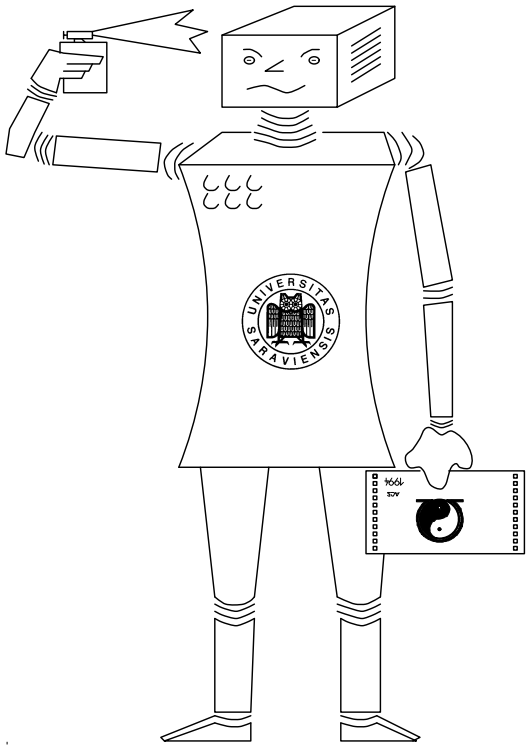
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

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



A Tableau Calculus for Partial Functions

Manfred Kerber and Michael Kohlhase

Published as: In Annals of the Kurt-Gödel-Society, forthcoming

A Tableau Calculus for Partial Functions*

Manfred Kerber Michael Kohlhase

Fachbereich Informatik, Universität des Saarlandes
66041 Saarbrücken, Germany
+49-681-302-{4628|4627}
{kerber|kohlhase}@cs.uni-sb.de

Abstract. Even though it is not very often admitted, partial functions do play a significant role in many practical applications of deduction systems. Kleene has already given a semantic account of partial functions using a three-valued logic decades ago, but there has not been a satisfactory mechanization. Recent years have seen a thorough investigation of the framework of many-valued truth-functional logics. However, strong Kleene logic, where quantification is restricted and therefore not truth-functional, does not fit the framework directly. We solve this problem by applying recent methods from sorted logics. This paper presents a tableau calculus that combines the proper treatment of partial functions with the efficiency of sorted calculi.

Keywords: Partial functions, many-valued logic, sorted logic, tableau.

1 Introduction

Many practical applications of deduction systems in mathematics and computer science rely on the correct and efficient treatment of partial functions. For this purpose different approaches—reaching from workarounds for concrete situations to a proper general treatment—have been developed. In the following we will introduce the main approaches and exemplify their advantages and disadvantages by some trivial examples from arithmetic. For a more detailed discussion of the different approaches compare [Far90].

There are essentially four approaches of treating partiality. First, these expressions can syntactically be excluded. Second, it is possible to disregard or bypass partiality. Third, partiality is taken serious and this is reflected in the semantics and the calculus. Fourth, there is some mixture between options two and three.

In the first approach terms like $\frac{x}{0}$ are treated as syntactically ill-formed, for instance, by using a sorted logic, in which the domain of the $\frac{x}{y}$ function is defined to be $\mathbb{R} \times \mathbb{R}^*$ (where $\mathbb{R}^*$ denotes the real numbers without 0). Thereby the whole problem of partiality has been bypassed. In the cases, where such a procedure is possible, this approach is quite adequate and reflects the usual way

* This work was supported by the Deutsche Forschungsgemeinschaft (SFB 314, D2)

of handling undefined expressions in mathematics: to assure that all expressions are defined before beginning to reason about them. It is, however, not always possible to exclude such expressions from the consideration a priori. For instance, if you consider terms like $\frac{1}{f(x)}$, it would be necessary to exclude this expression for those x where $f(x) = 0$; depending on the definition of f, this might be not computable at all. In consequence, this approach remedies the problem of partiality in certain cases only and does not provide a full solution.

In the second approach a value is assigned to $\frac{1}{0}$, either a fixed value (e.g. 0) or an undetermined one. In both cases it is necessary to tolerate undesired theorems, in the first case, for instance, $\frac{1}{0} = 0$, or in the second case from $0 \cdot x = 0$ the instance $0 \cdot \frac{1}{0} = 0$. This approach is not satisfying, if such theorems are unwanted, which is normally the case in mathematics.

In the third approach, terms like $\frac{1}{0}$ are not defined and semantically either uninterpreted or interpreted by some error element. In the same manner, atomic formulae, containing such an undefined term, like $\frac{1}{0} = 0$ are not interpreted by a truth value (true or false) at all or are interpreted by a third truth value (undefined). As in the first approach, partiality is taken serious, but it is no longer necessary to single out the undefined expressions a priori. The main drawback of this approach is that classical two-valued logic is not adequate for its mechanization. A possible formalization can be done by a three-valued logic, however. Kleene makes this approach formal, by introducing an individual $\perp$ denoting meaningless individuals and a third truth value u, standing for the "undefined" truth value. However, in contrast to the general framework for many-valued truth-functional logics, Kleene's quantifiers only range over defined values, that is, not over $\perp$, making a direct utilization of the methods developed by Carnielli [Car87, Car91], Hähnle [Häh92], Baaz and Fermüller [BF92] impossible. Kleene's approach has been used by Tichy [Tic82], Lucio-Carrasco and Gavilanes-Franco [LCGF89] to give logical systems for partial functions. Both approaches offer unsorted operationalizations of the systems in sequent calculi.

The fourth approach is less radical insofar as terms are treated as in the third approach, but the problems that accompany treating a third truth value are avoided (cf. [Bee85, Far90, Sch68, Wei89]): All atomic expressions containing a meaningless term are considered as false. This has the advantage that partial functions can be handled within the classical two-valued framework. However, the serious drawback is that the results of these logic systems can be unintuitive to the working mathematician. For instance in elementary arithmetic the following sentence

$$\forall x, y, z.\ z = \frac{x}{y} \Rightarrow x = y * z$$

is a theorem of such systems since the scope is true for the case $y \neq 0$ and for the case $y = 0$, the formula $z = \frac{x}{0}$ obtains the truth value f which in turn makes the implication true, too. However, it is mathematical consensus that the equation should only hold provided that y is not 0. It will turn out (cf. example 2.11) that the formula is not a theorem in our formalization, since the case $y = 0$ is a counterexample.

This paper formalizes Kleene's ideas for partial functions (the third approach) in a sorted three-valued logic, called $\mathcal{SKL}$, that uses Kleene's strong interpretation of connectives and quantifiers and adapts techniques from Weidenbach's sorted logic [Wei89] to handle definedness information. We furthermore present a tableau calculus $\mathcal{TPF}$ for partial functions that carries over the methods developed in the context of resolution theorem proving for partial functions [KK94] to the tableau framework. Standard first-order tableaux were introduced by Beth [Bet55] and Hintikka [Hin55] and later unified by Smullyan [Smu68]. The free variable tableau method has its origin in the work of Prawitz [Pra60] and has further been elaborated by Reeves [Ree87] and Fitting [Fit90]. Both calculi reported here are strongly influenced by Weidenbach's tableau calculus with sorts [Wei94], which introduces reasoning with dynamic sorts to tableau calculi.

We would like to thank Christian Fermüller, Reiner Hähnle, and Christoph Weidenbach for comments and clarifying discussions.

2 Strong Sorted Kleene Logic ($\mathcal{SKL}$)

In [Kle52] Kleene presents a logic, which he calls *strong three-valued logic* for reasoning about partial recursive predicates on the set of natural numbers. He argues that the intuitive meaning of the third truth value should be "undefined" or "unknown" and introduces the truth tables shown in definition 2.6. Similarly Kleene enlarges the universe of discourse by an element $\perp$ denoting the undefined number. In his exposition the quantifiers only range over natural numbers, in particular he does not quantify over the undefined individual (number).

