a function p from A to $\mathcal{D}\sigma$ (which corresponds to a tuple $\langle p(j)\rangle_{j\in A}$). A relation $\mathcal{R}$ on $\mathcal{D}\sigma$ is represented as a set of such functions. For example, the equality relation on the natural numbers can be defined as follows: Equality, as we normally use it, is a binary relation. One obvious index set is $A = \{1, 2\}$. Given the frame $\mathcal{D}\iota = \{0, 1, 2, 3, \dots\}$, the equality relation should contain the pairs $\langle 0, 0\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle, \dots$. This set can be represented by all constant functions from A to $\mathcal{D}\iota$, i.e., $\mathcal{R}_= = \{p \mid \exists n \in \mathcal{D}\iota.\ \forall j \in A.\ p(j) = n\}$.

Given an index set A, relations $\mathcal{R}_\beta$ for all base types β and a frame $\mathcal{D}$ which is defined for all base types, we define the frame and relations for functional types $\sigma\tau$ by induction. Assume $\mathcal{D}\sigma, \mathcal{D}\tau, \mathcal{R}_\sigma, \mathcal{R}_\tau$ are defined:

$$\begin{aligned}\mathcal{D}(\sigma\tau) &:= \{f : \mathcal{D}\sigma \to \mathcal{D}\tau \mid \forall p \in R_\sigma.\ (f \circ p) \in \mathcal{R}_\tau\}\\ R_{\sigma\tau} &:= \{q : A \to \mathcal{D}(\sigma\tau) \mid \forall p \in R_\sigma.\ S(q,p) \in \mathcal{R}_\tau\}\end{aligned}$$

where $S(q,p)(i) = q(i)(p(i))$. Since we want frames constructed this way to be combinatory, we will use the following lemma to enforce this property already at the time of construction:

Lemma 3.1. *Let $\mathcal{D}$ be a frame constructed using logical frames as described in this section. $\mathcal{D}$ is combinatory if the relations on the base types contain all constant functions.*

Proof. See Theorem 4.4.7 in [6]. □

The frames constructed here need to realize all logical constants in $\mathcal{LC}_{hol}$. To enforce this, we will always take $\mathcal{R}_o$ to be the full relation on $\mathcal{D}o$, i.e., $\mathcal{R}_o = (\mathcal{D}o)^A$. The types of these logical constants only consist of elements returning something of type o. For example $\exists : (\iota o)o$ takes an argument returning something of type o and returns something of type o itself. The conditions in the inductive definitions of $\mathcal{D}(\sigma\tau)$ and $\mathcal{R}_{\sigma\tau}$ only consider the relation on $\mathcal{D}\tau$ which is in this case $\mathcal{D}o$. Since we have the full relation on $\mathcal{D}o$, these conditions are trivially fulfilled. Hence, the relations on the types of the logical constants in $\mathcal{LC}_{hol}$ will be the full relations and the frame will realize all these constants.

3.3 If-Then-Else

3.3.1 Frame Construction

The goal in this section is to prove the following proposition:

Proposition 3.2. *Let $\mathcal{S} \subseteq \mathcal{LC}_{hol}$. $if_\iota \notin \overline{\mathcal{S}}$*

Proof. We need to construct a frame which realizes all logical constants in $\mathcal{LC}_{hol}$ but not if_ι. As described in the last section, the first condition can be easily fulfilled: We take $\mathcal{R}_o$ to be the full relation on $\mathcal{D}o$. This also ensures that we have all constant functions in $\mathcal{D}o$. The only remaining part is to give a definition of $\mathcal{D}\iota$ and a corresponding relation on that set such that if_ι is not realized by $\mathcal{D}$.

We follow our example from the last section: $\mathcal{D}\iota$ will be the set of the natural numbers, i.e., $\mathcal{D}\iota = \{0, 1, 2, \dots\}$. We take the index set A to be $\{1, 2\}$ and define R_ι to be equality on the natural numbers: $\mathcal{R}_\iota = \{p \mid \exists n \in \mathcal{D}\iota.\ \forall j \in A.\ p(j) = n\}$. This relation is exactly the set of all constant functions. Hence, by Lemma 3.1, the frame will be combinatory.

In order to show that if_ι is not realized by $\mathcal{D}$, we need to find functions $p \in \mathcal{R}_o, q \in \mathcal{R}_\iota, r \in \mathcal{R}_\iota$ such that $S(S(if \circ p, q), r) \notin \mathcal{R}_\iota$. The trick here is to take a non-constant p, for example $p \mathrel{\hat{=}} \langle 0, 1\rangle$. q and r can be arbitrary but they must be different, e.g. $q \mathrel{\hat{=}} \langle 0, 0\rangle$ and $r \mathrel{\hat{=}} \langle 1, 1\rangle$. In this case, $S(S(if \circ p, q), r) \mathrel{\hat{=}} \langle if\ 0\ 0\ 1, if\ 1\ 0\ 1\rangle = \langle 1, 0\rangle$. This is clearly not constant and hence not in $\mathcal{R}_\iota$. □

3.3.2 If-Then-Else at Higher Types

Another interesting fact is that having *if* at base types implies *if* at all types, as the following proposition states:

Proposition 3.3. *Let $\mathcal{S}$ be a signature such that $\mathcal{LC}_{hol} \subseteq \mathcal{S}$. If $if_\iota \in \overline{\mathcal{S}}$, then $if_\sigma \in \overline{\mathcal{S}}$ for any type σ.*

This proposition can be easily seen using the following two lemmas:

Lemma 3.4. *Let $\mathcal{S}$ be any signature. If $if_\beta \in \overline{\mathcal{S}}$ for any base type β, then $if_\sigma \in \overline{\mathcal{S}}$ for every type σ.*

Proof. We give an inductive construction of if_σ. The base cases follow by assumption. Let $\sigma = \mu\tau$. Consider the following term:

$$\lambda if : o\tau\tau\tau, x : o, f : \sigma, g : \sigma, y : \mu.\ if x(fy)(gy)$$

This term does not have any free variables so its interpretation is fixed: We will always get a function which, given if-then-else at type τ as argument, returns if-then-else at type σ. $\square$

Lemma 3.5. *Let $\mathcal{S}$ be a signature such that $\mathcal{LC}_{hol} \subseteq \mathcal{S}$. Then $if_\sigma \in \overline{\mathcal{S}}$ for any purely propositional type σ.*

Proof. We construct if_o using the following term:

$$\lambda x : o, y : o, z : o.\ \neg(\neg x \vee \neg y) \vee \neg(x \vee \neg z)$$

Starting from this term, we can construct if_σ for any purely propositional type using the same construction as in the previous lemma. $\square$

3.3.3 Swapping Functions and If-Then-Else

The frame constructed here says even more than "if-then-else is not implied by simply typed higher-order logic": The frame also contains *swapping functions* which are described by the following formula, called swap$_\sigma$:

$$\neg\exists x : \sigma, y : \sigma.\ \neg\exists f : \sigma\sigma.\ \neg(fx \neq y \vee fy \neq x)$$

To see this fact, we prove the following, even stronger, lemma:

Lemma 3.6. *Let $\mathcal{D}$ be the frame as constructed above. $\mathcal{D}(\iota\iota) = \mathcal{D}\iota \to \mathcal{D}\iota$, i.e., $\mathcal{D}(\iota\iota)$ contains all functions from $\mathcal{D}\iota$ to $\mathcal{D}\iota$.*

Proof. We need to show that, given a function $f \in \mathcal{D}\iota \to \mathcal{D}\iota$, $f \circ p \in \mathcal{R}_\iota$ for all $p \in \mathcal{R}_\iota$. Let $f \in \mathcal{D}\iota \to \mathcal{D}\iota$ and let $p \in \mathcal{R}_\iota$, i.e., $p \mathrel{\hat{=}} \langle a, a\rangle$ for some $a \in \mathcal{D}\iota$. $f \circ p \mathrel{\hat{=}} \langle fa, fa\rangle = \langle b, b\rangle$ for some $b \in \mathcal{D}\iota$. This pair is clearly constant and thus still in $\mathcal{R}_\iota$ $\square$

In [10], the authors give a model construction showing that there are frames which realize all logical constants in $\mathcal{LC}_{hol}$ but do not contain swapping functions. As a consequence, swap$_\sigma$ is not $\mathcal{LC}_{hol}$-valid. Although the frame constructed here contains swapping functions, it still does not realize *if*. In contrast to that, the following proposition holds:

Proposition 3.7. *Let $\mathcal{S}$ be a signature. If $\mathcal{LC}_{hol} \cup \{if_\sigma\} \subseteq \overline{\mathcal{S}}$, then* swap$_\sigma$ *is $\overline{\mathcal{S}}$-valid.*

Proof. We construct the swapping function at type σ:

$$\lambda x:\sigma, y:\sigma, z:\sigma.\ \mathit{if}(x = z)yx$$

Since we only consider combinatory frames, any frame realizing if_σ and $=_\sigma$ will also contain swapping functions at type σ. Hence, swap$_\sigma$ will be $\overline{\mathcal{S}}$-valid. □

3.4 Description

In [2], Andrews shows that ι_ι is not implied by $\mathcal{LC}_{hol}$. We give a stronger result in this section. We show that ι_ι is not implied by $\mathcal{LC}_{hol} \cup \{\mathit{if}_\iota\}$.

3.4.1 Frame Construction

The construction of a frame realizing all logical constants in $\mathcal{LC}_{hol}$ and if_ι but not ι_ι is not that straightforward. As in the last frame construction, we take $\mathcal{R}_o$ to be the full relation on $\{0,1\}$ and $\mathcal{D}\iota$ to be the the set of the natural numbers. It remains to find the corresponding relation on $\mathcal{D}\iota$. Taking equality as in Section 3.3 reveals a problem.

Equality is the "least" relation which fulfills the condition from Lemma 3.1 for a frame construction to result in a combinatory frame. Requiring if-then-else to be realized by the frame has the following effect: Let $p \mathrel{\hat{=}} \langle a, a\rangle$, $q \mathrel{\hat{=}} \langle b, b\rangle \in \mathcal{R}_\iota$. As we have the full relation on $\mathcal{R}_o$, we also need $\langle \mathit{if}1ab, \mathit{if}0ab\rangle = \langle a, b\rangle$ and $\langle \mathit{if}0ab, \mathit{if}1ab\rangle = \langle b, a\rangle$ to be in $\mathcal{R}_\iota$. This holds for any p and q which finally results in the full relation on $\mathcal{D}\iota$. The constructed frame will be a standard frame and therefore also realize description. This effect occurs with any relation of finite arity which fulfills the condition of Lemma 3.1. Hence, to avoid this situation, we need to find a relation of infinite arity. The solution is to take a relation of infinite arity which consists only of tuples containing a finite number of different elements:

$$\begin{array}{rcl} A & = & \mathbb{N} \\ \mathcal{R}_\iota & = & \{p \in A \to \mathbb{N} \mid \mathrm{Ran}\,(p) \text{ is finite}\} \end{array}$$

Obviously, $\mathcal{R}_\iota$ contains all constant functions. Hence, by Lemma 3.1, the constructed frame will be combinatory. The following proposition concludes this section:

Proposition 3.8. *Let $\mathcal{D}$ be the frame as constructed above. Then:*

1. *if_ι is realized by $\mathcal{D}$*

2. *ι_ι is not realized by $\mathcal{D}$*

Proof.

1. Let $p \in \mathcal{R}_o$, $q, r \in \mathcal{D}_\iota$. We need to show that $S(S(\mathit{if} \circ p, q), r) \in \mathcal{D}_\iota$, i.e., that $S(S(\mathit{if} \circ p, q), r) \in \mathcal{D}_\iota$ has a finite range. By the definition of *if*,

$$\operatorname{Ran}(S(S(\mathit{if} \circ p, q), r)) \subseteq \operatorname{Ran}(q) \cup \operatorname{Ran}(r)$$

   Since we know that $\operatorname{Ran}(q)$ as well as $\operatorname{Ran}(r)$ are finite, $\operatorname{Ran}(q) \cup \operatorname{Ran}(r)$ is finite, too. Hence, every subset of $\operatorname{Ran}(q) \cup \operatorname{Ran}(r)$ is finite and thus, $S(S(\mathit{if} \circ p, q), r) \in \mathcal{D}_\iota$.

2. We know that $\mathcal{R}_{\iota o}$ contains all functions from A to $\mathcal{D}_{\iota o}$, including the function p:

$$p(i)(j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

   p corresponds to the tuple containing all singleton sets, i.e., $p \mathrel{\hat{=}} \langle \{0\}, \{1\}, \{2\}, \ldots \rangle$. Assume $\iota \in \mathcal{D}((\iota o)\iota)$. Then $(\iota \circ p) \in \mathcal{R}_\iota$. By the definition of ι

$$(\iota \circ p) \mathrel{\hat{=}} \langle \iota\{0\}, \iota\{1\}, \iota\{2\}, \ldots \rangle = \langle 0, 1, 2 \ldots \rangle$$

   This tuple clearly has an infinite range. Contradiction. □

We now know that there is a frame which realizes all logical constants in $\mathcal{LC}_{\mathit{hol}} \cup \{\mathit{if}_\iota\}$ but not ι_ι. Hence, $\iota_\iota \notin \overline{\mathcal{LC}_{\mathit{hol}} \cup \{\mathit{if}_\iota\}}$ and thus if-then-else does not imply description.

3.4.2 Description at Higher Types

As for if-then-else, a description operator at type ι implies a description operator at all types:

Proposition 3.9. *Let $\mathcal{S}$ be a signature such that $\mathcal{LC}_{\mathit{hol}} \subseteq \mathcal{S}$. If $\iota_\iota \in \overline{\mathcal{S}}$, then $\iota_\sigma \in \overline{\mathcal{S}}$ for all types σ.*

Again, we present two lemmas which make it easy to see that this proposition holds:

Lemma 3.10. *Let $\mathcal{S}$ be a signature such that $\mathcal{LC}_{\mathit{hol}} \subseteq \mathcal{S}$. If $\iota_\beta \in \overline{\mathcal{S}}$ for any base type β, then $\iota_\sigma \in \overline{\mathcal{S}}$*

Proof. We give an inductive construction of ι_σ. The base cases follow by assumption. Let $\sigma = \tau\mu$. Consider the following term:

$$\lambda\iota : (\mu o)\mu, f : \sigma o,\ x : \tau.\ \iota(\lambda y : \mu.\ \exists g : \sigma.\ \neg(\neg f g \lor g x \neq_\mu y)$$

We use the same trick here which we already used in Lemma 3.4: The interpretation of this term is fixed. It will be interpreted as a function which returns description at type σ when getting description at type μ as argument. □

Note that this construction is not new. For example, Andrews presents it in his book [2, p. 234]. He also proves the following lemma [2, p. 233]. We give a slightly different proof.

Lemma 3.11. *Let $\mathcal{S}$ be be a signature such that $\mathcal{LC}_{\mathit{hol}} \subseteq \mathcal{S}$. Then $\iota_\sigma \in \overline{\mathcal{S}}$ for any purely propositional type σ.*

Proof. We construct ι_o using the following term:

$$\lambda p : oo.\ p\top \vee \neg p\bot$$

Starting from this term, we can construct ι_σ for any purely propositional type σ using the same construction as in the previous lemma. $\square$

4 A Signature Dependent Tableau System

4.1 Tableau Rules as Templates

Before we present the tableau system, we first need to clarify how the definitions for tableau rules and branches given in the introduction to this thesis fit into the world of $\mathcal{S}$-parameterized terms, frames and interpretations. For example can the rule

$$\frac{s \vee t}{s \mid t}$$

exist given a signature without $\vee$? And what are s and t? Another, even more important problem is that we will use discriminating terms (which are defined using disequations) for the definition of $\mathcal{D}\iota$. Moreover, disequations, in general, will be the "internal workhorses" of the system. What happens if the signature neither contains $=_\iota$ nor $\neg$? One solution could be to only consider signatures which contain at least $\neg$ and $=_\sigma$. Although this seems to be a good solution at first sight, this also has negative consequences: Equality is a very powerful logical constant. For example, it can be used to describe quantifiers. This means that adding equality (at all types) to a signature will very quickly end up in full simply typed higher-order logic and it is well known that there no proof system which is complete with respect to standard models (for example proven by Andrews [2, Theorem 7206 and Corollary 7207]).

To circumvent this problem, we introduce one additional level of terms, so-called *quasi-$\mathcal{S}$-formulas*: Let $\mathcal{S}$ be a signature. Every $\mathcal{S}$-formula is a quasi-$\mathcal{S}$-formula; if s is an $\mathcal{S}$-formula then $\neg s$ is a quasi-$\mathcal{S}$-formula; if s and t are $\mathcal{S}$-terms of type σ then $s \neq_\sigma t$ is a quasi-$\mathcal{S}$-formula. We use *qwff*$(\mathcal{S})$ to denote the set of all quasi-$\mathcal{S}$-formulas.

To make life easier, we define a *normalization operator* $[\cdot]$ on quasi-$\mathcal{S}$-formulas. If s is an $\mathcal{S}$-formula then [s] is already defined. In the other two cases, we define the normalization operator as follows:

$$\begin{array}{rcll} [\neg s] & = & \neg[s] & \text{if } s \text{ is an } \mathcal{S}\text{-formula} \\ [s \neq t] & = & [s] \neq [t] & \end{array}$$

A quasiformula s is *normal* iff $[s] = s$. A set of quasiformulas is *normal* iff every quasiformula in this set is normal.

Given the notion of quasi-$\mathcal{S}$-formulas, we also need a corresponding definition for interpretations. However, since this is not important at this point, we will postpone this problem and discuss it again in the next section.

Also note that when adding equality and negation, then the set of quasi-$\mathcal{S}$-formulas is subsumed by the set of $\mathcal{S}$-formulas. For example if the signature contains negation, the formula $\neg s$ can be a quasi-$\mathcal{S}$-formula as well as an $\mathcal{S}$-formula. It is intended that we will not be able to tell the difference. In order to avoid trouble, we will need to make sure that the definition of "satisfiability" on quasiformulas is compatible with the definition of satisfiability on formulas.

Back to the question of what s and t mean in the rule above and what happens in a signature without $\vee$. To make things more precise, we introduce the notion of a *step*: Let $\mathcal{S}$ be a signature. An $\mathcal{S}$-step is a pair $\langle A, \{A_1, \dots A_n\}\rangle$ such that $n \geq 0$, $A, A_1, \dots, A_n \subseteq$ *qwff*($\mathcal{S}$) are normal and finite and $A \subsetneq A_i$ for $1 \leq i \leq n$. An $\mathcal{S}$-step with $n = 0$ is called a *closing step*. Note that we only allow steps where all A_i (if there are any) contain something new.

In this context, tableau rules can be considered as *step templates* because they represent sets of steps. For example the disjunction rule from above corresponds to the set of all $\mathcal{S}$-steps such that there are normal $s, t \in$ *wff*(S) with $s \vee t \in A$, $s \notin A$, $t \notin A$, $A_1 = A \cup \{s\}$ and $A_2 = A \cup \{t\}$. Since the formulas in A, A_1 and A_2 need to be quasi-$\mathcal{S}$-formulas, this set will be always empty for signatures without $\vee$. Given a tableau rule $\mathcal{T}$, we use $\mathcal{R}_{\mathcal{S}}(\mathcal{T})$ to denote all $\mathcal{S}$-steps induced by $\mathcal{T}$.

An *$\mathcal{S}$-branch* is a finite set of normal quasi-$\mathcal{S}$-formulas. Given a set T of $\mathcal{S}$-steps, we say that a step $\langle A, \{A_1, \dots, A_n\} \in T$ *applies* to an $\mathcal{S}$-branch A' iff $A = A'$. An $\mathcal{S}$-branch A' is *closed (under T)* iff there is closing step in T that applies to A'. An $\mathcal{S}$-branch A' is *refutable (under T)* iff it is closed (under T) or there is a non-closing step in T that applies to A' and all alternatives of this step are refutable (under T).

4.2 The Basic Tableau System

We will now start to present the tableau system. Most rules are taken from [11], [9] and [8] where the authors discuss them in detail or give relevant references. We will explain them only very shortly and concentrate on the new rules for if-then-else, description and choice.

To begin with, we need rules for the empty signature. These rules handle almost all disequations and contrary formulas on the branch:

$$\mathcal{T}_{\text{FE}}\ \frac{s \neq_{\sigma\tau} t}{[sx] \neq_\tau [tx]}\ x \text{ fresh} \qquad \mathcal{T}_{\text{BE}}\ \frac{s \neq_o t}{s,\ \neg t \mid \neg s,\ t}$$

$$\mathcal{T}_{\text{DEC}}\ \frac{xs_1 \dots s_n \neq_\iota xt_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}\ n \geq 0 \qquad \mathcal{T}_{\text{MAT}}\ \frac{xs_1 \dots s_n,\ \neg xt_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}\ n \geq 0$$

Note that $\mathcal{T}_{\text{DEC}}$ and $\mathcal{T}_{\text{MAT}}$ only apply if there is a variable at the head of the formulas, respectively at the head of the terms on both sides of the disequation. Moreover, the set of steps induced by these two rules also contain closing steps for $n = 0$.

The first two logical constants we are considering are true and false. Having $\perp$ on the branch immediately represents a contradiction because $\bot$ can never be satisfied. Similarly, $\neg\top$ always represents a contradiction. Hence, the corresponding rules induce only closing steps:

$$\mathcal{T}_\bot\ \frac{\bot}{} \qquad \mathcal{T}_\top\ \frac{\neg\top}{}$$

Note that in the case of $\mathcal{T}_\top$, we cannot say whether $\neg\top$ is an $\mathcal{S}$-formula or a quasi-$\mathcal{S}$-formula. The rule applies in both cases.