The approach of this paper is to make Kleene's meta-level discussion of defined and undefined individuals explicit by structuring the universe of discourse with the sort $\mathfrak{D}$ for all defined individuals. Furthermore all functions and predicates are strict, that is, if one of the arguments of a compound term or an atom evaluates to $\perp$, then the term evaluates to $\perp$ or the truth value of the atom is u. Just as in Kleene's system, our quantifiers only range over individuals in $\mathfrak{D}$, that is, individuals that are not undefined. This is in contrast to the well-understood framework for truth-functional many-valued logics, where the concept of definedness and defined quantification cannot be easily introduced, since quantification is truth-functional and depends on the truth values for all (even the undefined) instantiations of the scope. Kleene's concept of bounded quantification is essential for our program of representing partial functions, since in a truth-functional approach no proper universally quantified expression can evaluate to the truth value t (dually for the existential quantifier), since all functions and predicates are assumed strict.

In the following we present the logic system $\mathcal{SKL}$, which is a sorted version of what we believe to be a faithful formalization of Kleene's ideas from [Kle52]. We treat the sorted version here, since we need the machinery for dynamic sorts in the calculus to be able to treat the sort $\mathfrak{D}$ (sort techniques as that from [Wei89, Wei91] give us the bounded quantification). We will call formulations of $\mathcal{SKL}$ where $\mathfrak{D}$ is the only sort in the signature *strong unsorted Kleene logic*, since the

sort $\mathfrak{D}$ is indispensable. The further use of sorts gives the well-known advantages of sorted logics for the conciseness of the representation and the reduction of search spaces.

2.1 Syntax and Semantics

Definition 2.1 (Signature) A *signature* $\Sigma := (\mathcal{S}, \mathcal{V}, \mathcal{F}, \mathcal{P})$ consists of the following disjoint sets

- $\mathcal{S}$ is a finite set of *sorts* including the sort $\mathfrak{D}$. We define $\mathcal{S}^* := \mathcal{S} \setminus \{\mathfrak{D}\}$
- $\mathcal{V}$ is a set of *variable symbols*. Each variable x is associated with a unique sort S, which we write in the index, i.e. x_S. We assume that for each sort $S \in \mathcal{S}$ there is a countably infinite supply of variables of sort S in $\mathcal{V}$.
- $\mathcal{F}$ is a set of *function symbols*.
- $\mathcal{P}$ is the set of *predicate symbols*.

The sets $\mathcal{F}$ and $\mathcal{P}$ are subdivided into the sets $\mathcal{F}^k$ of *function symbols of arity k* and $\mathcal{P}^k$ of *predicate symbols of arity k*. Note that individual constants are just nullary functions. We call a signature *unsorted* if $\mathcal{S}^*$ is empty, that is, if $\mathfrak{D}$ is the only sort.

Definition 2.2 (Terms and Formulae) We define the set of *terms* to be the set of variables together with *compound terms* $f(t^1, \ldots, t^k)$ for terms $t^1, \ldots, t^k$ and $f \in \mathcal{F}^k$.

If $P \in \mathcal{P}^k$, then $P(t^1, \ldots, t^k)$ is a *proper atom*. If t is a term and S a sort then $t{:\!\leqslant}S$ is a *sort atom*. The set of *formulae* contains all atoms and with formulae A and B the formulae $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $\neg A$, $!A$, $\forall x_S . A$, and $\exists x_S . A$. Here the intended meaning of $!A$ is that A is defined.

We will now define the three valued semantics for $\mathcal{SKL}$ by postulating an "undefined individual" $\perp$ in the universe of discourse. Note that this is similar to the classical flat CPO construction [Sco70], but Kleene's interpretation of truth values does not make u minimal. Since we are not interested in least fixpoints, monotonicity does not play a role in this paper.

Definition 2.3 (Strict Σ-Algebra) Let Σ be a signature, then a pair $(\mathcal{A}, \mathcal{I})$ is called a *strict Σ-algebra*, iff

1. the *carrier set* $\mathcal{A}$ is an arbitrary set that contains $\perp$,
2. the *interpretation function* $\mathcal{I}$ obeys the following restrictions:
   (a) For all function symbols f, the function $\mathcal{I}(f)\colon \mathcal{A}^k \longrightarrow \mathcal{A}$ is strict for $\perp$, that is, $\mathcal{I}(f)(a_1, \ldots, a_k) = \perp$, if $a_i = \perp$ for (at least) one i.
   (b) If P is a predicate symbol, then the relation $\mathcal{I}(P) \subseteq \mathcal{A}^k$ is strict for $\perp$, that is, $\mathcal{I}(P)(a_1, \ldots, a_k) = \mathsf{u}$, if $a_i = \perp$ for (at least) one i.
   (c) If $S \neq \mathfrak{D}$ is a sort, then $\mathcal{I}(S)$ is a total and strict unary relation, that is, $\mathcal{I}(S)(a) \in \{\mathsf{f}, \mathsf{t}\}$, if $a \neq \perp$ and $\mathcal{I}(S)(\perp) = \mathsf{u}$.
   (d) $\mathcal{I}(\mathfrak{D})(\perp) = \mathsf{f}$ and $\mathcal{I}(\mathfrak{D})(a) = \mathsf{t}$, if $a \neq \perp$. Note that in contrast to all other sorts and predicates, the denotation of $\mathfrak{D}$ is not a strict relation.

We define the *carrier* $\mathcal{A}_S$ *of sort* S as $\mathcal{A}_S := \{a \in \mathcal{A} \mid \mathcal{I}(S)(a) = \mathsf{t}\}$. Note that in contrast to other sorted logics, it is not assumed that the $\mathcal{A}_S$ are non-empty, in fact we do not even assume the existence of defined elements in the carrier. Furthermore $\bot \notin \mathcal{A}_S$ for any $S \in \mathcal{S}$.

By systematically deleting $\bot$ and u from the carrier and the truth values we can canonically transform strict Σ-algebras into algebras of partial functions. These are an algebraic account of the standard interpretation in mathematics, where partiality of functions is directly modeled by right-unique relations. Obviously these notions of algebras have a one-to-one correspondence, so both approaches are equivalent.

Definition 2.4 (Σ-assignment) Let $(\mathcal{A},\mathcal{I})$ be a strict Σ-algebra, then we call a total mapping $\varphi: \mathcal{V} \longrightarrow \mathcal{A}$ a Σ-*assignment*, iff $\varphi(x_S) \in \mathcal{A}_S$, provided $\mathcal{A}_S$ is non-empty and $\varphi(x_S) = \bot$ if $\mathcal{A}_S = \emptyset$. We denote the Σ-assignment that coincides with φ away from x and maps x to a with $\varphi, [a/x]$.

Definition 2.5 Let φ be a Σ-assignment into a strict Σ-algebra $(\mathcal{A},\mathcal{I})$ then we define the *value function* $\mathcal{I}_\varphi$ *from formulae to* $\mathcal{A}$ inductively to be

1. $\mathcal{I}_\varphi(f) := \mathcal{I}(f)$, if f is a function or a predicate.
2. $\mathcal{I}_\varphi(x) := \varphi(x)$, if x is a variable.
3. $\mathcal{I}_\varphi(f(t^1,\ldots,t^k)) := \mathcal{I}(f)(\mathcal{I}_\varphi(t^1),\ldots,\mathcal{I}_\varphi(t^k))$, if f is a function or predicate.
4. $\mathcal{I}_\varphi(t{\leqslant}S) := \mathcal{I}(S)(\mathcal{I}_\varphi(t))$.

Since this definition applies to $\mathcal{P}$ and $\mathcal{F}$ alike, we have given the semantics of all atomic formulae. The semantic status of sorts is that of total unary predicates; in particular we have $\mathcal{I}_\varphi(t{\leqslant}S) = \mathsf{u}$, iff $\mathcal{I}_\varphi(t) = \bot$ for $S \neq \mathfrak{D}$.

Definition 2.6 The value of a formula dominated by a connective is obtained from the value(s) of the subformula(e) in a truth-functional way. Therefore it suffices to define the truth tables for the connectives:

$\wedge$	f	u	t
f	f	f	f
u	f	u	u
t	f	u	t

$\vee$	f	u	t
f	f	u	t
u	u	u	t
t	t	t	t

$\Rightarrow$	f	u	t
f	t	t	t
u	u	u	t
t	f	u	t

$\neg$	
f	t
u	u
t	f

!	
f	t
u	f
t	t

The semantics of the quantifiers is defined with the help of function $\widetilde{\forall}$ and $\widetilde{\exists}$ from the non-empty subsets of the truth values in the truth values. We define

$$\mathcal{I}_\varphi(\mathsf{Q}x_S.\, A) := \widetilde{\mathsf{Q}}(\{\mathcal{I}_{\varphi,[a/x]}(A) \mid a \in \mathcal{A}_S\})$$

where $\mathsf{Q} \in \{\forall,\exists\}$ and furthermore

$$\widetilde{\forall}(T) := \begin{cases} \mathsf{t} & \text{for } T = \{\mathsf{t}\} \text{ or } T = \emptyset \\ \mathsf{u} & \text{for } T = \{\mathsf{t},\mathsf{u}\} \text{ or } \{\mathsf{u}\} \\ \mathsf{f} & \text{for } \mathsf{f} \in T \end{cases} \qquad \widetilde{\exists}(T) := \begin{cases} \mathsf{t} & \text{for } \mathsf{t} \in T \\ \mathsf{u} & \text{for } T = \{\mathsf{f},\mathsf{u}\} \text{ or } \{\mathsf{u}\} \\ \mathsf{f} & \text{for } T = \{\mathsf{f}\} \text{ or } T = \emptyset \end{cases}$$

Note that with this definition quantification is separated into a truth-functional part $\widetilde{\forall}$ and an instantiation part that only considers members of $\mathcal{A}_S$. Since $\perp$ is not a member of any $\mathcal{A}_S$, quantification never considers it and therefore cannot be truth-functional even for the unsorted case.

For lack of space we will in the following often only treat the (sufficient) subset $\{\wedge, \neg, !, \forall\}$ of logical symbols, since all others can be defined from these just as in the classical two-valued logic.

Kleene does not use the ! operator as a connective but treats it on the meta-level. While it is useful it is not necessary for the treatment. Furthermore, even this connective does not render $\mathcal{SKL}$ truth-functionally complete, since, just like the other connectives and the quantifiers, ! is *normal*, that is, when restricted to $\{\mathsf{f}, \mathsf{t}\}$ yields values in $\{\mathsf{f}, \mathsf{t}\}$.

Definition 2.7 (Σ-Model) Let A be a formula, then we call a strict Σ-algebra $\mathcal{M} := (\mathcal{A}, \mathcal{I})$ a Σ-*model for* A (written $\mathcal{M} \models A$), iff $\mathcal{I}_\varphi(A) = \mathsf{t}$ for all Σ-assignments φ. With this notion we can define the notions of *validity*, *(un)-satisfiability*, and *entailment* (i.e. $\Phi \models A$) in the usual way.

Remark 2.8 The "tertium non datur" principle of classical logic is no longer valid, since formulae can be undefined, in which case they are neither true nor false. We do, however, have a "quartum non datur" principle, that is, formulae are either true, false, or undefined, which allows us to derive the validity of a formula by refuting that it is false or undefined. We will use this observation in our tableau calculus.

The classical deduction theorem does not hold for $\mathcal{SKL}$ since the semantic status of a formula in the hypotheses is different from its status in the antecedent of an implication. A formula in the hypotheses is assumed to evaluate semantically to t, hence in particular it is defined. This leads to the following modified deduction theorem.

Theorem 2.9 (Deduction Theorem) $\Phi \cup \{A\} \models B$ *iff* $\Phi \models A \wedge !A \Rightarrow B$.

Proof: Let us first assume the first property and let $\mathcal{M}$ be a model of $\Phi \cup \{A\}$ then $\mathcal{M}$ is also a model of B, hence $\mathcal{M} \models A \wedge !A \Rightarrow B$. That means in order to show the second property we only have to look at interpretations which are models of Φ but not of A. For these, however, $A \wedge !A$ evaluates to f, hence they are models of $A \wedge !A \Rightarrow B$ too.

If the second property is given and $\mathcal{M}$ is a model of Φ then $\mathcal{M}$ is also a model of $A \wedge !A \Rightarrow B$. In order to prove the first property, only the subclass of those models has to be considered which are also models of A. These are, however, also models of $!A$, hence models of B too. □

Remark 2.10 While in classical logic, the consequence relation is directly connected to the implication, here things are a little bit more difficult. In particular, when proving mathematical theorems, it is quite usual to do this with respect to some background theory (axioms and definitions), which can no longer simply

be taken in the antecedent of an implication. Hence we will often consider for mathematical applications so-called *consequents*, that is, pairs consisting of a set of formulae Φ and a formula A. We call a consequent $\Phi \models A$ valid, if A is entailed by Φ in all Σ-models.

Example 2.11 Now we can come back to the example from the exposition. The assertion is not a theorem of $\mathcal{SKL}$, since the instance $1 = \frac{1}{0} \Rightarrow 1 = 0 \cdot 1$ is not a valid formula (in any reasonable axiomatization of elementary arithmetic). While the antecedent of the implication evaluates to u, the succedent evaluates to f, hence the whole expression to u. Thus, this theorem cannot be derived in our sound tableau calculus to be presented in section 3.

Example 2.12 (Extended Example) We will formalize an extended example from elementary algebra that shows the basic features of $\mathcal{SKL}$. Here the sort $\mathbb{R}^*$ denotes the real numbers without zero. Note that we use the sort information to encode definedness information for inversion: $\frac{1}{x}$ is defined for all $x \in \mathbb{R}^*$, since $\mathbb{R}^*$ is subsort of $\mathfrak{D}$ by definition. Naturally, we give only a reduced formalization of real number arithmetic that is sufficient for our example. (For instance, we could add expressions like $\frac{1}{0} \not< \mathfrak{D}$.) Consider the consequent $\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}, \mathrm{A4}, \mathrm{A5}\} \models \mathrm{T}$ with

$$
\begin{array}{ll}
\mathrm{A1} & \forall x_{|\mathbb{R}}.\ x \neq 0 \Rightarrow x < \mathbb{R}^* \\
\mathrm{A2} & \forall x_{|\mathbb{R}^*}.\ \frac{1}{x} < \mathbb{R}^* \\
\mathrm{A3} & \forall x_{|\mathbb{R}^*}.\ x^2 > 0 \\
\mathrm{A4} & \forall x_{|\mathbb{R}}.\ \forall y_{|\mathbb{R}}.\ x - y < \mathbb{R} \\
\mathrm{A5} & \forall x_{|\mathbb{R}}.\ \forall y_{|\mathbb{R}}.\ x - y = 0 \Rightarrow x = y \\
\mathrm{T} & \forall x_{|\mathbb{R}}.\ \forall y_{|\mathbb{R}}.\ x \neq y \Rightarrow \left(\frac{1}{x-y}\right)^2 > 0
\end{array}
$$

An informal mathematical argumentation why T is entailed by $\{\mathrm{A1}, \ldots, \mathrm{A5}\}$ can be as follows: In the consequent above, the Ai are assumed to be true, that is, neither false nor undefined. Let x and y be arbitrary elements of $\mathbb{R}$. If $x = y$, the premise of T is false, hence the whole expression true (in this case the conclusion evaluates to u). If $x \neq y$, then the premise is true and the truth value of the whole expression is equal to that of the conclusion $\left(\frac{1}{x-y}\right)^2 > 0$. Since $x \neq y$ we get by A5 that $x - y \neq 0$ and by A4 that $x - y < \mathbb{R}$, hence by A1 $x - y < \mathbb{R}^*$ and by A2 $\frac{1}{x-y} < \mathbb{R}^*$, which finally gives $\left(\frac{1}{x-y}\right)^2 > 0$ together with A3.

Note that this reasoning is not justified for the implication $\mathrm{A} := \mathrm{A1} \wedge \mathrm{A2} \wedge \mathrm{A3} \wedge \mathrm{A4} \wedge \mathrm{A5} \Rightarrow \mathrm{T}$, since there are hidden assumptions, for instance, the totality of the binary predicate $>$ on $\mathbb{R} \times \mathbb{R}$. In fact the formula A is not a tautology, since it is possible to interpret the $>$ predicate as undefined for the second argument being zero, so that A3 as well as T evaluate to u, while the other Ai evaluate to t, hence the whole expression evaluates to u.