Disjunction is straightforward. There is one rule for a negated $\vee$ and one rule for a non-negated $\vee$:

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

In the case of negation, we only need to consider formulas with two negations at their head. They cancel each other out:

$$\mathcal{T}_{\neg\neg}\ \frac{\neg\neg s}{s}$$

As for disjunction, there are two rules for the existential quantifiers:

$$\mathcal{T}_\exists\ \frac{\exists s}{[sx]}\ x \text{ fresh} \qquad \mathcal{T}_{\neg\exists}\ \frac{\neg(\exists s)}{[st]}\ t \in \Lambda^{\mathcal{S}}_{\sigma} \text{ normal}$$

The freshness condition "x fresh" means x does not occur free in any of the formulas on the branch before the application of the rule.

The last usual logical constant is equality:

$$\mathcal{T}_{\text{BQ}}\ \frac{s =_{\tau_1\ldots\tau_n o} t}{[su_1\ldots u_n],\ [tu_1\ldots u_n] \mid \neg[su_1\ldots u_n],\ \neg[tu_1\ldots u_n]}\ \begin{array}{l} n \geq 0, \\ u_i \in \Lambda^{\mathcal{S}}_{\tau_i} \text{ normal}\end{array}$$

$$\mathcal{T}_{\text{CON}}\ \frac{s =_{\tau_1\ldots\tau_n \iota} t,\ u \neq_\iota v}{[sw_1\ldots w_n] \neq_\iota u,\ [tw_1\ldots w_n] \neq_\iota u \mid [sw_1\ldots w_n] \neq_\iota v,\ [tw_1\ldots w_n] \neq_\iota v}\ \begin{array}{l} n \geq 0, \\ w_i \in \Lambda^{\mathcal{S}}_{\tau_i} \text{ normal}\end{array}$$

In the systems in [11], [9] and [8], the authors present two similar rules without the $\tau_1 \ldots \tau_n$ extension i.e., they restrict the rules to equations at ι respectively at o. The resulting gap is filled by an additional third rule to resolve functional equations:

$$\frac{s =_{\sigma\tau} t}{[su] =_\tau [tu]}\ u \text{ normal}$$

In the context of this thesis, this rule has a very negative effect: Adding $=_{\sigma\tau}$ to some signature forces us to also add $=_\tau$ to the signature if we want to preserve completeness. As a consequence, the tableau system does not only depend on the signature, but the signature also depends on the tableau system. In other words, we are restricted to a certain set of "nice" signatures. Hence, we decided to integrate this rule into $\mathcal{T}_{\text{BQ}}$ and $\mathcal{T}_{\text{CON}}$ to avoid this additional dependency.

The system we have seen so far has the following general concept: $\mathcal{T}_{\text{FE}}$ copes with functional disequations and turns them into disequations at o or at ι. $\mathcal{T}_{\text{BE}}$ handles disequations at o which are then mated by $\mathcal{T}_{\text{MAT}}$ to either close the branch or to get new disequations. $\mathcal{T}_{\text{DEC}}$ handles disequations at ι, again to either close the branch or to yield new disequations. In those cases, where formulas cannot be mated by $\mathcal{T}_{\text{MAT}}$, we check whether there is a logical constant at the head different from a disequation. If yes, the rules for this logical constant apply, if possible.

4.3 The New Logical Constants

4.3.1 Accessibility

We will need to ensure that certain terms do not occur underneath a lambda binder. For that purpose, we introduce the novel notion of *accessibility*.

We first define *elimination contexts*. Every $[]_\sigma$ is an elimination context. If C is an elimination context of type $\sigma\tau$ and s is a term of type σ, then Cs is an elimination context of type τ.

Using elimination contexts, we define *accessibility contexts*: If C is an elimination context of type o, then C and $\neg C$ are accessibility contexts. If s is a term of type ι and C is an elimination context of the same type, then $s \neq C$ and $C \neq s$ are accessibility contexts.

Note that we can see here one more advantage of having quasiformulas: We neither need to have $\neg$ nor $=$ in the signature to have accessibility contexts. On the other hand, we need to define how accessibility context can be applied to terms yielding quasiformulas. We do that the obvious way.

Let E be a set of quasiformulas. A term s is *accessible* in E iff there is an accessibility context C such that $C[s] \in E$.

4.3.2 If-Then-Else

As already explained in Section 1.2, one interesting fact about if-then-else is that, in contrast to all usual constants discussed in the previous section, it does not necessarily return something of type o. As a consequence, the general concept of the tableau system as described in the last section will not work anymore. Consider the following example:

$$\mathit{if}_\iota \bot xy \neq_\iota y$$

This formula is clearly unsatisfiable but we have no chance in accessing if_ι here as we did with the usual logical constant. Hence we need two new tableau rules, one for each side of a disequation:

$$\mathcal{T}_{\text{IFL}} \quad \frac{(\mathit{if}_\sigma stu)v_1 \ldots v_n \neq_\iota v'}{s,\ [tv_1 \ldots v_n] \neq_\iota v' \mid \neg s,\ [uv_1 \ldots v_n] \neq_\iota v'}\ n \geq 0$$

$$\mathcal{T}_{\text{IFR}} \quad \frac{v' \neq_\iota (\mathit{if}_\sigma stu)v_1 \ldots v_n}{s,\ [tv_1 \ldots v_n] \neq_\iota v' \mid \neg s,\ [uv_1 \ldots v_n] \neq_\iota v'}\ n \geq 0$$

The idea of these rules is not surprising: Informally, given some if-conditional which is different from some term, then either the condition is true and we know that the first branch is different from the term or the condition is false and the second branch is different from the term. The $v_1 \ldots v_n$ are to incorporate possible applications of $\mathcal{T}_{\text{FE}}$. As an alternative, we could require one side of the disequation in $\mathcal{T}_{\text{IFL}}$ and $\mathcal{T}_{\text{IFR}}$ to be $\mathit{if}stu$ (and nothing more) and introduce two additional rules:

$$\frac{(\mathit{if}_\sigma stu)v_1 \ldots v_n \neq_\iota v'}{\mathit{if}_\iota s[tv_1 \ldots v_n][uv_1 \ldots v_n] \neq_\iota v'}\ n \geq 1$$

$$\frac{v' \neq_\iota (\mathit{if}_\sigma stu)v_1 \ldots v_n}{v' \neq_\iota \mathit{if}_\iota s[tv_1 \ldots v_n][uv_1 \ldots v_n]}\ n \geq 1$$

The reason why we did not choose these rules is that they only have disadvantages: First, the tableau system would be bigger. Second, these rules introduce (at least) one intermediate step which we do not need: The proofs in the following sections will also work with $\mathcal{T}_{\text{IFL}}$ and $\mathcal{T}_{\text{IFR}}$. Third, similar to the equality rules, we would cause additional dependencies when adding *if* at a certain type to the signature.

Apart from the case that *if* occurs at the head of a discriminating term, it is still possible for *if* to return something of type o, i.e., to behave like a usual logical constant. The corresponding rules are straightforward:

$$\mathcal{T}_{\text{IFB}} \quad \frac{(\mathit{if}_\sigma stu)v_1 \ldots v_n}{s,\ [tv_1 \ldots v_n] \mid \neg s,\ [uv_1 \ldots v_n]}\ n \geq 0$$

$$\mathcal{T}_{\neg\text{IFB}} \quad \frac{\neg((\mathit{if}_\sigma stu)v_1 \ldots v_n)}{s,\ \neg[tv_1 \ldots v_n] \mid \neg s,\ \neg[uv_1 \ldots v_n]}\ n \geq 0$$

Given these rules for *if*, we can make an interesting observation: In all four cases, *if* is accessible in a certain accessibility context. We can make use of this fact to compress everything into one single rule:

$$\mathcal{T}_{\text{IF}} \quad \frac{C[\mathit{if}_\sigma stu]}{s,\ [C[t]] \mid \neg s,\ [C[u]]}\ C \text{ accessibility context}$$

Note that in both conclusions, the outer square brackets are the normalization operator and the inner square brackets correspond to the application of the contexts. Also note that the use of the accessibility contexts is more a cosmetic compression and will not save any work. For the proofs in the following sections, we will still need to expand the definition and cover all four cases.

4.3.3 Choice

As already mentioned in the introduction to this thesis, Mints [20] published in 1999 a proof system based on a sequent calculus by Takeuti [25]. This system had support for choice and was complete and cut-free. Since Mints did not use the simply typed lambda calculus as the underlying logic but the relational style, we needed to translate the new rules. The first rather direct translation resulted in the following three tableau rules (the subterm condition means that the term occurs somewhere as a subterm on the branch):

$$\frac{}{\neg[st] \mid [s(\varepsilon s)]} \quad \begin{array}{l}\varepsilon_\sigma s \text{ subterm,} \\ t \in \Lambda^{\mathcal{S}}_\sigma \text{ normal}\end{array}$$

$$\frac{}{[sx], \neg[tx] \mid \neg[sx], [tx] \mid s = t} \quad \begin{array}{l}\varepsilon_\sigma s, \varepsilon_\sigma t \text{ subterms,} \\ x \text{ fresh}\end{array}$$

$$\frac{as, \neg at}{[sx], \neg[tx] \mid \neg[sx], [tx]} \quad \begin{array}{l}a \text{ either some variable} \\ \text{or } \varepsilon,\ x \text{ fresh}\end{array}$$

Mints called the first rule *ε-rule*, the second one *choice extensionality* and the third one *standard extensionality*. Unfortunately, there were several problems with these translated rules:

- The subterm conditions are too weak. Consider for example the formula $\exists x.\ \neg(x(\varepsilon x))$. Applying the first rule from above pulls the bound variable x out of its binding context $\exists x.\ []$ where it does not make sense anymore. Mints did not have this problem because he had a clear distinction between bound and free variables.
- The second rule introduces $=$, i.e., requires equality to be in the signature. In addition, the rule considers two arbitrary subterms εs and εt on the branch. Given n different subterms of this kind, we can apply the rule $\frac{n^2-n}{2}$ times which results in $3^{\frac{n^2-n}{2}}$ branches. This exponential blow-up is a nasty property when it comes to automated proof search.
- The third rule looks similar to $\mathcal{T}_{\text{MAT}}$ but does not only allow variables at the head but also the choice operator. We were not sure whether we really needed this rule and, if yes, whether we also need a corresponding $\mathcal{T}_{\text{DEC}}$ rule.

The subterm problem could be easily solved: We developed the notion of *accessibility* and replaced the "is a subterm"-condition by an "is accessible"-condition. As already mentioned, accessibility ensures that the corresponding term cannot occur underneath a binder[1]. Using

[1]This problem was the original reason to introduce accessible terms. It turned out later on that it was also useful for simplifying the rules for if-then-else.

accessibility, the choice rule looks as follows:

$$\mathcal{T}_\varepsilon \quad \frac{C[\varepsilon s]}{\neg[st] \mid [s(\varepsilon s)]} \quad \begin{array}{l} C \text{ accessibility context,} \\ t \in \Lambda^{\mathcal{S}}_\sigma \text{ normal} \end{array}$$

This rule mimics the definition of choice: Given εs, either s corresponds to the empty set and hence $\neg(st)$ holds for arbitrary t, or s represents a set containing at least one element and choice will choose such an element. Thus $s(\varepsilon s)$ holds.

The other two problems could not be solved that easily. During our research, it turned out that there is a certain trade-off between having the second rule and having $\mathcal{T}_{\text{MAT}}$ and $\mathcal{T}_{\text{DEC}}$ with ε at their heads. Since the ε-extensionality rule has the disadvantageous properties mentioned above, we decided to try to prove completeness with modified versions of $\mathcal{T}_{\text{MAT}}$ and $\mathcal{T}_{\text{DEC}}$:

$$\mathcal{T}_{\text{MAT}_\varepsilon} \quad \frac{\varepsilon s_1 \dots s_n, \ \neg\varepsilon t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n} \qquad \mathcal{T}_{\text{DEC}_\varepsilon} \quad \frac{\varepsilon s_1 \dots s_n \neq_\iota \varepsilon t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}$$

As we will see in the next chapter, these two rules, together with $\mathcal{T}_\varepsilon$, indeed suffice.

4.3.4 Description

Given the rules for choice, it was straightforward to introduce corresponding rules for description: Besides the modified versions of $\mathcal{T}_{\text{MAT}}$ and $\mathcal{T}_{\text{DEC}}$, we copied $\mathcal{T}_\varepsilon$ and extended it with an additional third branch to cope with the case that a set contains more than one element:

$$\mathcal{T}_\iota \quad \frac{C[\iota s]}{\neg[st] \mid x \neq y, \ [sx], \ [sy] \mid [s(\iota s)]} \quad \begin{array}{l} C \text{ accessibility context,} \\ t \in \Lambda^{\mathcal{S}}_\sigma \text{ normal, x, y fresh} \end{array}$$

$$\mathcal{T}_{\text{MAT}_\iota} \quad \frac{\iota s_1 \dots s_n, \ \neg\iota t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n} \qquad \mathcal{T}_{\text{DEC}_\iota} \quad \frac{\iota s_1 \dots s_n \neq_\iota \iota t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}$$

4.4 The Final Tableau System

We have now seen all rules that will make up our tableau system. One interesting fact about this system is that, except for $\mathcal{T}_{\text{DEC}}$, $\mathcal{T}_{\text{MAT}}$, $\mathcal{T}_{\text{FE}}$ and $\mathcal{T}_{\text{BE}}$, there is no rule which requires certain logical constants to be in the signature, except the one for which the rule was made. Hence, we can always add a logical constant separately to the signature without the need to also add other logical constants.

We now define the signature dependent tableau system $T_{\mathcal{S}}$ given a signature $\mathcal{S}$. Note that the signature will not be affected by the tableau rules. Thus, all tableau systems will consist of the same tableau rules, regardless of the logical constants in the signature. What will change is the set of steps induced by these rules and we will use this fact to define the signature dependent tableau system T_S, given a signature $\mathcal{S}$:

$$\begin{aligned} T_\mathcal{S} \ = \ & R_\mathcal{S}(\mathcal{T}_{\text{FE}}) \cup R_\mathcal{S}(\mathcal{T}_{\text{BE}}) \cup R_\mathcal{S}(\mathcal{T}_{\text{DEC}}) \cup R_\mathcal{S}(\mathcal{T}_{\text{MAT}}) \cup R_\mathcal{S}(\mathcal{T}_\perp) \cup R_\mathcal{S}(\mathcal{T}_\vee) \cup R_\mathcal{S}(\mathcal{T}_{\neg\vee}) \\ & \cup R_\mathcal{S}(\mathcal{T}_{\neg\neg}) \cup R_\mathcal{S}(\mathcal{T}_\exists) \cup R_\mathcal{S}(\mathcal{T}_{\neg\exists}) \cup R_\mathcal{S}(\mathcal{T}_{\text{FQ}}) \cup R_\mathcal{S}(\mathcal{T}_{\text{BQ}}) \cup R_\mathcal{S}(\mathcal{T}_{\text{CON}}) \cup R_\mathcal{S}(\mathcal{T}_{\text{IF}}) \\ & \cup R_\mathcal{S}(\mathcal{T}_\varepsilon) \cup R_\mathcal{S}(\mathcal{T}_{\text{MAT}_\varepsilon}) \cup R_\mathcal{S}(\mathcal{T}_{\text{DEC}_\varepsilon}) \cup R_\mathcal{S}(\mathcal{T}_\iota) \cup R_\mathcal{S}(\mathcal{T}_{\text{MAT}_\iota}) \cup R_\mathcal{S}(\mathcal{T}_{\text{DEC}_\iota}) \end{aligned}$$

As a reference, Figure 4.1 shows all tableau rules. It remains to show that $T_\mathcal{S}$ is sound. This fact is expressed by the following proposition:

Proposition 4.1 (Soundness)**.** *Let A be a set of normal quasiformulas. If A is refutable, then A is unsatisfiable.*

Unfortunately, this proposition does not make sense at the moment. We first need to introduce a notion that deals with the problem of "satisfying quasiformulas".

Clearly, given a signature without the negation constant and a corresponding frame without the negation function, an interpretation into this frame cannot interpret the symbol $\neg$ as negation function. Hence, we need to introduce one more level above interpretations. We say that an *$\mathcal{S}$-interpretation $\mathcal{I}$ is a model for a quasi-$\mathcal{S}$-formula s* iff $\mathcal{I} \models s$ where $\models$ is defined on quasiformulas by cases:

$$\begin{aligned} \mathcal{I} \models s \quad & :\Longleftrightarrow \quad \hat{\mathcal{I}}s = 1 \quad \text{if } s \text{ is an } \mathcal{S}\text{-formula} \\ \mathcal{I} \models \neg s \quad & :\Longleftrightarrow \quad \hat{\mathcal{I}}s \neq 1 \\ \mathcal{I} \models s \neq t \quad & :\Longleftrightarrow \quad \hat{\mathcal{I}}s \neq \hat{\mathcal{I}}t \end{aligned}$$

Note that for $\mathcal{S}$-formulas, the notion of a model agrees with the notion of satisfiability on these terms. The definition of a model for quasiformulas reflects what we understand by "negation" or "disequation": An interpretation is a model for $\neg s$ iff it does *not* satisfy s. An interpetation is a model for $s \neq t$ iff the interpretation of s is *different* from the interpretation of t.

We extend the notions of satisfiability and validity to quasiformulas the obvious way and write "with respect to standard models" instead of "with respect to standard frames".

Proof of Proposition 4.1. This proof is straightforward. In the case that a closing step applies to A, the claim is obvious. Otherwise, it is enough to check for each step $\langle A, \{A_1, \cdots, A_n\}\rangle$ in $T_\mathcal{S}$ with $n > 0$ 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. $\square$

$$\mathcal{T}_{\text{FE}}\ \frac{s \neq_{\sigma\tau} t}{[sx] \neq_{\tau} [tx]}\ x \text{ fresh} \qquad \mathcal{T}_{\text{BE}}\ \frac{s \neq_{o} t}{s,\ \neg t \mid \neg s,\ t}$$

$$\mathcal{T}_{\text{MAT}}\ \frac{xs_1 \dots s_n,\ \neg xt_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}\ n \geq 1 \qquad \mathcal{T}_{\text{DEC}}\ \frac{xs_1 \dots s_n \neq_{\iota} xt_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}\ n \geq 1$$

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

$$\mathcal{T}_{\exists}\ \frac{\exists x.\ s}{[sx]}\ x \text{ fresh} \qquad \mathcal{T}_{\neg\exists}\ \frac{\neg(\exists x.\ s)}{[st]}\ t \in \Lambda_{\sigma}^{\mathcal{S}} \text{ normal}$$

$$\mathcal{T}_{\text{BQ}}\ \frac{s =_{\tau_1 \dots \tau_n o} t}{[su_1 \dots u_n],\ [tu_1 \dots u_n] \mid \neg[su_1 \dots u_n],\ \neg[tu_1 \dots u_n]}\ \begin{array}{l} n \geq 0, \\ u_i \in \Lambda_{\tau_i}^{\mathcal{S}} \text{ normal} \end{array}$$

$$\mathcal{T}_{\text{CON}}\ \frac{s =_{\tau_1 \dots \tau_n \iota} t,\ u \neq_{\iota} v}{[sw_1 \dots w_n] \neq_{\iota} u,\ [tw_1 \dots w_n] \neq_{\iota} u \mid [sw_1 \dots w_n] \neq_{\iota} v,\ [tw_1 \dots w_n] \neq_{\iota} v}\ \begin{array}{l} n \geq 0, \\ w_i \in \Lambda_{\tau_i}^{\mathcal{S}} \\ \text{normal} \end{array}$$

$$\mathcal{T}_{\text{IF}}\ \frac{C[\mathit{if}_{\sigma} stu]}{s,\ [C[t]] \mid \neg s,\ [C[u]]}\ C \text{ accessibility context}$$

$$\mathcal{T}_{\varepsilon}\ \frac{C[\varepsilon s]}{\neg[st] \mid [s(\varepsilon s)]}\ \begin{array}{l} C \text{ accessibility context,} \\ t \in \Lambda_{\sigma}^{\mathcal{S}} \text{ normal} \end{array}$$

$$\mathcal{T}_{\text{MAT}_{\varepsilon}}\ \frac{\varepsilon s_1 \dots s_n,\ \neg\varepsilon t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n} \qquad \mathcal{T}_{\text{DEC}_{\varepsilon}}\ \frac{\varepsilon s_1 \dots s_n \neq_{\iota} \varepsilon t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}$$

$$\mathcal{T}_{\iota}\ \frac{C[\iota s]}{\neg[st] \mid x \neq y,\ [sx],\ [sy] \mid [s(\iota s)]}\ \begin{array}{l} C \text{ accessibility context,} \\ t \in \Lambda_{\sigma}^{\mathcal{S}} \text{ normal, x, y fresh} \end{array}$$

$$\mathcal{T}_{\text{MAT}_{\iota}}\ \frac{\iota s_1 \dots s_n,\ \neg\iota t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n} \qquad \mathcal{T}_{\text{DEC}_{\iota}}\ \frac{\iota s_1 \dots s_n \neq_{\iota} \iota t_1 \dots t_n}{s_1 \neq t_1 \mid \dots \mid s_n \neq t_n}$$

Figure 4.1: The signature dependent tableau system T_S

5 Completeness

5.1 Evidence

Our goal in this section is to define the notion of *evident sets*. The definition will be based on a number of *evidence conditions* which look similar to the tableau rules presented in the last section.

Evident sets will have the property that there is a model for this set. This property is called *Model Existence* and its proof will be the most important part of this thesis.

Let $\mathcal{S}$ be a signature. A set $E \subseteq$ *qwff*$(\mathcal{S})$ is $\mathcal{S}$*-evident* iff it fullfils the evidence conditions from Figure 5.1 relative to $\mathcal{S}$.