2.2 Relativization into Truth-Functional Logic

In this section we show that we can always systematically transform $\mathcal{SKL}$ formulae to formulae in an unsorted truth-functional three-valued logic $\mathbf{K}^3$ in a

way that respects the semantics. However, we will see that this formulation will lose much of the conciseness of the presentation and enlarge the search spaces involved with automatic theorem proving.

At first glance it may seem that $\mathcal{SKL}$ is only a sorted variant of a three-valued instance of the truth functional many-valued logics that were very thoroughly investigated by Carnielli, Hähnle, Baaz and Fermüller [BF92, Car87, Car91, Häh92]. However, since all instances of this framework are truth-functional, that is, the denotations of the connectives and quantifiers only depend on the truth values of (certain instances of) their arguments, even unsorted Kleene logic does not fit into this paradigm, since quantification excludes the undefined element. In $\mathcal{SKL}$ we solve the problem with the quantification by postulating a sort $\mathfrak{D}$ of all defined individuals, which is a supersort of all other sorts. Therefore the relativization mapping not only considers sort information, it also has to care about definedness aspects in quantification.

Informally $\mathbf{K}^3$-formulae are just first-order formulae (with the additional unary connective !). While the three-valued semantics of the connectives is just that given in definition 2.6, the semantics of the quantifier uses unrestricted instantiation, that is,

$$\mathcal{I}_\varphi(\forall x_{\cdot}\, A) := \widetilde{\forall}(\{\mathcal{I}_{\varphi,[a/x]}(A) \mid a \in \mathcal{A}\})$$

Definition 2.13 (Relativization) We define transformations $\Re^{\mathcal{S}}$ and $\Re^{\mathfrak{D}}$, that map $\mathcal{SKL}$-sentences to unsorted $\mathcal{SKL}$-sentences and further into $\mathbf{K}^3$-sentences. $\Re^{\mathcal{S}}$ is the identity on terms and atoms, homomorphic on connectives, and

$$\Re^{\mathcal{S}}(\forall x_S{\cdot}\, \Phi) := \forall x_{\mathfrak{D}}{\cdot}\, S(x) \Rightarrow \Re^{\mathcal{S}}(\Phi)$$

Note that in order for these sentences to make sense in unsorted $\mathcal{SKL}$ we have to extend the set of predicate symbols by unary predicates S for all sort symbols $S \in \mathcal{S}^*$. Furthermore, for any of these new predicates we need the axiom: $\forall x_{\mathfrak{D}}{\cdot}\, !S(x)$. The set of all these axioms is denoted by $\Re^{\mathcal{S}}(\Sigma)$.

We define $\Re^{\mathfrak{D}}$ to be the identity (only dropping the sort references from the variables) on terms and proper atoms and

- $\Re^{\mathfrak{D}}(t{\leqslant}\mathfrak{D}) := \mathfrak{D}(t)$
- $\Re^{\mathfrak{D}}(\forall x_{\mathfrak{D}}{\cdot}\, A) := \forall x_{\cdot}\, \mathfrak{D}(x) \Rightarrow \Re^{\mathfrak{D}}(A)$

Just as above we have to extend the set of predicate symbols by a unary predicate $\mathfrak{D}$ and need a set $\Re^{\mathfrak{D}}(\Sigma)$ of signature axioms, which contains the axioms

$$\forall x_1, \ldots, x_n{\cdot}\, P^n(x_1, \ldots, x_n) \vee \neg P^n(x_1, \ldots, x_n) \Rightarrow (\mathfrak{D}(x_1) \wedge \ldots \wedge \mathfrak{D}(x_n))$$
$$\forall x_1, \ldots, x_n{\cdot}\, \mathfrak{D}(f(x_1, \ldots, x_n)) \Rightarrow (\mathfrak{D}(x_1) \wedge \ldots \wedge \mathfrak{D}(x_n))$$

for any predicate symbol $P \in \mathcal{P}^n$ and for any function symbol $f \in \mathcal{F}^n$, together with the axiom

$$\forall x_{\cdot}\, \mathfrak{D}(x) \vee \neg\mathfrak{D}(x)$$

These axioms axiomatize the $\mathcal{SKL}$ notion of definedness in $\mathbf{K}^3$. In particular the last axiom states that the predicate $\mathfrak{D}$ is two-valued, in contrast to all other

sort predicates which are strict and thus three-valued. The other axioms force all functions and predicates to be interpreted strictly with respect to the $\mathfrak{D}$ predicate.

Theorem 2.14 (Sort Theorem) *Let Φ be a set of sentences, then the following statements are equivalent*

1. *Φ has a Σ-model.*
2. *$\Re^{\mathcal{S}}(\Phi)$ has a $\Sigma \cup \mathcal{S}^*$-model that satisfies $\Re^{\mathcal{S}}(\Sigma)$.*
3. *$\Re^{\mathfrak{D}} \circ \Re^{\mathcal{S}}(\Phi)$ has a $\mathbf{K}^3$-model that satisfies $\Re^{\mathfrak{D}}(\Sigma \cup \mathcal{S}^*) \cup \Re^{\mathfrak{D}}(\Re^{\mathcal{S}}(\Sigma))$.*

Proof: We will only show the equivalence of 2. and 3. since the equivalence of 1. and 2. can be proven with the same methods. Therefore we can restrict our proof to unsorted $\mathcal{SKL}$, where $\mathcal{S}^* = \emptyset$

Let $\mathcal{M} := (\mathcal{A}, \mathcal{I})$ be a Σ-model for Φ, then we construct a $\mathbf{K}^3$-model $\mathcal{M}^3 = (\mathcal{A}^3, \mathcal{I}^3)$ for $\Re^{\mathfrak{D}}(\Phi)$. Let $\mathcal{A}^3 := \mathcal{A}$, $\mathcal{I}^3(f) := \mathcal{I}(f)$ and $\mathcal{I}^3(P) := \mathcal{I}(P)$ where f is a function symbol and P is a predicate symbols or the sort $\mathfrak{D}$. Clearly, we have $\mathcal{M}^3 \models^{\mathbf{K}^3} \Re^{\mathfrak{D}}(\Sigma)$, since $\mathcal{M}$ is a Σ-model, where all functions are strict and the carrier $\mathcal{A}$, defined as the image of $\mathcal{I}^3(\mathfrak{D})$, is nonempty.

Furthermore let φ be a Σ-assignment and $\mathcal{M} \models_\varphi \Phi$, then we show by structural induction that $\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\Phi)) = \mathcal{I}_\varphi(\Phi)$ and therefore $\mathcal{M}^3 \models^{\mathbf{K}^3}_\varphi \Re^{\mathfrak{D}}(\Phi)$. This claim is immediate for terms and proper atoms. For sort atoms we have

$$\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(t{\leqslant}\mathfrak{D})) = \mathcal{I}^3_\varphi(\mathfrak{D}(t)) = \mathcal{I}^3(\mathfrak{D})(\mathcal{I}^3_\varphi(t)) = \mathcal{I}(\mathfrak{D})(\mathcal{I}_\varphi(t)) = \mathcal{I}_\varphi(t{\leqslant}\mathfrak{D})$$

thus we have $\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(A)) = \mathcal{I}_\varphi(A)$ for all atoms A. For quantified formulae we have

$$\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathcal{I}^3_\varphi(\forall x.\, \mathfrak{D}(x) \Rightarrow \Re^{\mathfrak{D}}(\Psi)) = \widetilde{\forall}(\Theta^3)\ ,$$

where $\Theta^3 := \{\mathcal{I}^3_\psi((\mathfrak{D}(x)) \Rightarrow \Re^{\mathfrak{D}}(\Psi)) \mid a \in \mathcal{A}^3\}$ and $\psi := \varphi, [a/x]$. On the other hand

$$\mathcal{I}_\varphi(\forall x_{\mathfrak{D}}.\, \Psi) = \widetilde{\forall}\{\mathcal{I}_\psi(\Psi) \mid a \in \mathcal{A}\} = \widetilde{\forall}(\Theta)$$

Now $\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathcal{I}^3_\varphi(\forall x.\, \mathfrak{D}(X) \Rightarrow \Re^{\mathfrak{D}}(\Psi))$
$= \widetilde{\forall}(\{\mathcal{I}^3_{\varphi,[a/x]}(\mathfrak{D}(X) \Rightarrow \Re^{\mathfrak{D}}(\Psi)) \mid a \in \mathcal{A}^3\})$,
so we have to consider the following cases for a. If $a = \bot$, then $\mathcal{I}^3_\psi(\mathfrak{D}(x)) = \mathsf{f}$ and therefore $\mathcal{I}^3_\psi(\mathfrak{D}(x) \Rightarrow \Re^{\mathfrak{D}}(\Psi)) = \mathsf{t}$. If $a \neq \bot$, then by inductive hypothesis $\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\Psi)) = \mathcal{I}_\varphi(\Psi)$ and therefore $\Theta^3 = \Theta \cup \{\mathsf{t}\}$.