Informally, an evident set corresponds to a branch which is not closed and where no tableau rule applies. The big difference is that branches (as well as steps) cannot be infinite. We will establish the connection between evident sets and branches later using a technique called *abstract consistency*.

Note that in $\mathcal{E}_{\mathrm{MAT}}$ and $\mathcal{E}_{\mathrm{DEC}}$ (and in the corresponding conditions for choice and description), we allow n to be equal to zero but require i to be greater than zero. This will explicitly exclude branches closed by these rules. Also note that most evidence conditions are signature independent. For example if a signature does not contain $\vee$, the condition in $\mathcal{E}_{\vee}$ holds vacuously. On the other hand, $\mathcal{E}_{\neg\exists}$ is signature dependent because it considers a certain set of $\mathcal{S}$-terms. We marked all such rules by a superscript $\mathcal{S}$.

5.2 Model Existence

As already mentioned, this section represents the most important part of this thesis. Our goal here is to prove the following theorem:

Theorem 5.1 (Model Existence)**.** *Let $\mathcal{S}$ be a signature and $E \subseteq$ qwff$(\mathcal{S})$ be an evident set. Then, there is an $\mathcal{S}$-frame $\mathcal{D}$ and an interpretation $\mathcal{I}$ into $\mathcal{D}$ such that $\mathcal{I} \models s$ for all $s \in E$*

The proof of this theorem is not straightforward. In order to structure it, we will split it up into several parts. Most of the definitions and techniques are taken from [11], [9] and [8]. We will also reuse many lemmas and only sketch the corresponding proofs or give references.

Let $\mathcal{S}$ be some signature and $E \subseteq$ *qwff*$(\mathcal{S})$ be an evident set. The first important part when constructing a model for E is to define $\mathcal{D}$ and especially $\mathcal{D}\iota$. This definiton will be based on *discriminants*.

$\mathcal{E}_{\text{FE}}$	If $s \neq_{\sigma\tau} t$ in E, then $[sx] \neq [tx]$ is in E for some variable x
$\mathcal{E}_{\text{BE}}$	If $s \neq_o t$ in E, then either s and $\neg t$ are in E or $\neg s$ and t are in E
$\mathcal{E}_{\text{DEC}}$	If $xs_1 \dots s_n \neq_\iota xt_1 \dots t_n$ is in E for some $n \geq 0$ and some variable x, then $s_i \neq t_i$ is in E for some $1 \leq i \leq n$
$\mathcal{E}_{\text{MAT}}$	If $xs_1 \dots s_n$ and $\neg xt_1 \dots t_n$ are in E for some $n \geq 0$ and some variable x, then $s_i \neq t_i$ is in E for some $1 \leq i \leq n$
$\mathcal{E}_\bot$	$\bot$ is not in E
$\mathcal{E}_\top$	$\neg\top$ is not in E
$\mathcal{E}_\vee$	If $s \vee t$ is in E, then either s in E or t in E
$\mathcal{E}_{\neg\vee}$	If $\neg(s \vee t)$ is in E, then $\neg s$ and $\neg t$ are in E
$\mathcal{E}_{\neg\neg}$	If $\neg\neg s$ is in E, then s is in E
$\mathcal{E}_\exists$	If $\exists s$ is in E, then $[sx]$ is in E for some variable x
$\mathcal{E}_{\neg\exists}^{\mathcal{S}}$	If $\neg(\exists_\sigma s)$ is in E, then $\neg[st]$ is in E for all normal $\mathcal{S}$-terms t of type σ
$\mathcal{E}_{\text{BQ}}$	If $s =_{\tau_1 \dots \tau_n o} t$ is in E for some $n \geq 0$, then either $[su_1 \dots u_n]$ and $[tu_1 \dots u_n]$ are in E or $\neg[su_1 \dots u_n]$ and $\neg[tu_1 \dots u_n]$ are in E for all normal $u_i \in \Lambda_{\tau_i}^{\mathcal{S}}$ for $1 \leq i \leq n$
$\mathcal{E}_{\text{CON}}$	If $s =_{\tau_1 \dots \tau_n \iota} t$ and $u \neq_\iota v$ are in E for some $n \geq 0$, then either $[sw_1 \dots w_n] \neq u$ and $[tw_1 \dots w_n] \neq u$ are in E or $[sw_1 \dots w_n] \neq v$ and $[tw_1 \dots w_n] \neq v$ are in E for all normal $w_i \in \Lambda_{\tau_i}^{\mathcal{S}}$ for $1 \leq i \leq n$
$\mathcal{E}_{\text{IF}}$	If $C[\mathit{if}\, s\, t\, u]$ is in E and C is an accessibility context, then s and $[C[t]]$ are in E or $\neg s$ and $[C[u]]$ are in E
$\mathcal{E}_\iota^{\mathcal{S}}$	If $C[\iota_\sigma s]$ is in E and C is an accessibility context, then either $[s(\iota s)]$ is in E, or $\neg[st]$ is in E for all normal $\mathcal{S}$-terms t of type σ, or $x \neq y$, $[sx]$ and $[sy]$ are in E for some variables x and y
$\mathcal{E}_{\text{DEC}_\iota}$	If $\iota s_1 \dots s_n \neq_\iota \iota t_1 \dots t_n$ is in E for some $n \geq 0$, then $s_i \neq t_i$ is in E for some $1 \leq i \leq n$
$\mathcal{E}_{\text{MAT}_\iota}$	If $\iota s_1 \dots s_n$ and $\neg\iota t_1 \dots t_n$ are in E for some $n \geq 0$, then $s_i \neq t_i$ is in E for some $1 \leq i \leq n$
$\mathcal{E}_\varepsilon^{\mathcal{S}}$	If $C[\varepsilon_\sigma s]$ is in E and C is an accessibility context, then either $[s(\varepsilon s)]$ is in E, or $\neg[st]$ is in E for all normal $\mathcal{S}$-terms t of type σ
$\mathcal{E}_{\text{DEC}_\varepsilon}$	If $\varepsilon s_1 \dots s_n \neq_\iota \varepsilon t_1 \dots t_n$ is in E for some $n \geq 0$, then $s_i \neq t_i$ is in E for some $1 \leq i \leq n$
$\mathcal{E}_{\text{MAT}_\varepsilon}$	If $\varepsilon s_1 \dots s_n$ and $\neg\varepsilon t_1 \dots t_n$ are in E for some $n \geq 0$, then $s_i \neq t_i$ is in E for some $1 \leq i \leq n$

Figure 5.1: Evidence conditions for a set of quasi-$\mathcal{S}$-formulas

5.2.1 Discriminants

From now on, we write $s\sharp t$ iff $s \neq t \ \in E$ or $t \neq s \in E$. We will also use $s\sharp t$ to denote one of the two disequations. A term s of type ι is discriminating iff there is a term t of type ι such that $s\sharp t$. A discriminant d of E is a maximal set of discriminating terms of E such that there are no $s, t \in d$ with $s\sharp t$.

We define $\mathcal{D}\iota$ to be the set of all discriminants of E.

Example 5.2. Let E be $\{x \neq y, y \neq z\}$ where $x : \iota, y : \iota, z : \iota$. In this case, E has two discriminants: $\{x, z\}$ and $\{y\}$

Example 5.3. Let E be $\{fx\}$ where $f : \iota o, x : \iota$. Since there are no discriminating terms, $\emptyset$ is the only discriminant.

5.2.2 Compatibility

Before we continue to define $\mathcal{D}$, we introduce an auxiliary notion, called *compatibility*. Among other things, it will help us to prove that $s \neq s \notin E$ for any term s. Compatibility is defined by induction on types:

$$\begin{aligned} s \parallel_o t \ &:\Longleftrightarrow \ \{[s], \neg[t]\} \not\subseteq E \text{ and } \{\neg[s], [t]\} \not\subseteq E \\ s \parallel_\iota t \ &:\Longleftrightarrow \ \text{not } [s]\sharp[t] \\ s \parallel_{\tau\mu} t \ &:\Longleftrightarrow \ su \parallel_\mu tv \text{ whenever } u \parallel_\tau v \end{aligned}$$

We also define compatibility on substitutions:

$$\theta \parallel \theta' \ :\Longleftrightarrow \ \theta x \parallel \theta' x \text{ for all variables } x$$

We say that two terms s and t are *compatible* iff $s \parallel t$. Two substitutions θ and θ' are *compatible* iff $\theta \parallel \theta'$. A set A of equi-typed terms is compatible iff $s \parallel t$ for all $s, t \in A$. The compatibility relations are obviously symmetric. They are also reflexive. To show that, we first prove $x \parallel x$ for all variables x.

Lemma 5.4. *For every type σ and all terms $s, t, xs_1 \ldots s_n, xt_1 \ldots t_n$ of type σ with $n \geq 0$:*

1. *Not both $s \parallel_\sigma t$ and $[s]\sharp[t]$.*
2. *Either $xs_1 \ldots s_n \parallel xt_1 \ldots t_n$ or $[s_i]\sharp[t_i]$ for some $0 < i \leq n$.*

Proof. The full proof is given in [9, Lemma 7.6]. It is done by mutual induction on σ and uses $\mathcal{E}_{\text{BE}}$, $\mathcal{E}_{\text{FE}}$, $\mathcal{E}_{\text{MAT}}$ and $\mathcal{E}_{\text{DEC}}$ □

The second statement of this lemma implies $x \parallel x$ (for $n = 0$). The first statement is very interesting concerning our goal to prove $s \neq s \notin E$: It is indeed enough to show that the compatibility relations are reflexive.

Lemma 5.5. *Let s and t be terms of type σ. $s \parallel_\sigma t$ iff $[s] \parallel_\sigma [t]$.*

Proof. Proof by induction on σ. The base cases follow by N1 and N2. Let $\sigma = \mu\tau$ and let u, v be two terms of type μ such that $u \parallel_\mu v$. By the definition of $\parallel$, $su \parallel tv$. By the inductive hypothesis, $[su] \parallel [tv]$. $[[s]u] \parallel [[t]v]$ by N2. By the inductive hypothesis, $[s]u \parallel [t]v$ and by the definition of $\parallel$, $[s] \parallel [t]$. □

The following lemma shows that every logical constant is compatible to itself:

Lemma 5.6. *$c \parallel c$ for any $c \in \mathcal{S}$*

Proof.

- $c \in \{\bot, \top\}$: Follows by the definition of $\parallel_o$, N1 and N3
- $c = \neg$: Assume $\neg \nparallel \neg$. Then, there are terms t, u such that $t \parallel_o u$ but $\neg t \nparallel_o \neg u$. By $t \parallel_o u$, we know that $\{[t], \neg[u]\} \nsubseteq E$ and $\{\neg[t], [u]\} \nsubseteq E$. By $\neg t \nparallel_o \neg u$, N3 and $\mathcal{E}_{\neg\neg}$, we know that $\{\neg[t], [u]\} \subseteq E$ or $\{[t], \neg[u]\} \subseteq E$. Contradicition.
- $c = \vee$: Assume $\vee \nparallel \vee$. Then, there are terms t_1, t_2, u_1, u_2 such that $t_1 \parallel u_1$ and $t_2 \parallel u_2$ but $s_1 \vee s_2 \nparallel t_1 \vee t_2$. We know that none of $\{[t_1], \neg[u_1]\}$, $\{\neg[t_1], [u_1]\}$, $\{[t_2], \neg[u_2]\}$ and $\{\neg[t_2], [u_2]\}$ is a subset of E. Using N3, we also know that either $\{[t_1] \vee [t_2], \neg([u_1] \vee [u_2])\} \subseteq E$ or $\{\neg([t_1] \vee [t_2]), [u_1] \vee [u_2]\} \subseteq E$. Thus, by $\mathcal{E}_\vee$ and $\mathcal{E}_{\neg\vee}$, one of $\{[t_1], \neg[u_1], \neg[u_2]\}$, $\{[t_2], \neg[u_1], \neg[u_2]\}$, $\{\neg[t_1], \neg[t_2], [u_1]\}$ or $\{\neg[t_1], \neg[t_2], [u_2]\}$ is a subset of E. Contradiction.
- $c = \exists_\sigma$: Assume $\exists \nparallel \exists$. Then, there are terms t, u of type σo such that $t \parallel u$ but $\exists t \nparallel \exists u$. Without loss of generality, we assume $\{[\exists t], \neg[\exists u]\} \subseteq E$. By N3 $\{\exists[t], \neg\exists[u]\} \subseteq E$. By $\mathcal{E}_\exists$, $\mathcal{E}_{\neg\exists}$ and N2, there is a variable x of type σ such that $\{[tx], \neg[ux]\} \subseteq E$. By Lemma 5.4, we know that $x \parallel x$. Hence, $sx \parallel tx$. By definition of $\parallel_o$, $\{[tx], \neg[ux]\} \nsubseteq E$. Contradiction.
- $c = =_{\tau_1 \ldots \tau_n \mu}$ for $n \geq 0$ and $\mu \in \{\iota, o\}$: Assume $= \nparallel =$. Then, there are terms t_1, t_2, u_1, u_2 of type $\tau_1 \ldots \tau_n \mu$ such that $t_1 \parallel u_1$, $t_2 \parallel u_2$ but $t_1 = t_2 \nparallel u_1 = u_2$. By the definition of $\parallel_o$ and N3, we know that either $\{[t_1] = [t_2], [u_1] \neq [u_2]\} \subseteq E$ or $\{[t_1] \neq [t_2], [u_1] = [u_2]\} \subseteq E$. We will only show that the first case yields a contradiction. The second case works analogously.

  By $\mathcal{E}_{\text{FE}}$ and N2, there are fresh variables $x_1, \ldots, x_n$ such that

  $$[u_1 x_1 \ldots x_n] \neq_\mu [u_2 x_1 \ldots x_n] \in E$$

  $x_i \parallel x_i$ by Lemma 5.4 for $0 \leq i \leq n$ and hence,

  $$\overbrace{t_1 x_1 \ldots x_n}^{t_1'} \parallel_\mu \overbrace{u_1 x_1 \ldots x_n}^{u_1'} \text{ and } \overbrace{t_2 x_1 \ldots x_n}^{t_2'} \parallel_\mu \overbrace{u_2 x_1 \ldots x_n}^{u_2'}$$

Suppose $\mu = o$. Then, by the definition of $\|_o$, we know that none of $\{[t_1'], \neg[u_1']\}$, $\{\neg[t_1'], [u_1']\}$, $\{[t_2'], \neg[u_2']\}$ and $\{\neg[t_2'], [u_2']\}$ is a subset of E. On the other hand, by $\mathcal{E}_{\text{BQ}}$ and $\mathcal{E}_{\text{BE}}$, we know that one of $\{[t_1'], [t_2'], [u_1'], \neg[u_2']\}$, $\{[t_1'], [t_2'], \neg[u_1'], [u_2']\}$, $\{\neg[t_1'], \neg[t_2'], [u_1'], \neg[u_2']\}$ or $\{\neg[t_1'], \neg[t_2'], \neg[u_1'], [u_2']\}$ is a subset of E. Contradiction in all four cases.

Suppose $\mu = \iota$. Then, by the definition of $\|_\iota$, we know that neither $[t_1']\sharp[u_1']$ nor $[t_2']\sharp[u_2']$. By $\mathcal{E}_{\text{CON}}$, we also know that either $\{[t_1'] \neq [u_1'], [t_2'] \neq [u_1']\} \subseteq E$ or $\{[t_1'] \neq [u_2'], [t_2'] \neq [u_2']\} \subseteq E$. Contradiction in both cases.

- $c = \mathit{if}_{\tau_1 \dots \tau_n \mu}$ for some $n \geq 0$ and $\mu \in \{\iota, o\}$: Assume $\mathit{if}_{\tau_1 \dots \tau_n \mu} \nparallel \mathit{if}_{\tau_1 \dots \tau_n \mu}$. Then, there are terms $t_1, t_2, t_3, u_1, u_2, u_3, v_1 \dots v_n$ and $w_1 \dots w_n$ such that $t_1 \|_o u_1$, $t_2 \|_{\tau_1 \dots \tau_n \mu} u_2$, $t_3 \|_{\tau_1 \dots \tau_n \mu} u_3$ and $v_i \|_{\tau_i} w_i$ for $0 \leq i \leq n$ but

$$\mathit{if}\, t_1\, t_2\, t_3\, v_1 \dots v_n \nparallel_\mu \mathit{if}\, u_1\, u_2\, u_3\, w_1 \dots w_n$$

By Lemma 5.5 and definition of $\|$, we know that

$$[t_2][v_1] \dots [v_n] \|_\mu [u_2][w_1] \dots [w_n] \text{ and } [t_3][v_1] \dots [v_n] \|_\mu [u_3][w_1] \dots [w_n]$$

Suppose $\mu = o$. Then by N3, either

$$\{\mathit{if}[t_1][t_2][t_3][v_1] \dots [v_n], \neg(\mathit{if}[u_1][u_2][u_3][w_1] \dots [w_n])\} \subseteq E$$

or

$$\{\neg(\mathit{if}[t_1][t_2][t_3][v_1] \dots [v_n]), \mathit{if}[u_1][u_2][u_3][w_1] \dots [w_n]\} \subseteq E$$

We only show that the first case yields a contradiction. We use $\mathcal{E}_{\text{IF}}$ twice, once with accessibility context $[][v_1] \dots [v_n]$ and once with accessibility context $\neg([][w_1] \dots [w_n])$. We get four cases where we need to apply the definition of normalization on quasi-formulas. In two of the four cases, either $\{\neg[t_1], [u_1]\} \subseteq E$ or $\{[t_1], \neg[u_1]\} \subseteq E$ contradicting the compatibility of t_1 and u_1. In the other two cases, either

$$\{[[t_2][v_1] \dots [v_n]], \neg[[u_2][w_1] \dots [w_n]]\} \subseteq E$$

or

$$\{[[t_3][v_1] \dots [v_n]], \neg[[u_3][w_1] \dots [w_n]]\} \subseteq E$$

contradicting the compatibility of those terms as stated above.

Suppose $\mu = \iota$: Analogously to $\mu = o$ using accessibility contexts

$$[][v_1] \dots [v_n]\sharp\mathit{if}[u_1][u_2][u_3][w_1] \dots [w_n] \text{ and } \mathit{if}[t_1][t_2][t_3][v_1] \dots [v_n]\sharp[][w_1] \dots [w_n]$$

- $c = \varepsilon$: Assume for contradiction $\varepsilon \nparallel_{(\sigma o)\sigma} \varepsilon$. Then there are terms $s, t, u_1 \dots u_n$, $v_1 \dots v_n$ for $n \geq 0$ (depending on σ) such that $s \| t$, $u_i \| v_i$ but $\varepsilon s u_1 \dots u_n \nparallel_\mu \varepsilon t v_1 \dots v_n$ and $\mu = o$ or $\mu = \iota$.

Suppose $\mu = o$: By the definition of $\|_o$ and N3, $\varepsilon[s][u_1]\dots[u_n], \neg\varepsilon[t][v_1]\dots[v_n] \in E$ or $\neg\varepsilon[s][u_1]\dots[u_n], \varepsilon[t][v_1]\dots[v_n] \in E$. In both cases, by $\mathcal{E}_{\mathrm{MAT}_\varepsilon}$, either $[s]\sharp[t]$ or $[u_i]\sharp[v_i]$ for some i. Contradiction by Lemma 5.4.

Suppose $\mu = \iota$: Analogously to $\mu = o$ using $\mathcal{E}_{\mathrm{DEC}_\varepsilon}$ instead of $\mathcal{E}_{\mathrm{MAT}_\varepsilon}$.

- $c = \iota$: Analogously to ε using $\mathcal{E}_{\mathrm{DEC}_\iota}$ and $\mathcal{E}_{\mathrm{MAT}_\iota}$. □

Lemma 5.7. *Let θ, θ' be two compatible substitutions. Then $\widehat{\theta}s \parallel \widehat{\theta'}s$ for all terms s.*

Proof. Proof by induction on s. Case analysis:

- $s = x$: The claim holds by the compatibility of θ and θ'
- $s = c$: By S1, $\widehat{\theta}c \parallel \widehat{\theta'}c \ = \ c \parallel c$ which follows from Lemma 5.6.
- $s = tu$: We show $\widehat{\theta}(tu) \parallel \widehat{\theta'}(tu)$. By S2 this is equivalent to $(\widehat{\theta}t)(\widehat{\theta}u) \parallel (\widehat{\theta'}t)(\widehat{\theta'}u)$ which holds by the inductive hypothesis and by the definition of $\parallel$.
- $s = \lambda x.\ t$: We show $\widehat{\theta}(\lambda x.\ t) \parallel \widehat{\theta'}(\lambda x.\ t)$. Assume for contradiction that $\widehat{\theta}(\lambda x.\ t) \nparallel \widehat{\theta'}(\lambda x.\ t)$. Then there are terms u, v such that $u \parallel v$ but $(\widehat{\theta}(\lambda x.\ t))u \nparallel (\widehat{\theta'}(\lambda x.\ t))v$. By Lemma 5.5 $[(\widehat{\theta}(\lambda x.\ t))u] \nparallel [(\widehat{\theta'}(\lambda x.\ t))v]$, $[\widehat{\theta^x_u}t] \nparallel [\widehat{\theta'^x_v}t]$ by S3 and $\widehat{\theta^x_u}t \nparallel \widehat{\theta'^x_v}t$ again by Lemma 5.5. Since u and v are compatible, $\widehat{\theta^x_u}$ and $\widehat{\theta'^x_v}$ are still compatible, too. Hence, we have a contradiction using the inductive hypothesis. □

Lemma 5.8. *$s \parallel s$ for all terms s.*

Proof. By Lemma 5.5 and by S4, $s \parallel s$ is equivalent to $[\widehat{\emptyset}s] \parallel [\widehat{\emptyset}s]$. This is, again by Lemma 5.5, equivalent to $\widehat{\emptyset}s \parallel \widehat{\emptyset}s$ which holds by Lemma 5.7 and by the fact that the empty substitution is compatible to itself (which trivially follows from Lemma 5.4). □

Lemma 5.9. *$s \neq s \notin E$ for all terms s.*

Proof. Follows by Lemma 5.4 (1) and Lemma 5.8. □

5.2.3 Possible Values

Given $\mathcal{D}\iota$ it remains to define $\mathcal{D}o$ and $\mathcal{D}(\sigma\tau)$ for all types σ and τ. We do that by mutual induction on types using $\mathcal{D}$ on the one hand and a *possible values relation* $\triangleright_\sigma \subseteq \Lambda^{\mathcal{S}}_\sigma \times \mathcal{D}\sigma$ for all types σ on the other hand:

$$\begin{aligned}
s \triangleright_o 0 \ &:\Longleftrightarrow \ [s] \notin E \\
s \triangleright_o 1 \ &:\Longleftrightarrow \ \neg[s] \notin E \\
s \triangleright_\iota a \ &:\Longleftrightarrow \ ([s] \text{ discriminating } \Longrightarrow [s] \in a) \\
s \triangleright_{\sigma\tau} f \ &:\Longleftrightarrow \ st \triangleright_\tau fa \text{ whenever } t \triangleright_\sigma a
\end{aligned}$$

In the last case, a ranges over the elements of $\mathcal{D}\sigma$ which we define as $\mathrm{Ran}\,(\triangleright_\sigma)$ for $\sigma \neq \iota$. Informally, $\mathcal{D}$ contains all those values which are related to some term, i.e., which are a possible value for some term.

Proposition 5.10. *$\triangleright$ is surjective, i.e., for any type σ and any value $a \in D\sigma$, there is some term $s : \sigma$ such that $s \triangleright_\sigma a$*

Proof. For $\sigma \in \{\iota, o\}$, the claim trivially follows from the definition of $\mathcal{D}$. Let $\sigma = \iota$. Assume there are no discriminating terms in E. Then $\emptyset$ is the only discriminant and $x \triangleright_\iota \emptyset$ for any variable x of type ι. Assume there are discriminating terms in E. We first observe that for any discriminating term s, $s \neq s \notin E$ by Lemma 5.9. Hence, by the maximality of discriminants, there is at least one term t for each discriminant d such that $t \in d$. By the definition, $t \triangleright_\iota d$. □

Lemma 5.11. *For any term s and any value a, $s \triangleright a$ iff $[s] \triangleright a$.*

Proof. The full proof is given in [8, Lemma 3.1]. It is done by a simple induction on types using N2. □

Given the definition of $\mathcal{D}$, we need to show that $\mathcal{D}$ is an $\mathcal{S}$-frame. We first show that $\mathcal{D}$ is a frame, i.e., that $\mathcal{D}\sigma$ is nonempty for all σ. Let $T \subseteq \Lambda_\sigma^{\mathcal{S}}$ be a set of equi-typed terms. We write $T \triangleright_\sigma a$ iff $s \triangleright_\sigma a$ for all $s \in T$ and say that T has *common possible value* a. The following lemma shows that a set of compatible terms has a common possible value:

Lemma 5.12. *Let $T \subseteq \Lambda_\sigma^{\mathcal{S}}$. T is compatible iff T has a common possible value.*

Proof. The full proof given in [8, Lemma 6.3]. It is done by induction on σ. □

This lemma finally allows us to show that $\mathcal{D}$ is a frame:

Lemma 5.13. *$\mathcal{D}\sigma$ is nonempty for any σ.*

Proof. We know that for any σ there is a variable $x : \sigma$. By Lemma 5.4, we know that $x \parallel x$. By Lemma 5.12 there is a value a such that $x \triangleright_\sigma a$. By the definition of $\mathcal{D}$, $a \in \mathcal{D}\sigma$. □

It remains to show that $\mathcal{D}$ is an $\mathcal{S}$-frame, i.e., that $\mathcal{D}$ realizes all logical constants in $\mathcal{S}$. We first turn to choice and description.

Let $c \in \{\varepsilon, \iota\}$, $f \in \mathcal{D}(\sigma o)$ be a function and cs be a term in $\Lambda_\sigma^{\mathcal{S}}$. We write $f \propto cs$ (read f *chooses* cs) iff $s \triangleright f$ and $c[s]$ is accessible in E. We call the set $f^c = \{cs \in \Lambda_\sigma^{\mathcal{S}} | f \propto cs\}$ the set of *c-relevant* terms for f.

Lemma 5.14. *Let $c \in \{\varepsilon, \iota\}$, 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 $f^c \triangleright a$.*

Proof. We show that f^c is compatible. Lemma 5.12 gives us the claim. Let $cs, ct \in f^c$. By the definition of $\propto$, $s, t \triangleright f$ and hence, by Lemma 5.12, $s \parallel t$. By Lemma 5.8 $c \parallel c$. Thus $cs \parallel ct$. □

For any type σ, we define a choice operator $\Phi_\sigma \in \mathcal{D}(\sigma o) \to \mathcal{D}\sigma$ and a description operator $\Psi_\sigma \in \mathcal{D}(\sigma o) \to \mathcal{D}\sigma$:

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

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

Lemma 5.15. *Let E be an evident branch, $c \in \mathcal{S}$, $ct_1 \dots t_n \in \Lambda_\sigma^{\mathcal{S}}$ and $a \in \mathcal{D}\sigma$. If $ct_1 \dots t_n \not\triangleright a$, then $c[t_1] \dots [t_n]$ is accessible in E.*

Proof. By induction on σ. Case analysis:

- $\sigma = o$: If $a = 0$ then by the definition of $\triangleright_o$ and N3, $c[t_1] \dots [t_n] \in E$. If $a = 1$ then, again by the definition of $\triangleright_o$ and N3, $\neg c[t_1] \dots [t_n] \in E$.
- $\sigma = \iota$: By the definition of $\triangleright_\iota$ and N3, we know that $c[t_1] \dots [t_n]$ is discriminating and hence accessible.
- $\sigma = \mu\tau$: By the definition of $\triangleright_\sigma$, we know that there is some term $u \in \Lambda_\mu^{\mathcal{S}}$ and some value $b \in \mathcal{D}\mu$ such that $u \triangleright b$ but $ct_1 \dots t_n u \not\triangleright ab$. By the inductive hypothesis, we know that $c[t_1] \dots [t_n][u]$ is accessible in E. Hence, $c[t_1] \dots [t_n]$ is accessible. □

The next lemma shows that for any type σ, Φ_σ and Ψ_σ are indeed in $\mathcal{D}((\sigma o)\sigma)$

Lemma 5.16. *For any type σ, we have $\varepsilon_\sigma \triangleright \Phi_\sigma$ and $\iota_\sigma \triangleright \Psi_\sigma$*

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 5.15 $\varepsilon[s]$ is accessible in E. Hence $\varepsilon s \in f^\varepsilon$. There is some a such that $\Phi f = a$ and $f^\varepsilon \triangleright a$. Thus $\varepsilon s \triangleright a$, a contradiction. The proof for $\iota \triangleright \Psi$ can be done analogously. □

Now, we can show that $\mathcal{P}_{\varepsilon_\sigma}(\Phi_\sigma)$ (respectively $\mathcal{P}_{\iota_\sigma}(\Psi_\sigma)$) holds, i.e., that ε and ι are realized by $\mathcal{D}$ if these logical constants are in $\mathcal{S}$.

Lemma 5.17. *$\mathcal{P}_{\varepsilon_\sigma}(\Phi_\sigma)$ 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^{ε_σ} must be nonempty (by the definition of Φf). Choose some $\varepsilon s \in f^{\varepsilon_\sigma}$. 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 5.16) 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 normal $t \in \Lambda_\sigma^{\mathcal{S}}$. By Proposition 5.10 and Lemma 5.11 there is some normal term $u \in \Lambda_\sigma^{\mathcal{S}}$ such that $u \triangleright b$. Hence $\neg[su] \in E$. By the definition of $\triangleright_o$, $su \not\triangleright 1$. On the other hand, we know $su \triangleright fb$ since $s \triangleright f$ and $u \triangleright b$, contradicting the assumption that $fb = 1$. □

Lemma 5.18. *$\mathcal{P}_{\iota_\sigma}(\Psi_\sigma)$ holds. That is, Ψ as defined above is a description function.*

Proof. Let $f \in \mathcal{D}(\sigma o)$ be a function and $b \in \mathcal{D}\sigma$ be such that $fb = 1$ and $fc = 0$ for all $c \in \mathcal{D}\sigma$ different from b. Suppose $f(\Phi f) = 0$. Then f^{ι_σ} must be nonempty (by the definition of Ψf). Choose some $\iota s \in f^{\iota_\sigma}$. We will show a contradiction. By $\mathcal{E}_\iota$ there are three possibilities:

1. $[s(\iota s)] \in E$: In this case $s(\iota s) \not\triangleright 0$. On the other hand, $s \triangleright f$ and $\iota \triangleright \Psi$ (by Lemma 5.16) and so $s(\varepsilon s) \triangleright f(\Psi f)$. This contradicts our assumption that $f(\Psi f) = 0$.

2. $\neg[st] \in E$ for every normal $t \in \Lambda^{\mathcal{S}}_{\sigma}$. By Proposition 5.10 and Lemma 5.11 there is some normal term $u \in \Lambda^{\mathcal{S}}_{\sigma}$ such that $u \rhd b$. Hence $\neg[su] \in E$. By the definition of $\rhd_o$, $su \not\rhd 1$. On the other hand, we know $su \rhd fb$ since $s \rhd f$ and $u \rhd b$, contradicting the assumption that $fb = 1$.

3. $\{x \neq y, [sx], [sy]\} \subseteq E$ for some variables x and y. By Lemma 5.4(2) and Lemma 5.12, we know that there must be values c_1 and c_2 such that $x \rhd c_1$ and $y \rhd c_2$. On the other hand, by N3 and Lemma 5.4(1), we know that x and y are not compatible. Hence, by Lemma 5.12, there is no common possible value for x and y. As a consequence, c_1 and c_2 must be different. By $s \rhd f$, we know $sx \rhd fc_1$ and $sy \rhd fc_2$. Since $\{[sx], [sy]\} \subseteq E$, fc_1 and fc_2 must both be 1 contradicting the uniqueness of b. □

This concludes the main part for choice and description. The following lemma will help us to prove that $\mathcal{D}$ realizes $=_\sigma$ (assuming that $=_\sigma$ is in $\mathcal{S}$).

Lemma 5.19. *Let $s =_{\tau_1 \ldots \tau_n \mu} t \in E$, $s \rhd a$ and $t \rhd b$ for $n \geq 0$ and $\mu \in \{o, \iota\}$. Then $a = b$.*

Proof. By contradiction. Assume $s \rhd_\sigma a$, $t \rhd_\sigma b$, $(s{=}t) \in E$, and $a \neq b$. Then there are values $c_1, \ldots, c_n$ such that

$$\overbrace{ac_1 \ldots c_n}^{a'} \neq \overbrace{bc_1 \ldots c_n}^{b'}$$

By Proposition 5.10 and Lemma 5.11, there are normal terms $u_1, \ldots, u_n$ such that $u_i \rhd c_i$ for $0 \leq i \leq n$ and thus

$$\overbrace{su_1 \ldots u_n}^{s'} \rhd_\mu a' \text{ and } \overbrace{tu_1 \ldots u_n}^{t'} \rhd_\mu b'$$

Case analysis on μ.

$\mu = o$. By $\mathcal{E}_{\text{BQ}}$ either $[s'], [t'] \in E$ or $\neg[s'], \neg[t'] \in E$. Hence, using Lemma 5.11, a' and b' are either both 1 or both 0. Contradiction.

$\sigma = \iota$. Since $a' \neq b'$, there must be discriminating terms of type ι. Since the discriminant a' is maximal there is some $v \in a' \setminus b'$. Since b' is also maximal, $b' \cup \{v\}$ is not a discriminant. Hence there is some $w \in b'$ such that $v \sharp w$. By $\mathcal{E}_{\text{CON}}$, we know either $[s'] \sharp v$ or $[t'] \sharp w$. If $[s'] \sharp v$, then $[s']$ is discriminating and $[s'] \in a'$, contradicting that a' is a discriminant with $v \in a'$. Likewise, if $[t'] \sharp w$, then $[t'] \in b'$, contradicting $w \in b'$. □

Lemma 5.20. *For any logical constant $c : \sigma \in \mathcal{S}$, there is some $a \in \mathcal{D}\sigma$ such that $c \rhd a$ and $\mathcal{P}_c(a)$ holds.*

Proof. We show $c \rhd a$ for all possible logical constants c. Since these proofs are independent from each other, they still work when $\mathcal{S}$ is restricted to a certain subset of $\mathcal{LC}$.

The cases for ε and ι are already covered by Lemma 5.17 and Lemma 5.18. Let $n \in \mathcal{D}(oo)$ be the negation function, $d \in \mathcal{D}(ooo)$ the disjunction function, $i_\sigma \in \mathcal{D}(o\sigma\sigma\sigma)$ be the if-then-else function, $q_\sigma \in \mathcal{D}(\sigma\sigma o)$ the equality function and $e_\sigma \in \mathcal{D}((\sigma o)o)$ be the function such that $e_\sigma f = 1$ if f is not the constant 0 function.

- $\top \rhd 1$ and $\bot \rhd 0$: The claim follows by $\mathcal{E}_\top$ (respectively $\mathcal{E}_\bot$) and N3

- $\neg \triangleright n$: Assume $\neg \not\triangleright n$. Then, there is term s and a value a such that $s \triangleright a$ but $\neg s \not\triangleright na$. If $a = 0$ then $[s] \notin E$ and $\neg[\neg s] \in E$. By N3, $\neg\neg[s] \in E$ and $[s] \in E$ by $\mathcal{E}_{\neg\neg}$. Contradiction. If $a = 1$ then $\neg[s] \notin E$ and $[\neg s] \in E$. Contradiction by N3.

- $\vee \triangleright d$: Assume $\vee \not\triangleright d$. Then, there are are terms s, t and values a, b such that $s \triangleright a$ and $t \triangleright b$ but $s \vee t \not\triangleright dab$.

  If $a = 0$ and $b = 0$, then, $[s \vee t] \in E$ and by N3, $[s] \vee [t] \in E$. By $\mathcal{E}_{\vee}$, $[s] \in E$ or $[t] \in E$ contradicting $s \triangleright 0$ and $t \triangleright 0$.

  If $a = 1$ or $b = 1$, then $\neg[s \vee t] \in E$ and by N3, $\neg([s] \vee [t]) \in E$. By $\mathcal{E}_{\neg\vee}$, $\neg[s] \in E$ and $\neg[t] \in E$ contradicting $s \triangleright 1$ or $t \triangleright 1$

- $\mathit{if}_\sigma \triangleright i_\sigma$: Assume $\mathit{if}_\sigma \not\triangleright i_\sigma$. By the definition of $\triangleright$, there are $s, t, u, v_1, \ldots, v_n$ and $a, b, c, d_1, \ldots, d_n$ for some $n \geq 0$ (depending on σ) such that $s \triangleright a$, $t \triangleright_\sigma b$, $u \triangleright_\sigma c$ and $v_i \triangleright d_i$ but $\mathit{if}\, s\, t\, u\, v_1\, \ldots\, v_n \not\triangleright_\tau i\, a\, b\, c\, d_1\, \ldots\, d_n$ for $\tau = \iota$ or $\tau = o$. In addition, by Lemma 5.11 and definition of $\triangleright$, we know that $[t][v_1] \ldots [v_n] \triangleright_\tau bd_1 \ldots d_n$ and $[u][v_1] \ldots [v_n] \triangleright_\tau cd_1 \ldots d_n$. Case analysis on τ:

  - $\tau = o$: If $a = 1$, then we have $\neg[s] \notin E$ and $\mathit{if}\, s\, t\, u\, v_1 \ldots v_n \not\triangleright b\, d_1 \ldots d_n$. Suppose $b\, d_1 \ldots d_n = 1$. Then, using N3, $\neg(\mathit{if}[s][t][u][v_1] \ldots [v_n]) \in E$ and $\neg[[t][v_1] \ldots [v_n]] \notin E$. By definition of $[\cdot]$ and $\mathcal{E}_{\text{IF}}$ with accessibility context $\neg([][v_1] \ldots [v_n])$ we know that either $[s] \in E$ and $\neg[[t][v_1] \ldots [v_n]] \in E$ contradicting $\neg[[t][v_1] \ldots [v_n]] \notin E$ or $\neg[s] \in E$ and $\neg[[u][v_1] \ldots [v_n]) \in E$ contradicting $\neg[s] \notin E$. Suppose $bd_1 \ldots d_n = 0$. Then $\mathit{if}[s][t][u][v_1] \ldots [v_n] \in E$ and $[[t][v_1] \ldots [v_n]] \notin E$. By $\mathcal{E}_{\text{IF}}$ with accessibility context $[][v_1] \ldots [v_n]$, we know that either $[s] \in E$ and $[[t][v_1] \ldots [v_n]] \in E$ contradicting $[[t][v_1] \ldots [v_n]] \notin E$ or $\neg[s] \in E$ and $[[u][v_1] \ldots [v_n]] \in E$ contradicting $\neg[s] \notin E$.

    The case for $a = 0$ is analogous.

  - $\tau = \iota$ i.e., $b' = b\, d_1 \ldots d_n$ and $c' = c\, d_1 \ldots d_n$ are discriminants.

    If $a = 1$, then we have $\neg[s] \notin E$ and $\mathit{if}\, s\, t\, u\, v_1 \ldots v_n \not\triangleright b'$. By the definition of $\triangleright_\iota$ and N3 we know that $\mathit{if}[s][t][u][v_1] \ldots [v_n]$ is discriminating but $\mathit{if}[s][t][u][v_1] \ldots [v_n] \notin b'$. In particular, this means that there is a term v' such that $v' \triangleright b'$ and $\mathit{if}[s][t][u][v_1] \ldots [v_n] \sharp [v']$. By definition of $[\cdot]$ and $\mathcal{E}_{\text{IF}}$ with accessibility context $[][v_1] \ldots [v_n] \neq [v']$ or $[v'] \neq [][v_1] \ldots [v_n]$, we know that either $\neg[s] \in E$ and $[[u]\, [v_1] \ldots [v_n]] \sharp [v']$ contradicting $\neg[s] \notin E$ or $[s] \in E$ and $[[t][v_1] \ldots [v_n]] \sharp [v']$ contradicting $[t][v_1] \ldots [v_n] \triangleright b'$ and $v' \triangleright b'$, i.e., that both, $[[t][v_1] \ldots [v_n]]$ and $[v']$ are in b' which is not possible because of the disequation.

    The case for $a = 0$ is analogous.

- $\exists_\sigma \triangleright e_\sigma$. Assume $\exists_\sigma \not\triangleright e_\sigma$. Then, there is a term $s : \sigma o$ and a function $f \in \mathcal{D}(\sigma o)$ such that $s \triangleright f$ but $\exists s \not\triangleright ef$.

  Suppose $fa = 0$ for all $a \in \mathcal{D}\sigma$. Then $ef = 0$ and $\exists[s] \in E$ by N3. By $\mathcal{E}_\exists$ and N2, there is a variable x such that $[sx] \in E$. By Lemma 5.4 and Lemma 5.12 there is some value $b \in \mathcal{D}\sigma$ such that $x \triangleright b$. By the definition of $\triangleright$, $sx \triangleright fb$ and $sx \not\triangleright 0$. Hence, $fb \neq 0$ contradicting the assumption that f is a constant 0 function.

Suppose $fa = 1$ for some $a \in \mathcal{D}\sigma$. Then, $ef = 1$ and $\neg(\exists[f]) \in E$ by N3. By Proposition 5.10, there is some $t \in \Lambda^{\mathcal{S}}_{\sigma}$ such that $a \triangleright t$. By $s \triangleright f$, we know that $st \triangleright fa$. By the definition, $\neg[st] \notin E$. On the other hand, by $\mathcal{E}_{\neg\exists}$ and N2, $\neg[su] \in E$ for all $u \in \Lambda^{\mathcal{S}}_{\sigma}$. Contradiction.

- $=_\sigma \triangleright q_\sigma$. Assume for contradiction $=_\sigma \not\triangleright q_\sigma$. Then, there are terms s, t and values a, b such that $s \triangleright a$, $t \triangleright b$ but $s = t \not\triangleright qab$.

  Suppose $a = b$. Then $[s]\sharp[t]$ by N3 and $s, t \triangleright a$. Hence, $s \parallel t$ by Lemma 5.12. Contradiction by Lemma 5.4(1).

  Suppose $a \neq b$. Then by N3, $[s] = [t] \in E$. $a = b$ by Lemma 5.11 and by Lemma 5.19. Contradiction. □

We say that an assignment into an $\mathcal{S}$-frame $\mathcal{D}$ is *admissible* if $c \triangleright \mathcal{I}c$ for all c in $\mathcal{S}$ and $x \triangleright \mathcal{I}x$ for all variables x.

5.2.4 Model Existence Theorem

Before we show that every evident set has a model, we prove the following lemma which helps to close the gap between assignments and interpretations.