$\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathsf{t}$	iff	$\Theta^3 = \Theta = \{\mathsf{t}\}$ or $\emptyset$	iff	$\mathcal{I}_\varphi(\forall x_{\mathfrak{D}}.\, \Psi) = \mathsf{t}$
$\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathsf{u}$	iff	$\Theta^3 = \Theta = \{\mathsf{u}, \mathsf{t}\}$ or $\{\mathsf{u}\}$	iff	$\mathcal{I}_\varphi(\forall x_{\mathfrak{D}}.\, \Psi) = \mathsf{u}$
$\mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathsf{f}$	iff	$\mathsf{f} \in \Theta^3 = \Theta \cup \{\mathsf{t}\}$	iff	$\mathcal{I}_\varphi(\forall x_{\mathfrak{D}}.\, \Psi) = \mathsf{f}$

Since $\Re^{\mathfrak{D}}$ is homomorphic for connectives, we have completed the induction, thus $\mathcal{M}^3 \models^{\mathbf{K}^3} \Re^{\mathfrak{D}}(\Phi)$ and we have proven the necessitation direction of the theorem.

For the proof of sufficiency let $\mathcal{M}^3 := (\mathcal{A}^3, \mathcal{I}^3)$ be a $\mathbf{K}^3$-model, such that $\mathcal{M}^3 \models \Re^{\mathfrak{D}}(\Phi) \cup \Re^{\mathfrak{D}}(\Sigma)$, note that in our case $\Re^{\mathfrak{D}} \circ \Re^{\mathcal{S}} = \Re^{\mathfrak{D}}$. Let

$$\mathcal{A} := \{a \in \mathcal{A}^3 \mid \mathcal{I}^3(\mathfrak{D})(a) = \mathsf{t}\} \quad \text{and} \quad \mathcal{A}_\perp := \{a \in \mathcal{A}^3 \mid \mathcal{I}^3(\mathfrak{D})(a) = \mathsf{f}\}$$

then $\mathcal{A}^3 = \mathcal{A} \cup \mathcal{A}_\perp$, since $\forall x\mathbin{.}\ \mathfrak{D}(x) \vee \neg\mathfrak{D}(x) \in \Re^{\mathfrak{D}}(\Sigma)$. If $\mathcal{A}_\perp = \emptyset$, then it is easy to construct a strict Σ-algebra from $(\mathcal{A}^3, \mathcal{I}^3)$ by extending $\mathcal{A}^3$ with $\perp$ and interpreting each function and predicate with the strict extension of its $\mathcal{I}^3$ value. So in the following we will assume that $\mathcal{A}_\perp$ is nonempty. Now let $\pi\colon \mathcal{A}^3 \longrightarrow \mathcal{A}^\perp$ be a function that is the identity on $\mathcal{A}$ and $\pi(a) = \perp$ for all $a \in \mathcal{A}_\perp$. As $\mathcal{M}^3 \models \Re^{\mathfrak{D}}(\Sigma)$, we know that $\mathcal{I}^3(f)(a_1, \ldots, a_n) \in \mathcal{A}_\perp$ if one $a_i \in \mathcal{A}_\perp$, so the following definition is well-defined.

$$\mathcal{I}(f)(\pi(a_1), \ldots, \pi(a_n)) := \pi(\mathcal{I}^3(f)(a_1, \ldots, a_n))$$

Now we will see that $\mathcal{I}_{\pi\circ\varphi}(t) = \pi(\mathcal{I}^3_\varphi(t))$ for all well-formed $\mathcal{SKL}$ terms t and assignments φ into $\mathcal{M}^3$.

1. $\mathcal{I}_{\pi\circ\varphi}(x) = \pi \circ \varphi(x) = \pi(\mathcal{I}^3_\varphi(x))$.
2. $\mathcal{I}_{\pi\circ\varphi}(c) = \mathcal{I}(c) = \pi(\mathcal{I}^3(c)) = \pi(\mathcal{I}^3_\varphi(c))$.
3. $\begin{aligned}\mathcal{I}_{\pi\circ\varphi}(f(t^1, \ldots, t^n)) &= \mathcal{I}(f)(\mathcal{I}_{\pi\circ\varphi}(t^1), \ldots, \mathcal{I}_{\pi\circ\varphi}(t^n)) \\ &= \mathcal{I}(f)(\pi(\mathcal{I}^3_\varphi(t^1)), \ldots, \pi(\mathcal{I}^3_\varphi(t^n))) \\ &= \pi(\mathcal{I}^3(f)(\mathcal{I}^3_\varphi(t^1), \ldots, \mathcal{I}^3_\varphi(t^n))) \\ &= \pi(\mathcal{I}^3_\varphi(f(t^1, \ldots, t^n)))\end{aligned}$

Similarly the definition

$$\mathcal{I}(p)(\pi(a_1), \ldots, \pi(a_n)) := \mathcal{I}^3(p)(a_1, \ldots, a_n)$$

is well-defined, because $\mathcal{M}^3 \models \Re^{\mathfrak{D}}(\Sigma)$ and gives us $\mathcal{I}_{\pi\circ\varphi}(A) = \mathcal{I}^3(\Re^{\mathfrak{D}}(A))$ for all atoms A. From this, we obtain the general result $\mathcal{I}_{\pi\circ\varphi}(\Phi) = \mathcal{I}^3_\varphi(\Re^{\mathfrak{D}}(\Phi))$ by treating quantified formulae by a case analysis just as in the necessitation direction. In particular we have $\mathcal{I}_{\pi\circ\varphi}(\Phi) = \mathsf{t}$, iff $\mathcal{I}^3_\varphi(\Phi) = \mathsf{t}$ and therefore $\mathcal{M} \models \Phi$, whenever $\mathcal{M}^3 \models \Re^{\mathfrak{D}}(\Phi)$. □

Corollary 2.15 *Let Φ be a set of sentences and A be a sentence, then the following are equivalent*

1. $\Phi \models A$ *in all* Σ*-models.*
2. $\Re^{\mathcal{S}}(\Phi) \cup \Re^{\mathcal{S}}(\Sigma) \models \Re^{\mathcal{S}}(A)$ *in all unsorted* $\Sigma \cup \mathcal{S}^*$*-models.*
3. $\Re^{\mathfrak{D}} \circ \Re^{\mathcal{S}}(\Phi) \cup \Re^{\mathfrak{D}}(\Sigma \cup \mathcal{S}^*) \cup \Re^{\mathfrak{D}}(\Re^{\mathcal{S}}(\Sigma)) \models \Re^{\mathfrak{D}} \circ \Re^{\mathcal{S}}(A)$ *in all* $\mathbf{K}^3$*-models.*

As a consequence of the sort theorem, the standard operationalization for many-valued logics [BF92, Car87, Car91, Häh92] can be utilized to mechanize strong sorted Kleene logic and in fact the system of Lucio-Carrasco and Gavilanes-Franco [LCGF89] can be seen as a standard many-valued tableau operationalization [Häh92, BFZ93] of the relativization of $\mathcal{SKL}$. However, as the extended example shows, we can do better by using sorted methods, since relativization expands the size and number of input formulae and furthermore

expands the search spaces involved in automatic theorem proving by building up many meaningless branches. Note that already the formulation of $\mathcal{SKL}$ where we only have the required sort $\mathfrak{D}$ is much more concise than the relativized version. Furthermore we will see that the theory of definedness is treated goal-driven by the $\mathcal{TPF}$ calculus (cf. section 3). Thus the $\mathcal{TPF}$ calculus is closer to informal practice than the relativization in this respect.