Lemma 5.21. *Let $\mathcal{S}$ be a signature, $\mathcal{D}$ be an $\mathcal{S}$-frame, θ be a substitution and $\mathcal{I}$ be an admissible assignment into $\mathcal{D}$. Suppose $\theta x \triangleright \mathcal{I}x$ for every $x \in \operatorname{Dom}\theta$. Then $s \in \operatorname{Dom}\hat{\mathcal{I}}$ and $\hat{\theta}s \triangleright \hat{\mathcal{I}}s$ for every $\mathcal{S}$-term s.*

Proof. By induction on s. Suppose s is a variable x with $x \notin \operatorname{Dom}\theta$. Then $x \triangleright \mathcal{I}x$ by the admissibility of $\mathcal{I}$ and $\hat{\theta}x \triangleright \hat{\mathcal{I}}x$ by the definition of $\hat{\mathcal{I}}$ and by S1. If s is a variable x with $x \in \operatorname{Dom}\theta$ then $\theta x \triangleright \mathcal{I}x$ by assumption and so $\hat{\theta}s \triangleright \hat{\mathcal{I}}s$ by S1. If s is a logical constant c, then $\hat{\theta}s \triangleright \hat{\mathcal{I}}s$ by admissibility of $\mathcal{I}$, S4 and Lemma 5.11. The case where s is an application term follows from the inductive hypotheses, S2 and the definitions of $\hat{\mathcal{I}}$ and $\triangleright$. Finally, suppose s is of the form $\lambda x.t$ where $x : \sigma$ and $t \in \Lambda^{\mathcal{S}}_{\tau}$. Let $u \triangleright_\sigma a$ be given. We prove $(\hat{\theta}(\lambda x.t))u \triangleright (\hat{\mathcal{I}}(\lambda x.t))a$. Applying the inductive hypothesis to t with θ^x_u and $\mathcal{I}^x_a$, we have that $t \in \operatorname{Dom}\widehat{\theta^x_u}$ and $\widehat{\theta^x_u}t \triangleright \widehat{\mathcal{I}^x_a}t$. By S3 $[(\hat{\theta}(\lambda x.t))u]$ is $[\widehat{\theta^x_u}t]$. Two applications of Lemma 5.11 complete the proof. □

Now, we can finally prove Theorem 5.1, the Model Existence Theorem:

Proof of Theorem 5.1. Let $\mathcal{D}$ be defined using discriminants and the possible values relation as shown in the last section. We construct an $\mathcal{S}$-assignment $\mathcal{I}$ as follows: For each $c \in \mathcal{S}$, we define $\mathcal{I}c$ to be some value a such that $c \triangleright a$. The existence of such values follows from Lemma 5.20. By Lemma 5.4(2) and Lemma 5.12, we know that there is a value b for each variable x such that $x \triangleright b$. We define $\mathcal{I}x$ to be b. $\mathcal{I}$ is clearly admissible and $\emptyset$ trivially fulfills the condition of Lemma 5.21. Hence, $\mathcal{I}$ is an $\mathcal{S}$-interpretation and $s \triangleright \hat{\mathcal{I}}s$ by Lemma 5.11 and S4. It remains to show that $\mathcal{I} \models t$ for each $t \in E$. Note that t is normal and thus

$t = [t]$. Suppose $t \in$ *wff*$(\mathcal{S})$. We know that $t \triangleright \hat{\mathcal{I}}t$ and $t \not\triangleright 0$ by the definition of $\triangleright$. Hence $\hat{\mathcal{I}}t$ must be 1. Suppose $t = \neg u$ for some $u \in$ *wff*$(\mathcal{S})$. We know that $u \triangleright \hat{\mathcal{I}}u$ and $u \not\triangleright 1$. Hence, $\hat{\mathcal{I}}u \neq 1$. Suppose $t = (u \neq v)$ for some $u, v \in$ *wff*(S) and assume for contradiction that $\hat{\mathcal{I}}u = \hat{\mathcal{I}}v$. Then, $u, v \triangleright \hat{\mathcal{I}}u$ and by Lemma 5.12, $u \parallel v$. Contradiction by Lemma 5.4(1). $\square$

5.3 Standard Models

In the last section we showed that for arbitrary signatures $\mathcal{S}$, a set $E \subseteq$ *qwff*$(\mathcal{S})$ which fulfills the corresponding evidence conditions has a model over a certain $\mathcal{S}$-frame $\mathcal{D}$. This frame might be nonstandard.

In this section, we want to investigate what happens with the Model Existence Theorem if we only consider assignments into standard frames. The general idea from the last section including most of the proofs still work. However, there are lemmas which we need to modify. Instead of copying the last section, we will only discuss what needs to be changed and why it needs to be changed.

Assume $\mathcal{S} = \mathcal{LC}$. As already mentioned, it is a well known fact that we cannot get completeness (which is our goal in this chapter) with respect to standard models. This means that we cannot always construct a model over a standard frame for arbitrary evident sets. One solution to this problem is to only consider certain subsets of $\mathcal{LC}$ as possible candidates for the signatures. Unfortunately, this upper bound does not suffice.

Assume $\mathcal{S} = \{\forall_o\}$. We consider the branch $\{\forall_o x.x\}$. There is no $\mathcal{S}$-interpretation into a standard frame that satisfies this branch: The identity function at type o returns 0 for 0 as its argument. However, we cannot refute this branch. The only possibility is to add formulas to the branch consisting of variables and $\forall_o$. The problem is that we are not able to express 0 using $\mathcal{S}$-terms. To solve this problem, one might consider removing $\forall_o$ from the set of possible logical constants. Another less restricting solution is to require certain logical constants to always be elements of the signature. These logical constants should have the property that they can be used to express both, 0 and 1. Examples for such candidates are $\{\neg, \top\}$ or $\{=_o, \bot\}$. We decided to take the simplest solution: From now on, we only consider signatures in this section which contain at least $\top$ and $\bot$.

It remains to identify the logical constants which can prevent us from constructing standard models. The problem here is, similar to the problem just solved, that there are values which have are not related to any term. More precisely, the possible values relations are not surjective anymore for arbitrary types, i.e., Proposition 5.10 does not hold anymore in general. How are the possible values relations in the context of standard models (denoted by $\triangleright^{\mathsf{s}}$) defined? At first sight, they have not changed at all:

$$\begin{aligned}
s \triangleright^{\mathsf{s}}_o 0 \quad &:\Longleftrightarrow \quad [s] \notin E \\
s \triangleright^{\mathsf{s}}_o 1 \quad &:\Longleftrightarrow \quad \neg[s] \notin E \\
s \triangleright^{\mathsf{s}}_\iota a \quad &:\Longleftrightarrow \quad ([s] \text{ discriminating } \Longrightarrow [s] \in a) \\
s \triangleright^{\mathsf{s}}_{\sigma\tau} f \quad &:\Longleftrightarrow \quad st \triangleright^{\mathsf{s}}_\tau fa \text{ whenever } t \triangleright^{\mathsf{s}}_\sigma a
\end{aligned}$$

The important difference is that the frame is already fixed, i.e., there is no mutual induction going on:

$$\begin{array}{rcl} \mathcal{D}o & := & \{0,1\} \\ \mathcal{D}\iota & := & \text{set of all discriminants of } E \\ \mathcal{D}(\sigma\tau) & := & \mathcal{D}\sigma \to \mathcal{D}\tau \end{array}$$

Although we cannot preserve the surjectivity of the possible values relations, we can at least give certain types where they are still surjective:

Proposition 5.22. *Let $\mathcal{S}$ be a signature restricted as described in this section and let E be an $\mathcal{S}$-evident set. Let $\sigma \in \{o, \iota\}$. For any value $a \in \mathcal{D}\sigma$, there is a term $s \in \Lambda_\sigma^{\mathcal{S}}$ such that $s \rhd_\sigma^{\mathsf{s}} a$.*

Proof. Suppose $\sigma = o$. $\{\top, \bot\} \subseteq \mathcal{S}$ by assumption. It is easy to see that $\top \rhd^{\mathsf{s}} 1$ and $\bot \rhd^{\mathsf{s}} 0$ using $\mathcal{E}_\top$, $\mathcal{E}_\bot$ and N3. The case for $\sigma = \iota$ is identical to the proof of Proposition 5.10. □

Given this proposition, we know that $\rhd^{\mathsf{s}}$ is at least surjective for types ι and o. As a consequence, all proofs in the last section which used Proposition 5.10 must use Proposition 5.22 now and are restricted to base types. This affects the possible logical constants as follows:

1. In Lemma 5.17 and Lemma 5.18, we use all terms at arbitrary types to show that some function is a constant 0 function. Consequently, we need to restrict choice and description to base types, i.e., we only allow ε_σ and ι_σ for $\sigma \in \{\iota, o\}$

2. In Lemma 5.19, we consider $=_{\tau_1 \ldots \tau_n \mu}$ for arbitrary τ_i. We now need to restrict these τ_i to the base types. For example $=_{\iota(\iota\iota)}$ is still a possible candidate for being an element of the signature but $=_{(\iota\iota)\iota}$ is not.

3. As a consequence of (1) and (2), we are restricted in the same way in Lemma 5.20 because we use these lemmas there. Furthermore, we use Proposition 5.10 to prove $\exists_\sigma \rhd e_\sigma$. Hence, we also need to restrict the existential quantifier to base types.

Altogether, we found the following upper and lower bounds:

Theorem 5.23. *Let $\mathcal{S}$ be a signature such that $\mathcal{S}$ contains at least $\top$ and $\bot$ and contains at most $\exists_\sigma, \iota_\sigma, \varepsilon_\sigma$ or $=_{\tau_1 \ldots \tau_n \mu}$ if $\sigma, \tau_1 \ldots \tau_n, \mu$ are base types. Let $E \subseteq$ qwff$(\mathcal{S})$ be an evident set. Then, there is a standard frame $\mathcal{D}$ and an interpretation $\mathcal{I}$ into $\mathcal{D}$ such that $\mathcal{I} \models s$ for every $s \in E$.*

5.4 Abstract Consistency & Completeness

In the last two sections, we showed how to construct models for evident sets. It remains to show how to connect evident sets (which can possibly be infinite) to nonclosed branches (which can only be finite). This will finally allow us to prove completeness of a tableau system $T_{\mathcal{S}}$ for some signature $\mathcal{S}$. We will use a technique called *abstract consistency* which was first used by Smullyan [24] and later by several other authors, also for example in [9] or [8].

$\mathcal{C}_{\text{FE}}$	If $s \neq_{\sigma\tau} t$ is in A, then $A \cup \{[sx] \neq [tx]\}$ is in Γ for some variable x.
$\mathcal{C}_{\text{BE}}$	If $s \neq_{o} t$ is in E, then $A \cup \{s, \neg t\}$ or $A \cup \{\neg s, t\}$ is in Γ.
$\mathcal{C}_{\text{DEC}}$	If $xs_1 \ldots s_n \neq_{\iota} xt_1 \ldots t_n$ is in A for some $n \geq 0$ and some variable x, then $A \cup \{s_i \neq t_i\}$ is in Γ for some $1 \leq i \leq n$.
$\mathcal{C}_{\text{MAT}}$	If $xs_1 \ldots s_n$ and $\neg xt_1 \ldots t_n$ are in A for some $n \geq 0$ and some variable x, then $A \cup \{s_i \neq t_i\}$ is in Γ for some $1 \leq i \leq n$.
$\mathcal{C}_{\bot}$	$\bot$ is not in A.
$\mathcal{C}_{\top}$	$\neg\top$ is not in A.
$\mathcal{C}_{\vee}$	If $s \vee t$ is in E, then either $A \cup \{s\}$ is in Γ 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.
$\mathcal{C}_{\neg\neg}$	If $\neg\neg s$ is in A, then $A \cup \{s\}$ is in Γ.
$\mathcal{C}_{\exists}$	If $\exists s$ is in A, then $A \cup \{[sx]\}$ is in Γ for some variable x.
$\mathcal{C}^{\mathcal{S}}_{\neg\exists}$	If $\neg(\exists_{\sigma} s)$ is in A, then $A \cup \{\neg[st]\}$ is in Γ for all normal $\mathcal{S}$-terms t of type σ.
$\mathcal{C}_{\text{BQ}}$	If $s =_{\tau_1 \ldots \tau_n o} t$ is in A for some $n \geq 0$, then either $A \cup [su_1 \ldots u_n], [tu_1 \ldots u_n]\}$ is in Γ or $\{A \cup \neg[su_1 \ldots u_n], \neg[tu_1 \ldots u_n]\}$ is in Γ for all normal $u_i \in \Lambda^{\mathcal{S}}_{\tau_i}$ for $1 \leq i \leq n$.
$\mathcal{C}_{\text{CON}}$	If $s =_{\tau_1 \ldots \tau_n \iota} t$ and $u \neq_{\iota} v$ are in A for some $n \geq 0$, then either $A \cup \{[sw_1 \ldots w_n] \neq u, [tw_1 \ldots w_n] \neq u\}$ is in Γ or $A \cup \{[sw_1 \ldots w_n] \neq v, [tw_1 \ldots w_n] \neq v\}$ is in Γ for all normal $w_i \in \Lambda^{\mathcal{S}}_{\tau_i}$ for $1 \leq i \leq n$.
$\mathcal{C}_{\text{IF}}$	If $C[\mathit{if}\, s\, t\, u]$ is in A and C is an accessibility context, then $A \cup \{s, [C[t]]\}$ is in Γ or $A \cup \{\neg s, [C[u]]\}$ is in Γ.
$\mathcal{C}^{\mathcal{S}}_{\iota}$	If $C[\iota_{\sigma} s]$ is in A and C is an accessibility context, then either $A \cup \{\neg[st]\}$ is in Γ for all normal $\mathcal{S}$-terms t of type σ or $A \cup \{x \neq y, [sx], [sy]\}$ is in Γ for some variables x and y or $A \cup \{[s(\iota s)]\}$ is in Γ.
$\mathcal{C}_{\text{DEC}_{\iota}}$	If $\iota s_1 \ldots s_n \neq_{\iota} \iota t_1 \ldots t_n$ is in A for some $n \geq 0$, then $A \cup \{s_i \neq t_i\}$ is in Γ for some $1 \leq i \leq n$.
$\mathcal{C}_{\text{MAT}_{\iota}}$	If $\iota s_1 \ldots s_n$ and $\neg\iota t_1 \ldots t_n$ are in A for some $n \geq 0$, then $A \cup \{s_i \neq t_i\}$ is in Γ for some $1 \leq i \leq n$.
$\mathcal{C}^{\mathcal{S}}_{\varepsilon}$	If $C[\varepsilon_{\sigma} s]$ is in A and C is an accessibility context, then either $A \cup \{\neg[st]\}$ is in Γ for all normal $\mathcal{S}$-terms t of type σ or $A \cup \{[s(\varepsilon s)]\}$ is in Γ.
$\mathcal{C}_{\text{DEC}_{\varepsilon}}$	If $\varepsilon s_1 \ldots s_n \neq_{\iota} \varepsilon t_1 \ldots t_n$ is in A for some $n \geq 0$, then $A \cup \{s_i \neq t_i\}$ is in Γ for some $1 \leq i \leq n$.
$\mathcal{C}_{\text{MAT}_{\varepsilon}}$	If $\varepsilon s_1 \ldots s_n$ and $\neg\varepsilon t_1 \ldots t_n$ are in A for some $n \geq 0$, then $A \cup \{s_i \neq t_i\}$ is in Γ for some $1 \leq i \leq n$.

Figure 5.2: Abstract consistency conditions (must hold for every $A \in \Gamma$)

Definition 5.24. Let $\mathcal{S}$ be a signature. A set of $\mathcal{S}$-branches Γ is an abstract consistency class iff Γ fulfills the conditions from Figure 5.2.

Lemma 5.25. *Let Γ be an abstract consistency class with respect to a signature $\mathcal{S}$ and $A \in \Gamma$. Then, there exists an evident set E such that $A \subseteq E$.*

Proof. Let $u_1, u_2, u_3, \ldots$ be an enumeration of all normal quasi-$\mathcal{S}$-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 four subcases.

1. If u_n is of the form $\exists_\sigma s$, then choose A_{n+1} to be $B \cup \{[sx]\} \in \Gamma$ for some variable x. This is possible since Γ satisfies $\mathcal{C}_\exists$.

2. If u_n is of the form $s \neq_{\sigma\tau} t$, then choose A_{n+1} to be $B \cup \{[sx] \neq_\tau [tx]\} \in \Gamma$ for some variable x. This is possible by $\mathcal{C}_{\text{FE}}$.

3. Suppose u_n is of the form $C[\iota s]$ where C is an accessibility context and $B \cup \{[sx], [sy], x \neq y\}$ is in Γ for some variables x and y. We then choose A_{n+1} to be $B \cup \{[sx], [sy], x \neq y\}$. This is possible by $\mathcal{C}_\iota$.

4. If none of the previous cases applies then let A_{n+1} be B.

Let $E := \bigcup_{n \in \mathbb{N}} A_n$. We prove E satisfies the evidence conditions.

$\mathcal{E}_{\text{FE}}$ Assume $s \neq_{\sigma\tau} t$ is in E. Let n be such that u_n is $s \neq_{\sigma\tau} t$. Let $r \geq n$ be such that u_n is in A_r. Since $A_n \cup \{u_n\} \subseteq A_r$, there is some variable x such that $[sx] \neq [tx]$ is in A_{n+1} and hence in E.

$\mathcal{E}_{\text{BE}}$ Assume $s \neq_o t$ is in E. Let n, m, j, k be such that $u_n = s$, $u_m = t$, $u_j = \neg s$ and $u_k = \neg t$. Let $r \geq n, m, j, k$ be such that $s \neq_o t$ is in A_r. By $\mathcal{C}_{\text{BE}}$ either $A_r \cup \{s, \neg t\}$ or $A_r \cup \{\neg s, t\}$ is in Γ. Assume $A_r \cup \{s, \neg t\}$ is in Γ. Since $A_n \cup \{s\} \subseteq A_r \cup \{s, \neg t\}$, we have $s \in E$. Since $A_k \cup \{\neg t\} \subseteq A_r \cup \{s, \neg t\}$, we have $\neg t \in E$. Next assume $A_r \cup \{\neg s, t\}$ is in Γ. Since $A_j \cup \{\neg s\} \subseteq A_r \cup \{\neg s, t\}$, we have $\neg s \in E$. Since $A_m \cup \{t\} \subseteq A_r \cup \{\neg s, t\}$, we have $t \in E$.

$\mathcal{E}_{\text{MAT}}$ Assume $xs_1 \ldots s_n$ and $\neg xt_1 \ldots t_n$ are in E where $n \geq 1$. For each $i \in \{1, \ldots, n\}$, let m_i be such that u_{m_i} is $s_i \neq t_i$. Let $r \geq m_1, \ldots, m_n$ be such that $xs_1 \ldots s_n$ and $\neg xt_1 \ldots t_n$ are in A_r. By $\mathcal{C}_{\text{MAT}}$ there is some $i \in \{1, \ldots, n\}$ such that $A_r \cup \{s_i \neq t_i\} \in \Gamma$. Since $A_{m_i} \cup \{s_i \neq t_i\} \subseteq A_r \cup \{s_i \neq t_i\}$, we have $(s_i \neq t_i) \in A_{m_i+1} \subseteq E$.

$\mathcal{E}_{\text{DEC}}$ Similar to $\mathcal{E}_{\text{MAT}}$

$\mathcal{E}_\bot$ Assume E contains $\bot$. Then A_n contains $\bot$ for some n contradicting $\mathcal{C}_\bot$.

$\mathcal{E}_\top$ Assume E contains $\neg\top$. Then A_n contains $\neg\top$ for some n contradicting $\mathcal{C}_\top$.

$\mathcal{E}_\vee$ Assume $s \vee t$ is in E. Let n, m be such that $u_n = s$ and $u_m = t$. Let $r \geq n, m$ be such that $s \vee t$ is in A_r. By $\mathcal{C}_\vee$, $A_r \cup \{s\} \in \Gamma$ or $A_r \cup \{t\} \in \Gamma$. In the first case, $A_n \cup \{s\} \subseteq A_r \cup \{s\} \in \Gamma$, and so $s \in A_{n+1} \subseteq E$. In the second case, $A_m \cup \{t\} \subseteq A_r \cup \{t\} \in \Gamma$, and so $t \in A_{m+1} \subseteq E$. Hence either s or t is in E.

$\mathcal{E}_{\neg\vee}$ Assume $\neg(s \vee t)$ is in E. Let n, m be such that $u_n = s$ and $u_m = t$. Let $r \geq n, m$ be such that $\neg(s \vee t)$ is in A_r. By $\mathcal{C}_{\neg\vee}$, $A_r \cup \{\neg s, \neg t\} \in \Gamma$. Since $A_n \cup \{\neg s\} \subseteq A_r \cup \{\neg s, \neg t\}$, we have $\neg s \in A_{n+1} \subseteq E$. Since $A_m \cup \{\neg t\} \subseteq A_r \cup \{\neg s, \neg t\}$, we have $\neg t \in A_{m+1} \subseteq E$.

$\mathcal{E}_{\neg\neg}$ Assume $\neg\neg s$ is in E. Let n be such that $u_n = s$. Let $r \geq n$ be such that $\neg\neg s$ is in A_r. By $\mathcal{C}_{\neg\neg}$, $A_r \cup \{s\} \in \Gamma$. Since $A_n \cup \{s\} \subseteq A_r \cup \{s\}$, we have $s \in A_{n+1} \subseteq E$.

$\mathcal{E}_\exists$ Assume $\exists s$ is in E. Let n be such that u_n is $\exists s$. Let $r \geq n$ be such that u_n is in A_r. Since $A_n \cup \{u_n\} \subseteq A_r$, there is some variable x such that $[sx]$ is in A_{n+1} and hence in E.

$\mathcal{E}_{\neg\exists}^{\mathcal{S}}$ Assume $\neg(\exists_\sigma s)$ is in E and $u \in \Lambda_\sigma^{\mathcal{S}}$ is normal. Let n be such that u_n is $\neg[su]$. Let $r \geq n$ be such that $\exists s$ is in A_r. By $\mathcal{C}_{\neg\exists}$, we know that $A_r \cup \{\neg[su]\}$ is in Γ. Hence, $\neg[su]$ is in A_{n+1} and also in E.