Example 2.16 (continuing 2.12)
The relativization $\Re^{\mathcal{S}} \circ \Re^{\mathfrak{D}}$ of the $\mathcal{SKL}$-consequent $\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}, \mathrm{A4}, \mathrm{A5}\} \models \mathrm{T}$ is the $\mathbf{K}^3$-consequent $\{\mathrm{R1}, \mathrm{R2}, \mathrm{R3}, \mathrm{R4}, \mathrm{R5}, \mathrm{R}^{\mathsf{IR}}, \mathrm{R}^{\mathsf{IR}^*}, \mathrm{R}^{=}, \mathrm{R}^{>}, \mathrm{R}^{-}, \mathrm{R}^{/}, \mathrm{R}^{2}, \mathfrak{D}^{!}\} \models \mathrm{RT}$ with the following relativized formulae:

$\mathrm{R1}$ $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathsf{IR}(x) \Rightarrow (x \neq 0 \Rightarrow \mathsf{IR}^*(x)))$
$\mathrm{R2}$ $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathsf{IR}^*(x) \Rightarrow \mathsf{IR}^*(\frac{1}{x}))$
$\mathrm{R3}$ $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathsf{IR}^*(x) \Rightarrow x^2 > 0)$
$\mathrm{R4}$ $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathsf{IR}(x) \Rightarrow (\forall y.\, \mathfrak{D}(y) \Rightarrow (\mathsf{IR}(y) \Rightarrow \mathsf{IR}(x - y))))$
$\mathrm{R5}$ $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathsf{IR}(x) \Rightarrow (\forall y.\, \mathfrak{D}(y) \Rightarrow (\mathsf{IR}(y) \Rightarrow (x - y = 0 \Rightarrow x = y))))$
RT $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathsf{IR}(x) \Rightarrow (\forall y.\, \mathfrak{D}(y) \Rightarrow (\mathsf{IR}(y) \Rightarrow (x \neq y \Rightarrow \left(\frac{1}{x-y}\right)^2 > 0))))$

The set of signature axioms $\Re^{\mathfrak{D}}(\Sigma \cup \mathcal{S}^*) \cup \Re^{\mathfrak{D}}(\Re^{\mathcal{S}}(\Sigma))$ is the following set of $\mathbf{K}^3$-formulae:

R^{IR} $\forall x.\, \mathfrak{D}(x) \Rightarrow !\mathsf{IR}(x)$
$\mathrm{R}^{\mathsf{IR}^*}$ $\forall x.\, \mathfrak{D}(x) \Rightarrow !\mathsf{IR}^*(x)$
$\mathrm{R}^{=}$ $\forall x, y.\, (x = y \vee x \neq y) \Rightarrow \mathfrak{D}(x) \wedge \mathfrak{D}(y)$
$\mathrm{R}^{>}$ $\forall x, y.\, (x > y \vee x \not> y) \Rightarrow \mathfrak{D}(x) \wedge \mathfrak{D}(y)$
R^{-} $\forall x, y.\, \mathfrak{D}(x - y) \Rightarrow \mathfrak{D}(x) \wedge \mathfrak{D}(y)$
$\mathrm{R}^{/}$ $\forall x.\, \mathfrak{D}(\frac{1}{x}) \Rightarrow \mathfrak{D}(x)$
R^{2} $\forall x.\, \mathfrak{D}(x^2) \Rightarrow \mathfrak{D}(x)$
$\mathfrak{D}^{!}$ $\forall x.\, \mathfrak{D}(x) \vee \neg\mathfrak{D}(x)$

2.3 Model Existence

In this subsection we introduce an important tool for proving the completeness of calculi. The importance of model existence theorems lies in the fact that they abstract over the model theoretic part of various completeness proofs. Such theorems were first introduced by Smullyan (who calls them unifying principles) in [Smu63, Smu68] based on work by Hintikka and Beth.

Definition 2.17 Let ∇ be a class of sets.

1. ∇ is called *closed under subsets*, iff for all sets S and T the following condition holds: if $S \subset T$ and $T \in \nabla$, then $S \in \nabla$.
2. ∇ is called *compact*, iff for every set S the following condition holds: $S \in \nabla$, iff every finite subset of S is a member of ∇.

Lemma 2.18 *If ∇ is compact, then ∇ is closed under subsets.*

Proof: Suppose $S \subset T$ and $T \in \nabla$. Every finite subset A of S is a finite subset of T, and since ∇ is compact, we know that $A \in \nabla$. Thus $S \in \nabla$. □

Definition 2.19 (Labeled Formula) We will call a pair A^α, where A is an $\mathcal{SKL}$-formula and $\alpha \in \{\mathsf{f}, \mathsf{u}, \mathsf{t}\}$ a *labeled formula*. We say that a Σ-assignment φ satisfies a set Φ of labeled formulae in a strict Σ-algebra $\mathcal{M} = (\mathcal{A}, \mathcal{I})$, if $\mathcal{I}_\varphi(A) = \alpha$ for all $A^\alpha \in \Phi$.

In the following we will use $\Phi * A$ as an abbreviation for $\Phi \cup \{A\}$ in order to increase the legibility.

Definition 2.20 (Abstract Consistency Class) A class ∇ of sets of labeled formulae is called an *abstract consistency class*, iff it is closed under subsets, and for all sets $\Phi \in \nabla$ the following conditions hold:

1. If A is atomic, then $A^\alpha \in \Phi$ for at most one $\alpha \in \{\mathsf{f}, \mathsf{u}, \mathsf{t}\}$, furthermore for all terms t the literal $(t{\leqslant}\mathfrak{D})^\mathsf{u}$ is not in Φ.
2. If $(\neg A)^\alpha \in \Phi$, then $\Phi * A^\beta \in \nabla$, where $\beta = \mathsf{t}$, if $\alpha = \mathsf{f}$; $\beta = \mathsf{f}$, if $\alpha = \mathsf{t}$; and $\beta = \mathsf{u}$ else.
3. If $(!A)^\mathsf{t} \in \Phi$, then $\Phi * A^\gamma \in \nabla$ for some $\gamma \in \{\mathsf{t}, \mathsf{f}\}$; if $(!A)^\mathsf{f} \in \Phi$, then $\Phi * A^\mathsf{u} \in \nabla$. $(!A)^\mathsf{u}$ is not in any $\Phi \in \nabla$.
4. If $(A \vee B)^\alpha \in \Phi$, then
   $\alpha = \mathsf{t})$ $\Phi * A^\mathsf{t} \in \nabla$ or $\Phi * B^\mathsf{t} \in \nabla$.
   $\alpha = \mathsf{u})$ $\Phi \cup \{A^\mathsf{f}, B^\mathsf{u}\} \in \nabla$, or $\Phi \cup \{A^\mathsf{u}, B^\mathsf{f}\} \in \nabla$, or $\Phi \cup \{A^\mathsf{u}, B^\mathsf{u}\} \in \nabla$.
   $\alpha = \mathsf{f})$ $\Phi \cup \{A^\mathsf{f}, B^\mathsf{f}\} \in \nabla$
5. If $\forall x_S . A \in \Phi$, then
   $\alpha = \mathsf{t})$ for any term t, $\Phi * ([t/x_S]A)^\mathsf{t} \in \nabla$ or $\Phi * (t{\leqslant}S)^\alpha \in \nabla$ for some $\alpha \in \{\mathsf{f}, \mathsf{u}\}$.
   $\alpha = \mathsf{u})$ for any term t, and any constant c that does not occur, $\Phi \cup \{([c/x_S]A)^\mathsf{u}, (c{\leqslant}S)^\mathsf{t}, \mathcal{A}\} \in \nabla$, where $\mathcal{A}$ is $([t/x_S]A)^\mathsf{t}$ or $([t/x_S]A)^\mathsf{u}$ or $(t{\leqslant}S)^\mathsf{f}$ or $(t{\leqslant}S)^\mathsf{u}$.
   $\alpha = \mathsf{f})$ $\Phi \cup \{([c/x]A)^\mathsf{f}, (c{\leqslant}S)^\mathsf{t}\} \in \nabla$, for each constant c that does not occur in Φ.
6. If $A^\gamma \in \Phi$ with $\gamma \in \{\mathsf{f}, \mathsf{t}\}$, then $\Phi * (t{\leqslant}\mathfrak{D})^\mathsf{t} \in \nabla$, for all subterms t of A.
7. If $(t{\leqslant}S)^\mathsf{u} \in \Phi$, then $\Phi * (t{\leqslant}\mathfrak{D})^\mathsf{f} \in \nabla$.