$\mathcal{E}_{\mathrm{BQ}}$ Assume $s =_{\tau_1 \ldots \tau_h o} t$ is in E for some $h \geq 0$ and $u_1 \in \Lambda_{\tau_1}^{\mathcal{S}}, \ldots, u_h \in \Lambda_{\tau_h}^{\mathcal{S}}$ are normal. Let u_n be $[sw_1 \ldots w_h]$, u_m be $[tw_1 \ldots w_h]$, u_j be $\neg[sw_1 \ldots w_h]$ and u_k be $\neg[tw_1 \ldots w_h]$. Let $r \geq m, n, j, k$ such that $s =_{\tau_1 \ldots \tau_h o} t \in A_r$. By $\mathcal{C}_{\mathrm{BQ}}$ either $A_r \cup \{u_n, u_m\}$ or $A_r \cup \{u_j, u_k\}$ is in Γ. Assume $A_r \cup \{u_n, u_m\}$ is in Γ. Since $A_n \cup \{u_n\} \subseteq A_r \cup \{u_n, u_m\}$, we have $u_n \in E$. Since $A_m \cup \{u_m\} \subseteq A_r \cup \{u_n, u_m\}$, we have $u_m \in E$. Next assume $A_r \cup \{u_j, u_k\}$ is in Γ. Since $A_j \cup \{u_j\} \subseteq A_r \cup \{u_j, u_k\}$, we have $u_j \in E$. Since $A_k \cup \{u_k\} \subseteq A_r \cup \{u_j, u_k\}$, we have $u_k \in E$.

$\mathcal{E}_{\mathrm{CON}}$ Assume $s =_{\tau_1 \ldots \tau_h \iota} t$ and $u \neq_\iota v$ are in E for some $h \geq 0$. Assume $w_1 \in \Lambda_{\tau_1}^{\mathcal{S}}, \ldots, w_h \in \Lambda_{\tau_h}^{\mathcal{S}}$ are normal. Let n, m, j, k be such that u_n is $[sw_1 \ldots w_h] \neq u$, u_m is $[tw_1 \ldots w_h] \neq u$, u_j is $[sw_1 \ldots w_h] \neq v$ and u_k is $[tw_1 \ldots w_h] \neq v$. Let $r \geq n, m, j, k$ be such that $s =_{\tau_1 \ldots \tau_h \iota} t$ and $u \neq_\iota v$ are in A_r. By $\mathcal{C}_{\mathrm{CON}}$ either $A_r \cup \{u_n, u_m\}$ or $A_r \cup \{u_j, u_k\}$ is in Γ. Assume $A_r \cup \{u_n, u_m\}$ is in Γ. Since $A_n \cup \{u_n\} \subseteq A_r \cup \{u_n, u_m\}$, we have $u_n \in A_{n+1} \subseteq E$. Since $A_m \cup \{u_m\} \subseteq A_r \cup \{u_n, u_m\}$, we have $u_m \in A_{m+1} \subseteq E$. Next assume $A_r \cup \{u_j, u_k\}$ is in Γ. By a similar argument we know u_j and u_k must be in E.

$\mathcal{E}_{\mathrm{IF}}$ Assume $C[\mathit{if}\, s\, t\, u]$ is in E and C is an accessibility context. Let n, m, j, k be such that $u_n = s$, $u_m = [C[t]]$, $u_j = \neg s$ and $u_k = [C[u]]$. Let $r \geq n, m, j, k$ such that $C[\mathit{if}\, s\, t\, u]$ is in A_r. By $\mathcal{C}_{\mathrm{IF}}$ either $A_r \cup \{u_n, u_m\}$ or $A_r \cup \{u_j, u_k\}$ is in Γ. Assume $A_r \cup \{u_n, u_m\}$ is in Γ. Since $A_n \cup \{u_n\} \subseteq A_r \cup \{u_n, u_m\}$, we have $u_n \in A_{n+1} \subseteq E$. Since $A_m \cup \{u_m\} \subseteq A_r \cup \{u_n, u_m\}$, we have $u_m \in A_{m+1} \subseteq E$. Next assume $A_r \cup \{u_j, u_k\}$ is in Γ. As for $\mathcal{E}_{\mathrm{CON}}$, we know by a similar argument that u_j and u_k must be in E.

$\mathcal{E}_\iota^{\mathcal{S}}$ Assume $C[\iota_\sigma s]$ is in E and C is an accessibility context. Let $u \in \Lambda_\sigma^{\mathcal{S}}$ be normal. Let n, m, j be such that u_n is $C[\iota_\sigma s]$, u_m is $\neg[su]$ and u_j is $[s(\iota s)]$. Let $r \geq n, m, j$ such that $C[\iota_\sigma s]$ is in A_r. A_r witnesses that there is some B such that $A_n \cup \{u_n\} \subseteq B$. Suppose $B \cup \{[sx], [sy], x \neq y\} \in \Gamma$ for some variables x and y. Then, by definition, $\{[sx], [sy], x \neq y\} \subseteq A_{n+1}$ and we are done. Otherwise, by $\mathcal{C}_\iota$ either $A_r \cup \{\neg[su]\} \in \Gamma$ or $A_r \cup \{[s(\iota s)]\} \in \Gamma$. In the first case, $A_m \cup \{\neg[su]\} \subseteq A_r \cup \{\neg[su]\}$ and hence $\neg[su] \in A_{m+1} \subseteq E$. In the second case, $A_j \cup \{[s(\iota s)]\} \subseteq A_r \cup \{[s(\iota s)]\}$ and hence $[s(\iota s)] \in A_{j+1} \subseteq E$.

$\mathcal{E}_{\mathrm{DEC}_\iota}$ Similar to $\mathcal{E}_{\mathrm{MAT}}$

$\mathcal{E}_{\mathrm{MAT}_\iota}$ Similar to $\mathcal{E}_{\mathrm{MAT}}$

$\mathcal{E}_\varepsilon^{\mathcal{S}}$ Assume $C[\varepsilon_\sigma s]$ is in E and C is an accessibility context. Let $u \in \Lambda_\sigma^{\mathcal{S}}$ be normal. Let n, m be such that u_n is $\neg[su]$ and u_m is $[s(\varepsilon s)]$. Let $r \geq n, m$ such that $C[\varepsilon_\sigma s]$ is in A_r. By $\mathcal{C}_\iota$ either $A_r \cup \{\neg[su]\} \in \Gamma$ or $A_r \cup \{[s(\varepsilon s)]\} \in \Gamma$. In the first case, $A_n \cup \{\neg[su]\} \subseteq A_r \cup \{\neg[su]\}$ and hence $\neg[su] \in A_{n+1} \subseteq E$. In the second case, $A_m \cup \{[s(\varepsilon s)]\} \subseteq A_r \cup \{[s(\varepsilon s)]\}$ and hence $[s(\varepsilon s)] \in A_{m+1} \subseteq E$.

$\mathcal{E}_{\mathrm{DEC}_\varepsilon}$ Similar to $\mathcal{E}_{\mathrm{MAT}}$

$\mathcal{E}_{\mathrm{MAT}_\varepsilon}$ Similar to $\mathcal{E}_{\mathrm{MAT}}$ □

We now show that the set of all branches which are not refutable is an abstract consistency class.

Lemma 5.26. *Let $\mathcal{S}$ be a signature. Let Γ_T be the set of all $\mathcal{S}$-branches which are not refutable under $T_{\mathcal{S}}$. Γ_T is an abstract consistency class.*

Proof. We need to check all conditions for an abstract consistency class. As this is very straightforward using the corresponding tableau rules, we only give two examples. Let $A \in \Gamma_T$.

$\mathcal{C}_\varepsilon$ Assume $C[\varepsilon_\sigma s] \in A$ for some accessibility context C but $A \cup \{[s(\varepsilon s]]\} \notin \Gamma_T$ and $A \cup \{\neg[st]\} \notin \Gamma_T$ for some normal $t \in \Lambda_\sigma^{\mathcal{S}}$. We know $A \cup \{[s(\varepsilon s]]\}$ and $A \cup \{\neg[st]\}$ are refutable. Hence, A is refutable using $\mathcal{T}_\varepsilon$ with t as the corresponding instantiation. Contradiction to $A \in \Gamma_T$.

$\mathcal{C}_\exists$ Assume $\exists_\sigma s \in A$ but $A \cup \{[sx]\} \notin A$ for all variables x of type σ. Then $A \cup \{[sx]\}$ is refutable for all variables $x : \sigma$. Hence, A is refutable using $\mathcal{T}_\exists$ and the finiteness of A.

□

We can finally show that our tableau system is complete.

Theorem 5.27 (Completeness)**.** *Let $\mathcal{S}$ be a signature and $A \subseteq$ qwff$(\mathcal{S})$ be a branch. A is refutable using $T_{\mathcal{S}}$ if A is unsatisfiable. Moreover, if $\mathcal{S}$ fulfills the conditions from Theorem 5.23, then A is refutable using $T_{\mathcal{S}}$ if A is unsatisfiable with respect to standard models.*

Proof. Suppose A is unsatisfiable but not refutable. Then, $A \in \Gamma_T$. By Lemma 5.26, there is an evident set E such that $A \subseteq E$. By Theorem 5.1, there is a model for E. Contradiction.

Suppose $\mathcal{S}$ fulfills the conditions from Theorem 5.23 and A is unsatisfiable with respect to standard models but not refutable. By Lemma 5.26, there is an evident set E such that $A \subseteq E$. By Theorem 5.23, there is a standard model for E. Contradiction. □

Since the second claim sounds weaker than the first claim, we consider the contraposition of the second caim: A is satisfiable with respect to standard models if A is not refutable using the tableau system. This means, that if it is impossible to refute a branch, there is not just *some* model which satisfies all formulas in A but a model into a standard frame.

6 Examples

In this chapter, we will give several examples showing how the new tableau rules work in practice. To make life as easy as possible, we assume the full signature and take β-reduction as normalization operator (it is easy to check that β-reduction fulfills all N properties).

6.1 Axiom of Choice

We start by proving the validity of the Axiom of Choice at type ι:

$$\exists f : (\iota o)\iota.\ \forall p : \iota o.\ (\exists x : \iota.\ px) \rightarrow p(fp)$$

Since we have neither the forall quantifier nor logical implication defined, we replace $\forall x.\ sx$ by $\neg(\exists x.\ \neg(sx))$ and $s \rightarrow t$ by $\neg s \vee t$:

$$\exists f.\ \neg(\exists p.\ \neg(\neg(\exists x.\ px) \vee p(fp)))$$

To prove the validity of this formula, we need to refute its negation:

$$\begin{array}{c|c}
\multicolumn{2}{c}{\neg(\exists f.\ \neg(\exists p.\ \neg(\neg(\exists x.\ px) \vee p(fp))))} \\
\multicolumn{2}{c}{\neg\neg(\exists p.\ \neg(\neg(\exists x.\ px) \vee p(\varepsilon p)))} \\
\multicolumn{2}{c}{\exists p.\ \neg(\neg(\exists x.\ px) \vee p(\varepsilon p))} \\
\multicolumn{2}{c}{\neg(\neg(\exists x.\ px) \vee p(\varepsilon p))} \\
\multicolumn{2}{c}{\neg\neg(\exists x.\ px)} \\
\multicolumn{2}{c}{\neg(p(\varepsilon p))} \\
\multicolumn{2}{c}{\exists x.\ px} \\
\multicolumn{2}{c}{px} \\
\multicolumn{2}{c}{x \neq_\iota \varepsilon p} \\
\hline
\begin{array}{c} \neg(px) \\ x \neq x \end{array} &
\begin{array}{c|c}
\multicolumn{2}{c}{p(\varepsilon p)} \\
\multicolumn{2}{c}{\varepsilon p \neq \varepsilon p} \\
\multicolumn{2}{c}{p \neq p} \\
\multicolumn{2}{c}{py \neq py} \\
\hline
py & \neg(py) \\
\neg(py) & py \\
y \neq y & y \neq y
\end{array}
\end{array}$$

We instantiated ε for f and applied $\mathcal{T}_\varepsilon$ to εp because it was accessible in the formula $x \neq \varepsilon p$. The tableau refutation for the Axiom of Description is very similar so we do not present it here. Instead, we turn to if-then-else.

6.2 Axiom of If-Then-Else

In the introduction, we presented two axioms defining if-then-else. We combine them into one single Axiom of if-then-else as follows:

$$\exists f : o\iota\iota\iota.\ \forall x : \iota, y : \iota.\ (f\top xy = x) \wedge (f\bot xy = y)$$

We need to replace $s \wedge t$ by $\neg(\neg s \vee \neg t)$ and $\forall$ as above and refute the negation of the formula (note that we also removed some double negations):

$$\begin{array}{c}
\neg(\exists f.\ \neg(\exists x, y.\ (f\top xy \neq x) \vee (f\bot xy \neq y)))\\
\neg\neg(\exists x, y.\ (\textit{if}\top xy \neq x) \vee (\textit{if}\bot xy \neq y))\\
\exists x, y.\ (\textit{if}\top xy \neq x) \vee (\textit{if}\bot xy \neq y)\\
\exists y.\ (\textit{if}\top xy \neq x) \vee (\textit{if}\bot xy \neq y)\\
(\textit{if}\top xy \neq x) \vee (\textit{if}\bot xy \neq y)\\
\hline
\begin{array}{c|c}
\begin{array}{c}
\textit{if}\top xy \neq x\\
\hline
\begin{array}{c|c}
\top & \neg\top\\
x \neq x & y \neq x
\end{array}
\end{array}
&
\begin{array}{c}
\textit{if}\bot xy \neq y\\
\hline
\begin{array}{c|c}
\bot & \neg\bot\\
x \neq y & y \neq y
\end{array}
\end{array}
\end{array}
\end{array}$$

We instantiated f by *if* and used $\mathcal{T}_{\text{IF}}$ once with accessibility context $[] \neq x$ and once with accessibility context $[] \neq y$.

Another possibility to prove this axiom is to use the fact "description implies if-then-else". Since the corresponding tableau refutation is very long we moved it to Appendix A.

6.3 Swapping Functions

In this section, we prove the existence of swapping functions using if-then-else, similar to Proposition 3.7. Swapping functions at type ι are described by the following formula:

$$\forall x : \iota, y : \iota.\ \exists f : \iota\iota.\ fx = y \wedge fy = x$$

Again, we rewrite the formula so that it only contains logical constants we can use:

$$\neg\exists x, y. \neg\exists f.\ \neg(fx \neq y \vee fy \neq x)$$

We refute the negation:

$$\begin{array}{c}
\neg\neg\exists x, y.\neg\exists f.\ \neg(fx \neq y \vee fy \neq x) \\
\exists x, y.\neg\exists f.\ \neg(fx \neq y \vee fy \neq x) \\
\exists y.\neg\exists f.\ \neg(fx \neq y \vee fy \neq x) \\
\neg\exists f.\ \neg(fx \neq y \vee fy \neq x) \\
\neg\neg((\mathit{if}(x = x)yx) \neq y \vee (\mathit{if}(x = y)yx) \neq x) \\
(\mathit{if}(x = x)yx) \neq y \vee (\mathit{if}(x = y)yx) \neq x \\
\hline
\begin{array}{c|c}
\begin{array}{c}
(\mathit{if}(x = x)yx) \neq y \\
\hline
\begin{array}{c|c}
x = x & x \neq x \\
y \neq y & x \neq y
\end{array}
\end{array}
&
\begin{array}{c}
(\mathit{if}(x = y)yx) \neq x \\
\hline
\begin{array}{c|c}
\begin{array}{c}
x = y \\
y \neq x \\
\hline
\begin{array}{c|c}
x \neq y & x \neq x \\
y \neq y & y \neq x
\end{array}
\end{array}
&
\begin{array}{c}
x \neq y \\
x \neq x
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}$$

As already shown in the proof of Proposition 3.7, we instantiated f by $\lambda z.\ \mathit{if}(x = z)yx$.

6.4 Skolem

Skolem functions can be used can be used to remove certain existential quantifiers in formulas. One possibility to describe the existence of Skolem functions at type ι is the following formula:

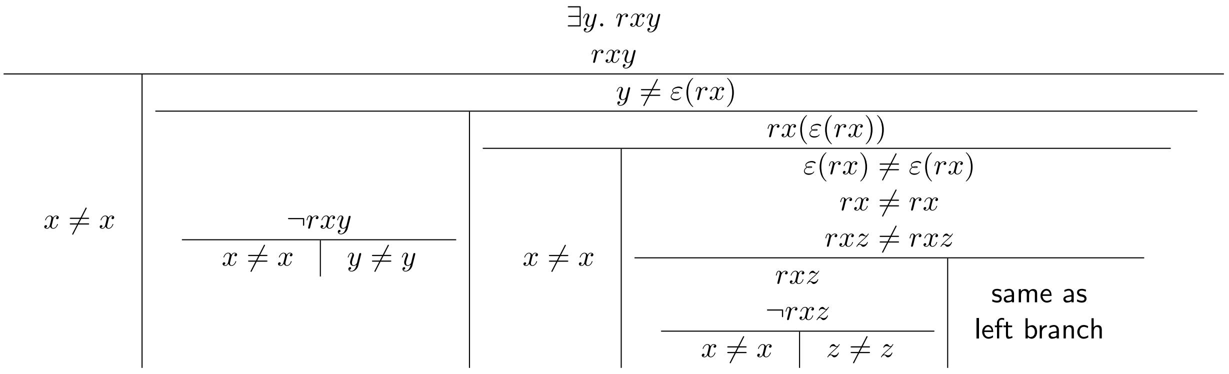
$$\forall r : \iota\iota o.\ (\forall x : \iota.\ \exists y : \iota.\ rxy) \rightarrow \exists f : \iota\iota.\ \forall x : \iota.\ rx(fx)$$

We rewrite this formula to

$$\neg\exists r.\ \neg((\exists x.\ \neg\exists y.\ rxy) \vee \exists f.\ \neg\exists x : \iota.\ \neg rx(fx))$$

and prove its validity by refuting its negation. The idea is that we can use a choice operator to choose the y for each rx. Hence, the instantiation for f will be $\lambda x.\ \varepsilon(rx)$:

$$\begin{array}{c}
\neg\neg\exists r.\ \neg((\exists x.\ \neg\exists y.\ rxy) \vee \exists f.\ \neg\exists x.\ \neg rx(fx)) \\
\exists r.\ \neg((\exists x.\ \neg\exists y.\ rxy) \vee \exists f.\ \neg\exists x.\ \neg rx(fx)) \\
\neg((\exists x.\ \neg\exists y.\ rxy) \vee \exists f.\ \neg\exists x.\ rx(fx)) \\
\neg\exists x.\ \neg\exists y.\ rxy \\
\neg\exists f.\ \neg\exists x.\ \neg rx(fx) \\
\neg\neg\exists x.\ \neg rx(\varepsilon(rx)) \\
\exists x.\ \neg rx(\varepsilon(rx)) \\
\neg rx(\varepsilon(rx)) \\
\neg\neg\exists y.\ rxy
\end{array}$$



6.5 Choice Complement

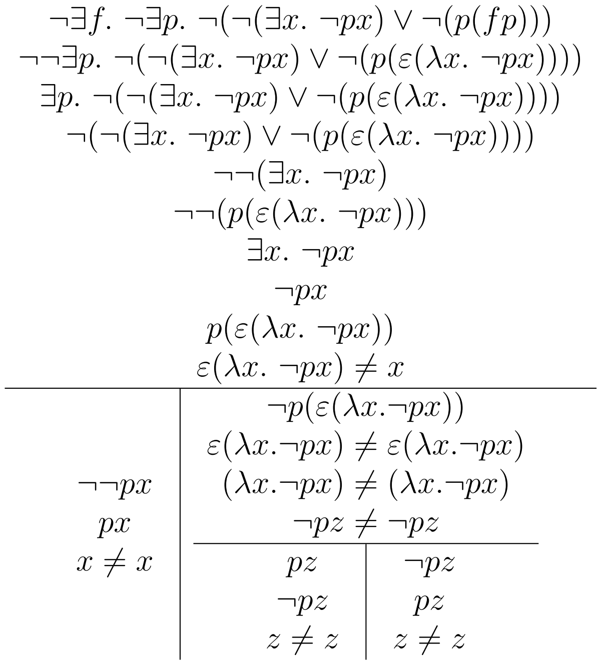
While choice functions choose for every nonempty set an element out of this set, there are also choice complement functions which choose for every set which is not full an element which is not in this set. We can describe choice complement at type ι using the following formula:

$$\exists f : (\iota o)\iota.\ \forall p : \iota o.\ (\exists x : \iota.\ \neg px) \to \neg(p(fp))$$

We rewrite this formula to

$$\exists f.\ \neg\exists p.\ \neg(\neg(\exists x.\ \neg px) \lor \neg(p(fp)))$$

and refute its negation. We use choice to prove the existence of choice complement and instantiate $\lambda p.\ \varepsilon(\lambda x.\ \neg px)$ for f, i.e., given some set we use choice to choose an element out of the complement of the set:


$\neg\exists f.\ \neg\exists p.\ \neg(\neg(\exists x.\ \neg px) \lor \neg(p(fp)))$
$\neg\neg\exists p.\ \neg(\neg(\exists x.\ \neg px) \lor \neg(p(\varepsilon(\lambda x.\ \neg px))))$
$\exists p.\ \neg(\neg(\exists x.\ \neg px) \lor \neg(p(\varepsilon(\lambda x.\ \neg px))))$
$\neg(\neg(\exists x.\ \neg px) \lor \neg(p(\varepsilon(\lambda x.\ \neg px))))$
$\neg\neg(\exists x.\ \neg px)$
$\neg\neg(p(\varepsilon(\lambda x.\ \neg px)))$
$\exists x.\ \neg px$
$\neg px$
$p(\varepsilon(\lambda x.\ \neg px))$
$\varepsilon(\lambda x.\ \neg px) \neq x$
$\neg\neg px$
px
$x \neq x$
$\neg p(\varepsilon(\lambda x.\neg px))$
$\varepsilon(\lambda x.\neg px) \neq \varepsilon(\lambda x.\neg px)$
$(\lambda x.\neg px) \neq (\lambda x.\neg px)$
$\neg pz \neq \neg pz$
pz
$\neg pz$
$z \neq z$
$\neg pz$
pz
$z \neq z$

7 Extensions

In this chapter, we will try to improve our tableau system by adding even more logical constants plus the corresponding tableau rules. These new logical constants will not necessarily be more powerful in terms of Chapter 3 but they will make our life easier when refuting formulas.