Theorem 2.21
For each abstract consistency class ∇ there exists an abstract consistency class ∇' such that $\nabla \subset \nabla'$, and ∇' is compact.

Proof: (following [And86]) Let $\nabla' := \{\Phi \mid \text{every finite subset of } \Phi \text{ is in } \nabla\}$. To see that $\nabla \subset \nabla'$, suppose that $\Phi \in \nabla$. ∇ is closed under subsets, so every finite subset of Φ is in ∇, and thus $\Phi \in \nabla'$.

Next let us show that ∇' is compact. Suppose $\Phi \in \nabla'$ and Ψ is an arbitrary finite subset of Φ. By definition of ∇' all finite subsets of Ψ are in ∇, and therefore $\Psi \in \nabla'$. Thus all finite subsets of Φ are in ∇' whenever Ψ is in ∇'. On the other

hand, suppose all finite subsets of Ψ are in ∇'. Then by the definition of ∇' the finite subsets of Ψ are also in ∇, so $\Phi \in \nabla'$. Thus ∇' is compact.

Finally we show that ∇' is an abstract consistency class. By lemma 2.18 it is closed under subsets. Of the conditions for the abstract consistency class we will only explicitly present the first two cases, since the proofs of the others are analogous. Let $\Phi \in \nabla'$ be given arbitrarily.

Suppose there is an atom A, such that $\{A^\alpha, A^\beta\} \subseteq \Phi$ for $\alpha \neq \beta$. By the definition of ∇' we get $\{A^\alpha, A^\beta\} \in \nabla$ contradicting 2.20(1).

Let $(\neg A)^\alpha \in \Phi$, and Ψ be any finite subset of $\Phi * A^\beta$ (where α and β are as in 2.20(2)) and let $\Theta := (\Psi \setminus \{A^\beta\}) * (\neg A)^\alpha$. Θ is a finite subset of Φ, so $\Theta \in \nabla$. Since ∇ is an abstract consistency class and $(\neg A)^\alpha \in \Theta$, we get $\Theta * A^\beta \in \nabla$. We know that $\Psi \subset \Theta * A^\beta$, and ∇ is closed under subsets, so $\Psi \in \nabla$. Thus every finite subset Ψ of $\Phi * A^\beta$ is in ∇, therefore by definition $\Phi * A^\beta \in \nabla'$. □

Definition 2.22 (Σ-Hintikka Set) Let ∇ be an abstract consistency class and $\Phi \in \nabla$. Then $\mathcal{H} \in \nabla$ is called a ∇-*extension of* Φ, iff $\Phi \subset \mathcal{H}$. A set $\mathcal{H}$ is called *maximal in* ∇, iff for each formula $D \in \nabla$ such that $\mathcal{H} * D \in \nabla$, we already have $D \in \mathcal{H}$. A set $\mathcal{H} \in \nabla$ is called a Σ-*Hintikka set for* ∇ *and* Φ, iff $\mathcal{H}$ is maximal in ∇ and $\Phi \subseteq \mathcal{H}$.

We now give some technical properties of Σ-Hintikka sets that are useful for manipulating formulae.

Theorem 2.23 *If ∇ is an abstract consistency class, and $\mathcal{H}$ is maximal in ∇, then the following statements hold:*

1. *If A is a proposition, then $A^\alpha \in \mathcal{H}$ for at most one $\alpha \in \{\mathsf{f}, \mathsf{u}, \mathsf{t}\}$. Furthermore $(t{\Leftarrow}\mathfrak{D})^\mathsf{u} \notin \mathcal{H}$ for all t.*
2. *If $(\neg A)^\alpha \in \mathcal{H}$, then $A^\beta \in \mathcal{H}$, where $\beta = \mathsf{t}$, if $\alpha = \mathsf{f}$; $\beta = \mathsf{f}$, if $\alpha = \mathsf{t}$; and $\beta = \mathsf{u}$ else.*
3. *If $(!A)^\alpha \in \mathcal{H}$, then either $\alpha = \mathsf{t}$ and $A^\gamma \in \mathcal{H}$ for $\gamma \in \{\mathsf{f}, \mathsf{t}\}$ or $\alpha = \mathsf{f}$ and $A^\mathsf{u} \in \mathcal{H}$. In particular, there is no formula B, such that $(!B)^\mathsf{u} \in \mathcal{H}$.*
4. *If $(A \vee B)^\alpha \in \mathcal{H}$, then*
   $\alpha = \mathsf{t}$) $A^\mathsf{t} \in \mathcal{H}$ or $B^\mathsf{t} \in \mathcal{H}$.
   $\alpha = \mathsf{u}$) $A^\mathsf{f}, B^\mathsf{u} \in \mathcal{H}$, or $A^\mathsf{u}, B^\mathsf{f} \in \mathcal{H}$, or $A^\mathsf{u}, B^\mathsf{u} \in \mathcal{H}$.
   $\alpha = \mathsf{f}$) $A^\mathsf{f}, B^\mathsf{f} \in \mathcal{H}$
5. *If $\forall x_S . A \in \mathcal{H}$, then*
   $\alpha = \mathsf{t}$) for any term t, $[t/x_S]A^\mathsf{t} \in \mathcal{H}$ or $(t{\Leftarrow}S)^\alpha \in \mathcal{H}$ for some $\alpha \in \{\mathsf{f}, \mathsf{u}\}$.
   $\alpha = \mathsf{u}$) for any term t, there is a term s, with $\Phi \cup \{([s/x_S]A)^\mathsf{u}, (s{\Leftarrow}S)^\mathsf{t}, \mathcal{A}\} \in \mathcal{H}$, where $\mathcal{A} \in \{([t/x_S]A)^\mathsf{t}, ([t/x_S]A)^\mathsf{u}, (t{\Leftarrow}S)^\mathsf{f}\}$.
   $\alpha = \mathsf{f}$) there is a term t, such that $([t/x]A)^\mathsf{f}, (t{\Leftarrow}S)^\mathsf{t} \in \mathcal{H}$.
6. *If $A^\gamma \in \mathcal{H}$ with $\gamma \in \{\mathsf{t}, \mathsf{f}\}$, then $(t{\Leftarrow}\mathfrak{D})^\mathsf{t} \in \mathcal{H}$, for all subterms t of A.*
7. *If $(t{\Leftarrow}S)^\mathsf{u} \in \mathcal{H}$, then $(t{\Leftarrow}\mathfrak{D})^\mathsf{f} \in \mathcal{H}$.*

Proof: We prove the first assertion by induction on the structure of A. If A is atomic, then the assertion is a simple consequence of 2.20(1).

Let $A = \neg B$ and $A^{\mathsf{f}}, A^{\mathsf{t}}$ be in $\mathcal{H}$. By 2.20(2) we have $B^{\mathsf{f}}, B^{\mathsf{t}} \in \mathcal{H}$ contradicting the induction hypothesis. The remaining cases can be shown analogously, so we have proven the first assertion.

The rest of the assertions are all of the same form, and have analogous proofs, therefore we only prove the second. If $(\neg A)^{\mathsf{f}} \in \mathcal{H}$, then $\mathcal{H} * A^{\mathsf{t}} \in \nabla$ (∇ is an abstract consistency class). The maximality of $\mathcal{H}$ now gives the assertion. □

Lemma 2.24 (Hintikka Lemma) *If ∇ is an abstract consistency class and $\mathcal{H}$ is maximal in ∇, then there is an $\mathcal{SKL}$-model $\mathcal{M}$ and a Σ-assignment φ, such that φ satisfies $\mathcal{H}$ in $\mathcal{M}$.*

Proof: We prove the assertion by constructing a model $\mathcal{M} = (\mathcal{A}, \mathcal{I})$ for $\mathcal{H}$, which is derived from the ground term algebra.