7.1 n-ary Choice

In Section 6.4, we have seen that we can prove the existence of Skolem functions at type ι using a choice operator. We expressed Skolem functions using the following formula:

$$\forall r : \iota\iota o.\ (\forall x : \iota.\ \exists y : \iota.\ rxy) \rightarrow \exists f : \iota\iota.\ \forall x : \iota.\ rx(fx)$$

Let us perform a *Gedankenexperiment*. Imagine that r does not map individuals to individuals but individuals to coordinates, i.e., to pairs of individuals. Pairs are commonly expressed using product types:

$$\forall r : \iota(\iota \times \iota)o.\ (\forall x : \iota.\ \exists y : (\iota \times \iota).\ rxy) \rightarrow \exists f : \iota(\iota \times \iota).\ \forall x : \iota.\ rx(fx)$$

Given this formula, it is easy to see that we can still use $\varepsilon_{\iota\times\iota}$ to prove its validity. Unfortunately, we do not have product types in our simply typed lambda calculus.

To solve this problem, we can use the well-known technique of *Currying*. Doing so results in the following formula:

$$\forall r : \iota\iota\iota o.\ (\forall x : \iota.\ \exists y : \iota,\ z : \iota.\ rxyz) \rightarrow \exists f : \iota\iota,\ g : \iota\iota.\ \forall x : \iota.\ rx(fx)(gx)$$

Is it still possible to prove the validity of this formula? It turns out that it is. The hard part is to find instantiations for f and g. We start by giving the instantiation for f:

$$f := \lambda x : \iota.\ \varepsilon(\lambda y : \iota.\ \exists z : \iota.\ rxyz)$$

Unfortunately, using the same idea for the instantiation of g does not work:

$$g' := \lambda x : \iota.\ \varepsilon(\lambda z : \iota.\ \exists y : \iota.\ rxyz)$$

The problem is that f and g' are completely independent. Imagine that ι represents $\{1, 2, 3, 4, 5\}$ and r represents a relation that contains the tuples (n, n, n) for all $n \in \iota$ plus the tuples $(1, 2, 3)$ and $(1, 4, 5)$. Depending on the choice function, it is possible that for the argument 1, f returns 2 and g' returns 5. However, the tuple $(1, 2, 5)$ is not in r.

The solution to this problem is to define g depending on the choice of f:

$$g := \lambda x : \iota.\ \varepsilon(\lambda z : \iota.\ rx(fx)z)$$

Given these two instantiation, it is possible (after some rewriting) to prove the validity of the Curried Skolem formula by refuting its negation using our tableau system.

Unfortunately, we cannot always use Currying. Consider the following example which makes again use of product types:

$$\forall r : \iota(\iota \times \iota)o.\ (\forall x : \iota.\ \exists y : \iota \times \iota.\ rxy) \rightarrow \forall x : \iota.\ rx(\varepsilon_{\iota\times\iota}(rx))$$

There is no obvious way to Curry the choice operator.

We will now present a solution to this problem. Moreover, this solution will provide a better way to prove the validity of curried formula from above. We introduce two new logical constants $\varepsilon_{2,1} : (\iota\iota o)\iota$ and $\varepsilon_{2,2} : (\iota\iota o)\iota$ where $\varepsilon_{2,1}$ takes the position of f and $\varepsilon_{2,1}$ takes the position of g as follows:

$$\begin{array}{lcl} f & := & \lambda x : \iota.\ \varepsilon_{2,1}(rx) \\ g & := & \lambda x : \iota.\ \varepsilon_{2,2}(rx) \end{array}$$

If we want to do the refutation using these two instantiations, we need to give tableau rules for $\varepsilon_{2,1}$ and $\varepsilon_{2,2}$. We can find them by looking at the fundamental property of $\varepsilon_{2,1}$ and $\varepsilon_{2,2}$, i.e., at the predicate which must hold for two functions if they want to be represented by the two logical constants:

$$\mathcal{P}_{\varepsilon_2}(f, g) = \forall r \in \mathcal{D}(\iota\iota o).\ (\exists x, y \in \mathcal{D}\iota.\ rxy) \rightarrow r(fr)(gr)$$

Using this axiom and the ideas we already developed for $\mathcal{T}_\varepsilon$, we present the following two new tableau rules:

$$\begin{array}{lcl} \mathcal{T}_{\varepsilon_{2,1}} & \dfrac{C[\varepsilon_{2,1}s]}{\neg[stu] \mid [s(\varepsilon_{2,1}s)(\varepsilon_{2,2}s)]} & \begin{array}{l} C \text{ accessibility context,} \\ t, u \in \Lambda^{\mathcal{S}}_\iota \text{ normal} \end{array} \\ \mathcal{T}_{\varepsilon_{2,2}} & \dfrac{C[\varepsilon_{2,2}s]}{\neg[stu] \mid [s(\varepsilon_{2,1}s)(\varepsilon_{2,2}s)]} & \begin{array}{l} C \text{ accessibility context,} \\ t, u \in \Lambda^{\mathcal{S}}_\iota \text{ normal} \end{array} \end{array}$$

As we can see, there is again a dependence between both choice operators: We cannot add $\varepsilon_{2,1}$ to the signature without adding $\varepsilon_{2,2}$ and vice versa. This is not surprising since the predicate shown above is also defined on two functions. Moreover, we have already seen that there must be a connection between $\varepsilon_{2,1}$ and $\varepsilon_{2,2}$. The evidence conditions for $\mathcal{T}_{\varepsilon_{2,1}}$ and $\mathcal{T}_{\varepsilon_{2,2}}$ look as follows:

$\mathcal{E}^{\mathcal{S}}_{\varepsilon_{2,1}}$	If $C[\varepsilon_{2,1}s]$ is in E and C is an accessibility context, then either $[s(\varepsilon_{2,1}s)(\varepsilon_{2,2}s)]$ is in E, or $\neg[stu]$ is in E for all normal $\mathcal{S}$-terms t, u of type ι.
$\mathcal{E}^{\mathcal{S}}_{\varepsilon_{2,2}}$	If $C[\varepsilon_{2,2}s]$ is in E and C is an accessibility context, then either $[s(\varepsilon_{2,1}s)(\varepsilon_{2,2}s)]$ is in E, or $\neg[stu]$ is in E for all normal $\mathcal{S}$-terms t, u of type ι.

To show that $\varepsilon_{2,1} \parallel \varepsilon_{2,1}$ and $\varepsilon_{2,2} \parallel \varepsilon_{2,2}$ we also need decomposition and mating rules for both logical constants:

$$\mathcal{T}_{\mathrm{MAT}_{\varepsilon_{2,1}}}\ \frac{\varepsilon_{2,1}s_1\dots s_n,\ \neg\varepsilon_{2,1}t_1\dots t_n}{s_1\neq t_1 \mid \dots \mid s_n\neq t_n} \qquad \mathcal{T}_{\mathrm{DEC}_{\varepsilon_{2,1}}}\ \frac{\varepsilon_{2,1}s_1\dots s_n\neq_\iota \varepsilon_{2,1}t_1\dots t_n}{s_1\neq t_1 \mid \dots \mid s_n\neq t_n}$$

$$\mathcal{T}_{\mathrm{MAT}_{\varepsilon_{2,2}}}\ \frac{\varepsilon_{2,2}s_1\dots s_n,\ \neg\varepsilon_{2,2}t_1\dots t_n}{s_1\neq t_1 \mid \dots \mid s_n\neq t_n} \qquad \mathcal{T}_{\mathrm{DEC}_{\varepsilon_{2,2}}}\ \frac{\varepsilon_{2,2}s_1\dots s_n\neq_\iota \varepsilon_{2,2}t_1\dots t_n}{s_1\neq t_1 \mid \dots \mid s_n\neq t_n}$$

We will not present the corresponding evidence conditions here because they can be read off the tableau rules in a straightforward way. Instead, we turn to the possible values for $\varepsilon_{2,1}$ and $\varepsilon_{2,2}$. We first extend the notion of relevant terms to $\varepsilon_{2,1}$ and $\varepsilon_{2,2}$ the obvious way and observe that Lemma 5.15 also holds for $c\in\{\varepsilon_{2,1},\varepsilon_{2,2}\}$. We define $\Phi_{2,1}\in\mathcal{D}((\iota\iota o)\iota)$ and $\Phi_{2,2}\in\mathcal{D}((\iota\iota o)\iota)$:

$$\Phi_{2,1}f \;=\; \begin{cases}\text{some } b & \text{such that there is a } c \text{ such that } fbc=1 \text{ if } f^{\varepsilon_{2,1}} \text{ is empty}\\ & \text{and such a } b \text{ exists.}\\ \text{some } a & \text{such that } f^{\varepsilon_{2,1}}\triangleright a.\end{cases}$$

$$\Phi_{2,2}f \;=\; \begin{cases}\text{some } b & \text{such that } f(\Phi_{2,1}f)b=1 \text{ if } f^{\varepsilon_{2,2}} \text{ is empty and such a } b \text{ exists.}\\ \text{some } a & \text{such that } f^{\varepsilon_{2,2}}\triangleright a.\end{cases}$$

The idea is similar to the one we used for the first working instantiations of f and g in the Skolem example above: We use $\Phi_{2,1}$ to define $\Phi_{2,2}$. The following two lemmas correspond to Lemmas 5.16 and 5.17:

Lemma 7.1. *We have $\varepsilon_{2,1}\triangleright\Phi_{2,1}$ and $\varepsilon_{2,2}\triangleright\Phi_{2,2}$*

Proof. We only show $\varepsilon_{2,1}\triangleright\Phi_{2,1}$. $\varepsilon_{2,2}\triangleright\Phi_{2,2}$ can analogously be proven. Assume $\varepsilon_{2,1}\not\triangleright\Phi_{2,1}$. Then, there are s,f such that $s\triangleright f$ but $\varepsilon_{2,1}s\not\triangleright\Phi_{2,1}f$. By Lemma 5.15 $\varepsilon_{2,1}[s]$ is accessible in E. Hence $\varepsilon_{2,1}s\in f^{\varepsilon_{2,1}}$. By definition, there is some a such that $\Phi_{2,1}f=a$ and $f^{\varepsilon_{2,1}}\triangleright a$. Thus $\varepsilon_{2,1}s\triangleright a$, a contradiction. □

Lemma 7.2. *$\mathcal{P}_{\varepsilon_2}(\Phi_{2,1},\Phi_{2,2})$ holds. That is, $\Phi_{2,1}$ and $\Phi_{2,2}$ as defined above are choice functions for binary relations.*

Proof. Let $f\in\mathcal{D}(\iota\iota o)$ be a function and $a,b\in\mathcal{D}\iota$ be such that $fab=1$. Suppose $f(\Phi_{2,1}f)(\Phi_{2,2}f)=0$. Then, by definition of $\Phi_{2,1}$ and $\Phi_{2,2}$, at least one $f^{\varepsilon_{2,i}}$ must be nonempty for $i\in\{1,2\}$. We choose such an i and some $\varepsilon_{2,i}s\in f^{\varepsilon_{2,i}}$. We will show a contradiction. By $\mathcal{E}_{\varepsilon_{2,i}}$ there are two possibilities:

1. $[s(\varepsilon_{2,1}s)(\varepsilon_{2,2}s)]\in E$: In this case $s(\varepsilon_{2,1}s)(\varepsilon_{2,2}s)\not\triangleright 0$. On the other hand, $s\triangleright f$, $\varepsilon_{2,1}\triangleright\Phi_{2,1}$ and $\varepsilon_{2,2}\triangleright\Phi_{2,2}$ (by Lemma 7.1) and so $s(\varepsilon_{2,1}s)(\varepsilon_{2,2}s)\triangleright f(\Phi_{2,1}f)(\Phi_{2,2}f)$. This contradicts our assumption that $f(\Phi_{2,1}f)(\Phi_{2,2}f)=0$.

2. $\neg[stu]\in E$ for every normal $t,u\in\Lambda^{\mathcal{S}}_\iota$. By Proposition 5.10 and Lemma 5.11 there are normal terms $t',u'\in\Lambda^{\mathcal{S}}_\iota$ such that $t'\triangleright a$ and $u'\triangleright b$. Hence $\neg[s\,t'u']\in E$. By the

definition of $\rhd_o$, $s\,t'u' \not\rhd 1$. On the other hand, we know $s\,t'u' \rhd fab$ since $s \rhd f$, $t' \rhd a$ and $u' \rhd b$, contradicting the assumption that $fab = 1$. $\square$

Using these two lemmas, we see that we can easily integrate choice for binary relations at type ι into Theorem 5.1 (Model Existence Theorem). We omit the rest of the completeness proof here (Abstract Consistency and Completeness) because it does not reveal anything new. Instead, we show how we can generalize choice even further by considering relations of arbitrary arity at arbitrary types, i.e., terms $s : \sigma$ where $\sigma = \tau_1 \dots \tau_n o$ for some $n \geq 1$ and some types $\tau_1, ..., \tau_n$. In this case, we need n choice constants $\varepsilon_{\sigma,i} : (\tau_1 \dots \tau_n o)\tau_i$ for $1 \leq i \leq n$.

Consequently, we have $3n$ new tableau rules, three for each $\varepsilon_{\sigma,i}$

$$\mathcal{T}_{\varepsilon_{\sigma,i}} \quad \frac{C[\varepsilon_{\sigma,i}s]}{\neg[st_1 \dots t_n] \mid [s(\varepsilon_{\sigma,1}s) \dots (\varepsilon_{\sigma,n}s)]} \quad \begin{array}{l} C \text{ accessibility context,} \\ t_1, \dots, t_n \in \Lambda^{\mathcal{S}}_{\sigma} \text{ normal} \end{array}$$

$$\mathcal{T}_{\text{MAT}\varepsilon_{\sigma,i}} \quad \frac{\varepsilon_{\sigma,i}s_1 \dots s_m,\ \neg\varepsilon_{\sigma,i}t_1 \dots t_m}{s_1 \neq t_1 \mid \dots \mid s_m \neq t_m} \qquad \mathcal{T}_{\text{DEC}\varepsilon_{\sigma,i}} \quad \frac{\varepsilon_{\sigma,i}s_1 \dots s_m \neq_\iota \varepsilon_{\sigma,i}t_1 \dots t_m}{s_1 \neq t_1 \mid \dots \mid s_m \neq t_m}$$

together with the corresponding evidence conditions (again, we omit the conditions for decomposition and mating):

$\mathcal{E}^{\mathcal{S}}_{\varepsilon_{n,i}}$ If $C[\varepsilon_{\sigma,i}s]$ is in E and C is an accessibility context, then either $[s(\varepsilon_{\sigma,1}s) \dots (\varepsilon_{\sigma,n}s)]$ is in E, or $\neg[st_1 \dots t_n]$ is in E for all normal $\mathcal{S}$-terms $t_1 : \tau_1, \dots, t_n : \tau_n$.

It remains to give a possible value $\Phi_{\sigma,i}$ for each $\varepsilon_{\sigma,i}$. Moreover, these functions together should satisfy the following predicate:

$$\mathcal{P}_{\varepsilon_\sigma}(f_1, \dots, f_n) = \forall r \in \mathcal{D}(\tau_1 \dots \tau_n o).\ (\exists x_1 \in \mathcal{D}\tau_1, \dots, x_n \in \mathcal{D}\tau_n.\ rx_1 \dots x_n) \to r(f_1 r) \dots (f_n r)$$

Again, we extend the notion of relevant terms to each $\varepsilon_{\sigma,i}$ and define $\Phi_{\sigma,i}$ as follows:

$$\Phi_{\sigma,i}f \quad = \quad \begin{cases} \text{some } b & \text{such that } f(\Phi_{\sigma,1}f) \dots (\Phi_{\sigma,i-1}f)\, b\, c_{i+1} \dots c_n = 1 \text{ if } f^{\varepsilon_{\sigma,i}} \text{ is empty} \\ & \text{and such } c_{i+1} \dots c_n \text{ exist.} \\ \text{some } a & \text{such that } f^{\varepsilon_{\sigma,i}} \rhd a. \end{cases}$$

We can use these definitions to prove two lemmas similar to Lemma 7.1 and Lemma 7.2 and can finally integrate choice for relations of arity n at types $\tau_1, \dots, \tau_n$ into the Model Existence Theorem. For signatures as described in Section 5.3 and types $\tau_1, \dots, \tau_n \in \{\iota, o\}$, this even holds with respect to standard models. We omit these proofs here since they are essentially the same as above with some modified indices.

7.2 Restricting Instantiations

While this thesis was still being written, we published parts of it in [3]. We want to give a short overview of this paper. It is focused on two aspects.

First, we present a complete tableau calculus for higher-order logic with choice. For this purpose, we fixed the signature $\{=_\sigma, \forall_\sigma, \varepsilon_\sigma, \vee, \neg, \top\}$ where $\forall_\sigma$ can be seen as $\lambda x : \sigma o.\ \neg\exists y.\ \neg(xy)$. We mainly follow the completeness proof of this thesis but it was possible to avoid some of the problems we had here: Since we fixed a single signature, it was not necessary to define for example signature dependent terms or signature dependent frames. Moreover, we have $\neg$ and $=$ in the signature so we did not need to introduce quasiformulas or models.

The second aspect of the paper concentrates on restricting the instantiations for ε, $\forall$ and $=$. We found out that we can restrict the set of possible instantiations to a *universe* depending on the type as follows:

- $\bot$ and $\neg\bot$ (acting as $\top$) at type o
- discriminating terms at type ι, or a special logical constant $* : \iota$ if there are no discriminating terms
- terms only containing free variables which were already free on the branch before at all other types

While the restrictions at type o and type ι caused no problems, the restrictions at functional types had several impacts. It was necessary to define the compatibility relations and the possible values relations on the limited universes of terms. To see the consequences of these changes, consider Lemma 5.4. This lemma implies that $x \parallel x$ for all variables x. Applying the restriction results in $x \parallel x$ for all x *that are free in* E.

We use this lemma for example to prove the $\exists$-part in Lemma 5.20. This proof says "By $\mathcal{E}_\exists$ and N2, there is a variable x such that $[sx] \in E$. By Lemma 5.4...". However, can we really apply Lemma 5.4? No, we cannot. Depending on the normalization operator, it is possible that x is normalized away. For example when assuming β-normalization, the normal form of $(\lambda x.y)z$ is just y, i.e., z is not contained in the normal form anymore. In this case, we could replace z by an arbitrary term. Unfortunately, it is not possible to show this using the existing N- or S-properties. We needed to modify and extend them as follows:

- Add N5: $\mathcal{V}[s] \subseteq \mathcal{V}s$, i.e., the normalization operator never introduces additional free variables
- Modify S4: $[\hat{\theta}s] = [s]$ if $\theta x = x$ for all $x \in \mathrm{Dom}\,\theta \cap \mathcal{V}s$. This means we do not only consider the empty substitution but all substitutions that behave on a specific term like the empty substitution.
- Add S5: $[\hat{\theta}[s]] = [\hat{\theta}s]$, i.e. it does not matter if we normalize before substituting when normalizing the result again.

We need N5 to prove the modified version of Lemma 5.11 (which now only considers terms from the restricted universe). S4 and S5 are used to prove the following proposition:

Proposition 7.3. *If $x \notin \mathcal{V}[sx]$ then $[sx] = [st]$ for any term t.*

Proof. Let $\theta = \emptyset^x_t$. We compute

$$[sx] \overset{\text{N1}}{=} [[sx]] \overset{\text{S4}}{=} [\hat{\theta}[sx]] \overset{\text{S5}}{=} [\hat{\theta}(sx)] \overset{\text{S1,S2}}{=} [(\hat{\theta}s)t] \overset{\text{N2}}{=} [[\hat{\theta}s]t] \overset{\text{S4}}{=} [[s]t] \overset{\text{N2}}{=} [st] \quad \square$$

Another lemma that caused problems was Lemma 5.13. It says that $\mathcal{D}\sigma$ is nonempty for all types σ, i.e., that the $\mathcal{D}$ we constructed to show the Model Existence Theorem is really a frame. Recall the proof:

Proof of Lemma 5.13. We know that for any σ there is a variable $x : \sigma$. By Lemma 5.4, we know that $x \parallel x$. By Lemma 5.12 there is a value a such that $x \triangleright_\sigma a$. By the definition of $\mathcal{D}$, $a \in \mathcal{D}\sigma$. $\square$

As above, we have the problem that we do not know whether x is really free in E, i.e., it is possible that $\parallel$ is not defined on x. In such cases, the proof breakes down. As a solution to this problem, we used the choice operator and showed that $\varepsilon_\sigma(\lambda x : \sigma.\ \bot)$ is compatible to itself at all types σ. The claim still follows from Lemma 5.12.

In the context of this thesis, this is a suboptimal solution. It forces the signatures to contain at least ε at all types and $\bot$. It is future work to find a better solution.

7.3 Primitive Recursion and The Natural Numbers

In this section we want to propose tableau rules for a tableau system with support for the natural numbers as well as primitive recursion. Due to Gödel's first incompleteness theorem, we know that there is no chance of getting a corresponding complete proof system. We explicitly do not claim that these rules suffice to show completeness. Instead, it should be seen as a collection of thoughts which came up while this thesis was being written.

7.3.1 The Peano Axioms

We fix a type η and assume two new logical constants $0 : \eta$ (zero) and $S : \eta\eta$ (the successor function). Following the ideas by Peano, we use the following three axioms to specify η as a type corresponding to an infinite set:

$$\forall x : \eta.\ 0 \neq S(x)$$
$$\forall x, y : \eta.\ Sx = Sy \to x = y$$
$$\forall p : \eta o.\ p0 \to ((\forall x : \eta.\ px \to p(Sx)) \to \forall x : \eta.\ px)$$

The first axiom says that zero is different from the successor of any element and the second axiom expresses the injectivity of the successor function. The third axiom is also known as the induction axiom. We need to translate these axioms to tableau rules. The first axiom induces two closing rules:

$$\frac{St = 0}{} \qquad \frac{0 = St}{}$$

The second axiom is also straightforward:

$$\frac{St = Su}{t = u}$$

The next tableau rule is to represent the induction axiom:

$$\frac{[tu]}{[t0] \mid \neg[ty],\ [t(Sy)]}\ t : \eta o,\ y \text{ fresh}$$

It can be explained as follows: If we know that a property t holds for some u, then either t holds for 0 or it does not hold for some y but for the successor of y. As we will later see, we also need a form of decomposition rules for 0, S and variables $x : \eta$:

$$\frac{x \neq x}{} \qquad \frac{0 \neq 0}{} \qquad \frac{St \neq Su}{t \neq u}$$

We will now show that these rules are indeed enough to prove the validity of the induction axiom. For this proof, we assume the full signature, including the new logical constants. We first replace any logical constant that is not supported:

$$\neg\exists p.\ \neg(\neg(p0) \lor ((\exists x.\ \neg(\neg(px) \lor p(Sx))) \lor \neg(\exists x.\ \neg(px))))$$

We assume β-reduction as normalization operator and refute the negation:

$$\begin{array}{c|c}
\multicolumn{2}{c}{\neg\neg\exists p.\ \neg(\neg(p0) \lor ((\exists x.\ \neg(\neg(px) \lor p(Sx))) \lor \neg(\exists x.\ \neg(px))))} \\
\multicolumn{2}{c}{\exists p.\ \neg(\neg(p0) \lor ((\exists x.\ \neg(\neg(px) \lor p(Sx))) \lor \neg(\exists x.\ \neg(px))))} \\
\multicolumn{2}{c}{\neg(\neg(p0) \lor ((\exists x.\ \neg(\neg(px) \lor p(Sx))) \lor \neg(\exists x.\ \neg(px))))} \\
\multicolumn{2}{c}{\neg\neg(p0)} \\
\multicolumn{2}{c}{p0} \\
\multicolumn{2}{c}{\neg((\exists x.\ \neg(\neg(px) \lor p(Sx))) \lor \neg(\exists x.\ \neg(px)))} \\
\multicolumn{2}{c}{\neg(\exists x.\ \neg(\neg(px) \lor p(Sx)))} \\
\multicolumn{2}{c}{\neg\neg(\exists x.\ \neg(px))} \\
\multicolumn{2}{c}{\exists x.\ \neg(px)} \\
\multicolumn{2}{c}{\neg(px)} \\
\hline
\begin{array}{c} \neg(p0) \\ 0 \neq 0 \end{array} &
\begin{array}{c|c}
\multicolumn{2}{c}{\neg\neg(py)} \\
\multicolumn{2}{c}{\neg(p(Sy))} \\
\multicolumn{2}{c}{py} \\
\multicolumn{2}{c}{\neg\neg(\neg(py) \lor p(Sy))} \\
\multicolumn{2}{c}{\neg(py) \lor p(Sy)} \\
\hline
\begin{array}{c} \neg(py) \\ y \neq y \end{array} & \begin{array}{c} p(Sy) \\ Sy \neq Sy \\ y \neq y \end{array}
\end{array}
\end{array}$$

Note that we used $[(\lambda x.\ \neg px)x]$ to apply the tableau rule corresponding to the third axiom.

The next example shows that there is still at least one rule missing. We want to prove the validity of $\forall x : \eta.\ (Sx) \neq x$. We rewrite the formula and refute its negation:

$$\begin{array}{c|c}
\multicolumn{2}{c}{\neg\neg\exists x.\ (Sx) = x} \\
\multicolumn{2}{c}{\exists x.\ (Sx) = x} \\
\multicolumn{2}{c}{(Sx) = x} \\
\hline
 & (Sy) \neq y \\
(S0) = 0 & (S(Sy)) = (Sy) \\
 & (Sy) = y
\end{array}$$

While the left branch is already closed at this point, we are stuck at the right branch. Although it obviously contains a contradiction, we cannot apply any tableau rule in a useful way to close this branch. The solution is to introduce confrontation at type η:

$$\frac{s =_\eta t,\ u \neq_\eta v}{s \neq u,\ t \neq u \mid s \neq v,\ t \neq v}$$

Using this rule, we can finally finish the example from above. We present again the full refutation:

$$\begin{array}{c|c}
\multicolumn{2}{c}{\neg\neg\exists x.\ (Sx) = x} \\
\multicolumn{2}{c}{\exists x.\ (Sx) = x} \\
\multicolumn{2}{c}{(Sx) = x} \\
\hline
 & (Sy) \neq y \\
 & (S(Sy)) = (Sy) \\
(S0) = 0 & (Sy) = y \\
 & \begin{array}{c|c} \hline (Sy) \neq (Sy) & \\ y \neq (Sy) & y \neq y \\ y \neq y & \end{array}
\end{array}$$

To prove completeness using the logical relations method, we would need to define $\mathcal{D}_\eta$ and $\rhd_\eta$. As already said, due to Gödel's first incompleteness theorem, we cannot expect to obtain completeness if we assume $\mathcal{D}_\eta$ is the set of natural numbers. We leave determining the right notion of model and corresponding completeness proof as future work. Instead, we turn to another logical constant.

7.3.2 Primitive Recursion

Primitive recursion or primitive recursive functions are very important in theoretical computer science as well as in programming languages. Given a logical constant *pr* : $\sigma(\eta\sigma\sigma)\eta\sigma$ at each type σ, we describe its properties by the following two axioms:

$$\begin{array}{rcl}
\mathit{pr}\, a\, g\, 0 & = & a \\
\mathit{pr}\, a\, g\, (Sx) & = & g\, x\, (\mathit{pr}\, a\, g\, x)
\end{array}$$

Similar to ε or *if*, *pr* does not necessarily return something of type o. Hence, our idea was to use again accessibility contexts. The corresponding rule looks as follows:

$$\frac{C[\mathit{pr}\,s\,t\,u]}{u = 0,\ [C[s]] \ \mid\ u = S(x),\ [C[t\,x\,(\mathit{pr}\,s\,t\,x)]]} \ \text{C accessibility context, x fresh}$$

The connection to the axioms from above is obvious: Either u is 0 and we simply return s (embedded into the corresponding accessibility context) or u is the successor of some value and we go into the recursion case. While this seems to be working at first sight, we are facing a problem here which also affects all other rules that make use of accessibility contexts: *pr* could also return something of type η and so far, we have not defined accessibility contexts for that type. Consider the following example:

$$\mathit{if} \top 0 S(0) \neq_\eta 0$$

This formula is clearly unsatisfiable but we cannot make any progress. This suggests to extend accessibility contexts to disequations at type η. However, the following example shows that this is not enough:

$$\mathit{if} \top 0 S(0) =_\eta S(0)$$

It seems like we also need to define accessibility contexts for positive equations at type η. This could also be a hint to extend the definition for quasiformulas using equations as well as disequations at type η. Again, we leave this question open as future work.

8 Conclusion

In this thesis, we have presented a modular cut-free tableau calculus with support for if-then-else, description and choice. Up to this point, there was no known tableau system with support for these three logical constants.

Using the logical relation frames technique, we have proven that if-then-else is independent of classical simply typed higher-order logic. In a similar way, we have shown that description is independent of classical simply typed higher-order logic including if-then-else. Together with the well-known fact that choice is independent of description, we have motivated why it is interesting to consider these three logical constants.

The modularity of our tableau system is achieved by introducing signatures. They represent sets of logical constants. Notions like terms, frames or interpretations all depend on a signature. Especially, the tableau system itself depends on a signature. Using the notion of quasiformulas we ensure that we always have a basic tableau system, regardless of the signature. We designed the remaining tableau rules in such a way that each logical constant can separately be added to the signature without being forced to add other logical constants.

We have given a general completeness result. This means in particular that we can present a cut-free and complete tableau calculus for arbitrary signatures. We have also shown how signatures can be restricted to get completeness with respect to standard models.

In Section 1.3, we claimed that we will extend two tableau systems presented by Brown and Smolka while maintaining their properties. Using signatures, this can be easily achieved: In [9], the tableau calculus is cut-free and complete with respect to standard models. The supported logical constants are $\bot$, $\neg$, $\wedge$, $=_\iota$ and $\forall_\iota$. Since not all of them are available in the context of this thesis, we replace $\forall_\iota$ by $\lambda x.\ \neg(\exists_\iota(\lambda y.\ \neg(xy))$ and $\wedge$ by $\lambda x, y.\ \neg(\neg x \vee \neg y)$. The resulting signature is $\{\bot, \neg, \vee, =_\iota, \exists_\iota\}$. We extend this signature to $\{\bot, \top, \neg, \vee, =_\iota, \exists_\iota, \mathit{if}_\sigma, \iota_\iota, \varepsilon_\iota\}$. By Theorem 5.27, we have a cut-free tableau system for this signature which is complete with respect to standard models.

In [8], the tableau system is cut-free and complete (with respect to general models). $=_\sigma$, $\rightarrow$ and $\bot$ are the supported logical constants. We replace $\rightarrow$ by $\vee$ and $\neg$ the obvious way and extend the resulting signature to $\{=_\sigma, \bot, \vee, \neg, \mathit{if}_\sigma, \iota_\sigma, \varepsilon_\sigma\}$. Again, Theorem 5.27 tells us that we have a complete and cut-free tableau calculus.

8.1 Future Work

We have already presented many ideas for future work in Chapter 7. Besides these ideas, we also wanted to develop a tableau system which has only very few logical constants but is still very powerful. Having as few logical constants as possible is especially interesting for automated theorem proving because it restricts the set of possible instantiations. Our first idea was to introduce a logical constant $\mathit{ifeq}_{\sigma,\tau} : \sigma\sigma\tau\tau\tau$ which combines if-then-else and equality. The corresponding function is defined as follows:

$$\mathit{ifeq}\, a\, b\, c\, d := \text{if } a = b \text{ then } c \text{ else } d$$

This logical constant is very powerful. Together with $\top$ and $\bot$, it can be used to express for example quantifiers but of course also if-then-else and equality. So far, we have no tableau rules which might yield a reasonable tableau system.

Another idea for future work is to find an abstraction for the interpretations we gave for ε and ι. It looks like these interpretations have the pattern:

$$\mathcal{I}(c)f \quad = \quad \begin{cases} \text{some } b & \text{such that \textit{some nice property} holds for } b \text{ and } f^c \text{ is empty} \\ \text{some } a & \text{such that } f^c \rhd a \text{ otherwise.} \end{cases}$$

As a consequence, almost all of the proofs based on these interpretations are very similar. Hence, a suitable abstraction could be a great simplification. Moreover, it might yield a procedure which allows to easily add other logical constant whose interpretation is not unique.

The last idea we want to present here is to extend the first tableau system mentioned in Section 1.3. This system does not support lambda abstracts but it is cut-free, terminating and complete with respect to standard models. There is strong evidence that we can at least add if_σ at all types σ, ι_ι and ε_ι to the corresponding signature while preserving all three properties and especially termination.

Bibliography

[1] Thorsten Altenkirch and Tarmo Uustalu. Normalization by Evaluation for $\lambda^{\to 2}$. In *FLOPS*, pages 260–275, 2004.

[2] Peter B. Andrews. General models, descriptions, and choice in type theory. *J. Symb. Log.*, 37(2):385–394, 1972.

[3] Julian Backes and Chad E. Brown. Analytic tableaux for higher-order logic with choice. In *Fifth International Joint Conference on Automated Reasoning (IJCAR 2010)*, 2010. to appear.

[4] Michael Beeson. Unification in lambda-calculi with if-then-else. In *CADE-15: Proceedings of the 15th International Conference on Automated Deduction*, pages 103–118, London, UK, 1998. Springer-Verlag.

[5] Evert W. Beth. Semantic entailment and formal derivability. *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, N.R.*, 18(13):309–342, 1955.

[6] Chad E. Brown. *Automated Reasoning in Higher-order Logic: Set Comprehension and Extensionality in Church's Type Theory*, volume 10 of *Studies in Logic: Logic and Cognitive Systems*. College Publications, 2007.

[7] Chad E. Brown and Gert Smolka. Analytic tableaux for simple type theory and its first-order fragment. Technical report, Saarbrücken, Germany, Dec 2009. submitted.

[8] Chad E. Brown and Gert Smolka. Complete Cut-Free Tableaux for Equational Simple Type Theory. Technical report, Saarbrücken, Germany, April 2009.

[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. Introduction to Computational Logic, Lecture Notes 2009, 2009.

[11] Chad E. Brown and Gert Smolka. Terminating Tableaux for the Basic Fragment of Simple Type Theory. In M. Giese and A. Waaler, editor, *TABLEAUX 2009*, volume 5607 of *LNCS (LNAI)*, pages 138–151. Springer LNCS 5607, Jul 2009.

[12] Alonzo Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[13] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[14] Adolf Fraenkel. Über den Begriff "definit" und die Unabhängigkeit des Auswahlsaxioms. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physik-math. Klasse*, pages 253–257, 1922.

[15] Adolf Fraenkel. Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen*, 86:230–237, 1922.

[16] Gottlob Frege. *Grundgesetze der Arithmetik (Band I)*. 1893.

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

[18] Michael Kohlhase. Higher-order tableaux. In *TABLEAUX*, pages 294–309, 1995.

[19] Adolf Lindenbaum and Andrzej Mostowski. Über die Unabhängigkeit des Auswahlsaxioms und einiger seiner Folgerungen. *Comptes Rendus des Séances de la Société des Sciences et des Lettres de Varsovie*, 31:27–32, 1938.

[20] Grigori Mints. Cut-elimination for simple type theory with an axiom of choice. *J. Symb. Log.*, 64(2):479–485, 1999.

[21] Lawrence C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

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

[23] Bertrand Russell. On Denoting. *Mind*, 14:479–493, 1905.

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

[25] Gaisi Takeuti. *Proof Theory*. North-Holland, Amsterdam, 1975.

[26] Ernst Zermelo. Beweis, daß jede Menge wohlgeordnet werden kann. *Mathematische Annalen*, 59:514–516, 1904.

[27] Ernst Zermelo. Neuer Beweis für die Möglichkeit einer Wohlordnung. *Mathematische Annalen*, 65:107–128, 1908.

[28] Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre I. *Mathematische Annalen*, 65:261–281, 1908.

A Description Implies If-Then-Else

As already mentioned, it is possible to prove the existence of the if-then-else function using a description operator. The tableau refutation will be very long. We split it up into several parts and also omit some branches which are not so interesting. We start by proving a trivial fact: Given variables x and y of type ι, we can always refute the set $\{x = y, x \neq y\}$:

$$\begin{array}{c|c} \multicolumn{2}{c}{x = y} \\ \multicolumn{2}{c}{x \neq y} \\ \hline x \neq x & x \neq y \\ y \neq x & y \neq y \end{array}$$

Branches which can be closed this way will be marked by a *. We start the tableau refutation. The most important part is the instatiation for f in the first step:

$$\lambda x, y, z.\ \iota(\lambda u.\ \neg(\neg x \vee y \neq u) \vee \neg(x \vee z \neq u))$$

Note that we split several terms into two lines and mark such situations by a $\rightsquigarrow$:

$$\begin{array}{c} \neg(\exists f.\ \neg(\exists x, y.\ (f\top xy \neq x) \vee (f\bot xy \neq y))) \\ \neg\neg(\exists x, y.\ (\iota(\lambda u.\ \neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \neq x) \rightsquigarrow \\ \vee(\iota(\lambda u.\ \neg(\neg\bot \vee x \neq u) \vee \neg(\bot \vee y \neq u)) \neq y)) \\ \exists x, y.\ (\iota(\lambda u.\ \neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \neq x) \rightsquigarrow \\ \vee(\iota(\lambda u.\ \neg(\neg\bot \vee x \neq u) \vee \neg(\bot \vee y \neq u)) \neq y) \\ \exists y.\ (\iota(\lambda u.\ \neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \neq x) \rightsquigarrow \\ \vee(\iota(\lambda u.\ \neg(\neg\bot \vee x \neq u) \vee \neg(\bot \vee y \neq u)) \neq y) \\ (\iota(\lambda u.\ \neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \neq x) \rightsquigarrow \\ \vee(\iota(\lambda u.\ \neg(\neg\bot \vee x \neq u) \vee \neg(\bot \vee y \neq u)) \neq y) \end{array}$$

This branch now splits into two branches. We only refute the left branch. The right branch can be analogously refuted.

$$\begin{array}{c|c|c} \multicolumn{3}{c}{\iota(\lambda u.\ \neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \neq x} \\ \hline (1) & (2) & (3) \end{array}$$

where (1) is

$$\begin{array}{c}
\neg(\neg(\neg\top \lor x \neq x) \lor \neg(\top \lor y \neq x)))\\
\neg\neg(\neg\top \lor x \neq x))\\
\neg\neg(\top \lor y \neq x))\\
\neg\top \lor x \neq x\\ \hline
\begin{array}{c|c} \neg\top & x \neq x \end{array}
\end{array}$$

(2) is

$$\begin{array}{c}
x' \neq y'\\
\neg(\neg\top \lor x \neq x') \lor \neg(\top \lor y \neq x')\\
\neg(\neg\top \lor x \neq y') \lor \neg(\top \lor y \neq y')\\ \hline
\begin{array}{c|c}
\begin{array}{c}
\neg(\neg\top \lor x \neq x')\\
\neg\neg\top\\
\neg(x \neq x')\\
x = x'\\ \hline
\begin{array}{c|c}
\begin{array}{c}
\neg(\neg\top \lor x \neq y')\\
\neg\neg\top\\
\neg(x \neq y')\\
x = y'\\ \hline
\begin{array}{c|c}
\begin{array}{c} x \neq x'\\ x' \neq x' \end{array} & \begin{array}{c} x \neq y'\\ x' \neq y'\\ * \end{array}
\end{array}
\end{array}
&
\begin{array}{c} \neg(\top \lor y \neq y')\\ \neg\top \end{array}
\end{array}
\end{array}
&
\begin{array}{c} \neg(\top \lor y \neq x')\\ \neg\top \end{array}
\end{array}
\end{array}$$

(3) is

$$\begin{array}{c}
\neg(\neg\top \lor x \neq \iota(\lambda u.\neg(\neg\top \lor x \neq u) \lor \neg(\top \lor y \neq u))) \rightsquigarrow\\
\lor\neg(\top \lor y \neq \iota(\lambda u.\neg(\neg\top \lor x \neq u) \lor \neg(\top \lor y \neq u)))\\ \hline
\begin{array}{c|c} (4) & (5) \end{array}
\end{array}$$

(4) is

$$\begin{array}{c}
\neg(\neg\top \lor x \neq \iota(\lambda u.\neg(\neg\top \lor x \neq u) \lor \neg(\top \lor y \neq u)))\\
\neg\neg\top\\
\neg(x \neq \iota(\lambda u.\neg(\neg\top \lor x \neq u) \lor \neg(\top \lor y \neq u)))\\
x = \iota(\lambda u.\neg(\neg\top \lor x \neq u) \lor \neg(\top \lor y \neq u))\\ \hline
\begin{array}{c|c} (6) & x \neq x \end{array}
\end{array}$$

(5) is

$$\begin{array}{c}
\neg(\top \lor y \neq \iota(\lambda u.\neg(\neg(\top \lor x \neq u) \lor \neg(\top \lor y \neq u)))\\
\neg\top\\
\neg(y \neq \iota(\lambda u.\neg(\neg(\top \lor x \neq u) \lor \neg(\top \lor y \neq u)))
\end{array}$$

and (6) is

$$\begin{array}{c}
x \neq \iota(\lambda u.\neg(\neg(\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \\
\iota(\lambda u.\neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \neq \iota(\lambda u.\neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \\
(\lambda u.\neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u)) \neq \lambda u.\neg(\neg\top \vee x \neq u) \vee \neg(\top \vee y \neq u) \\
(\neg(\neg\top \vee x \neq z) \vee \neg(\top \vee y \neq z)) \neq (\neg(\neg\top \vee x \neq z) \vee \neg(\top \vee y \neq z)) \\
\hline
\begin{array}{c|c}
\begin{array}{c}
\neg(\neg\top \vee x \neq z) \vee \neg(\top \vee y \neq z) \\
\neg(\neg(\neg\top \vee x \neq z) \vee \neg(\top \vee y \neq z)) \\
\neg\neg(\neg\top \vee x \neq z) \\
\neg\neg(\top \vee y \neq z) \\
\neg\top \vee x \neq z \\
\hline
\begin{array}{c|c}
\begin{array}{c}
\neg(\neg\top \vee x \neq z) \\
\neg\neg\top \\
\neg(x \neq z) \\
x = z \\
\hline
\begin{array}{c|c}
\neg\top & \begin{array}{c} x \neq z \\ * \end{array}
\end{array}
\end{array}
&
\begin{array}{c}
\neg(\top \vee y \neq z) \\
\neg\top
\end{array}
\end{array}
\end{array}
&
\text{identical to the left branch}
\end{array}
\end{array}$$