Let $\mathcal{T}_\perp$ be the set of closed well-formed terms together with $\perp$. In order to construct the carrier $\mathcal{A}$, we have to identify all elements in $\mathcal{T}_\perp$ that are undefined ($(t{\lessdot}\mathfrak{D})^{\mathsf{f}} \in \mathcal{H}$) and identify them with $\perp$. Traditional proofs of the Hintikka-Lemma for total-function logics now define $\mathcal{I}: \mathcal{F} \longrightarrow \mathcal{A}$ to be the identity map. However this definition does not make $\mathcal{I}(f)$ strict, since $\mathcal{I}(f)(\perp) = f(\perp) \neq \perp$. To repair this defect we take the carrier $\mathcal{A}$ to be the quotient of $\mathcal{T}_\perp$ with respect to the equality theory $=_\perp$ induced by the set

$$E_\perp = \{t =_\perp \perp \;\big|\; (t{\lessdot}\mathfrak{D})^{\mathsf{f}} \in \mathcal{H}\} \cup \{f^k(x_1, \ldots, \perp, \ldots, x_k) = \perp \;\big|\; f^k \in \Sigma^k\}$$

of equations. Thus $\mathcal{A}$ is the set of equivalence classes $[\![t]\!]_\perp = \{s \;\big|\; E_\perp \models s =_\perp t\}$. The function $f_\perp: ([\![t_1]\!]_\perp, \ldots, [\![t_1]\!]_\perp) \mapsto [\![f(t_1, \ldots, t_n)]\!]_\perp$ is a well-defined function, since $=_\perp$ is a congruence relation. We define $\mathcal{I}(f) := f_\perp$ and note that the special construction of $E_\perp$ entails the strictness of $f_\perp$.

For any finite set $\mathcal{W}$ of variables a Σ-assignment φ can be restricted to a substitution $\varphi\mathcal{W} = \varphi\big|_{\mathcal{W}}$. A simple induction on the structure of a term t can be used to show that $\mathcal{I}_\varphi(t) = \mathcal{I}_{\varphi\mathbf{Free}_{(t)}}(t)$.

For $P \in \mathcal{P}^n$ let $\mathcal{I}(P): \mathcal{A}^n \longrightarrow \{\mathsf{f}, \mathsf{u}, \mathsf{t}\}$ with $P_\mathcal{H}([\![t^1]\!]_\perp, \ldots, [\![t^n]\!]_\perp) = \alpha$, iff $P(t^1, \ldots, t^n)^\alpha \in \mathcal{H}$. Clearly $P_\mathcal{H}$ is a partial function (cf. 2.23.1), since the definition only depends on $=_\perp$-equivalence classes. With the help of 2.23.6 it is easy to see that $P_\mathcal{H}$ is a strict function. We can extend $P_\mathcal{H}$ to a total strict function $\mathcal{I}(P)$ by evaluating all remaining proper atoms with u and all remaining sort atoms with f. Thus sorts are everywhere defined (the value u is only obtained on $\perp$) and the strictness of the predicates is preserved.

Clearly, this construction entails that for any atom $A \in \mathcal{H}$ and any Σ-assignment φ we have $\mathcal{I}_\varphi(A) = \alpha$, iff $\mathcal{I}_{\varphi\mathbf{Free}_{(A)}}(A) \in \mathcal{H}$. Now a simple induction on the number of connectives and quantifications, using the properties of 2.23 can be used to extend this property to arbitrary formulae. Thus we have $\mathcal{I}_\varphi(A) = \alpha$ for all $A^\alpha \in \mathcal{H}$, if we take φ to be the identity. □

We now come to the proof of the abstract extension lemma, which nearly immediately yields the model existence theorem.

Theorem 2.25 (Abstract Extension Lemma) *Let ∇ be a compact abstract consistency class, and let $H \in \nabla$ be a set of propositions. Then there exists a Σ-Hintikka set $\mathcal{H}$ for ∇ and H.*

Proof: We will construct $\mathcal{H}$ by inductively constructing a sequence of sets $\mathcal{H}^i$ and taking $\mathcal{H} := \bigcup_{i \in \mathbb{N}} \mathcal{H}^i \in \nabla$. We can arrange all labeled formulae in an infinite sequence $C^1, C^2, \ldots$. For each $n \in \mathbb{N}$ we inductively define a set $\mathcal{H}^n$ of propositions by

1. $\mathcal{H}^0 := \Phi$.
2. If $\mathcal{H}^n * C^n \notin \nabla$, then $\mathcal{H}^{n+1} := \mathcal{H}^n$.
3. If $\mathcal{H}^n * C^n \in \nabla$, and C^n is of the form $(\forall x_S\, A)^\alpha$ with $\alpha \in \{\mathsf{f}, \mathsf{u}\}$ then $\mathcal{H}^{n+1} := \mathcal{H}^n \cup \{C^n, [c^n/x]A^\alpha, (c \ll S)^\mathsf{t}\}$.
4. $\mathcal{H}^{n+1} := \mathcal{H}^n * C^n$ else.

Let $\mathcal{H} := \bigcup_{n \in \mathbb{N}} \mathcal{H}^n$. Clearly each of the $\mathcal{H}^n \in \nabla$, and therefore $\mathcal{H} \in \nabla$, since ∇ is compact.

In order to prove the maximality of $\mathcal{H}$, let A be an arbitrary proposition such that $\mathcal{H} * A \in \nabla$. We know that $A = C^n$ for some $n \in \mathbb{N}$, so $\mathcal{H}^n * A \subset \mathcal{H} * A \in \nabla$ and $\mathcal{H}^n * A \in \nabla$, since ∇ is closed under subsets. Hence by definition we know that $A \in \mathcal{H}^{n+1}$, and therefore $A \in \mathcal{H}$. □

Corollary 2.26 (Model Existence) *Let $\Phi \in \nabla$ and ∇ be an abstract consistency class, then there is an $\mathcal{SKL}$-model $\mathcal{M}$ and a Σ-assignment φ, such that φ satisfies Φ in $\mathcal{M}$.*

Proof: Let ∇' be the compact abstract consistency class of theorem 2.21 and let $\mathcal{H}$ be the maximal ∇-extension of Φ guaranteed by 2.25. Furthermore let $\mathcal{M}$ be the $\mathcal{SKL}$-model, and φ the Σ-assignment for $\mathcal{H}$ guaranteed by 2.24. Then φ satisfies Φ in $\mathcal{M}$, since $\Phi \subseteq \mathcal{H}$. □

3 Tableau

Now we turn to the exposition of our tableau calculus. The case of standard tableaux for partial functions is a simple extension of first-order tableau methods to $\mathcal{SKL}$. Therefore we will only concern ourselves with free variable tableaux.

While a labeled formula A^α means that A has the truth value α, we also make use of multi-indices as introduced by Hähnle and write $A^{\alpha\beta}$ as an abbreviation for $A^\alpha \vee A^\beta$ (Normally, we do not have to consider three different truth values, since the corresponding formulae are tautological and cannot contribute to refutations.) As has been pointed out by Hähnle [Häh92], the use of multi-indices does not only offer a concise notation, but can drastically improve a calculus, when special rules for their treatment are introduced. In the following, we add corresponding rules for handling multi-indices, where one label is u. Although not necessary in principle, this treatment results in a significant improvement of the search complexity of the calculus, which can thereby be reduced to the complexity in the two-valued case. This relationship will be made formal in theorem 3.9.

Definition 3.1 (Tableau Rules) The tableau rules consist of the traditional tableau rules for the propositional connectives, augmented by the case of the label u.

$$\frac{(A \wedge B)^{\mathsf{t}}}{\begin{array}{c} A^{\mathsf{t}} \\ B^{\mathsf{t}} \end{array}} \qquad \frac{(A \wedge B)^{\mathsf{u}}}{\begin{array}{c} A^{\mathsf{ut}} \\ B^{\mathsf{ut}} \\ A^{\mathsf{u}} \mid B^{\mathsf{u}} \end{array}} \qquad \frac{(A \wedge B)^{\mathsf{f}}}{A^{\mathsf{f}} \mid B^{\mathsf{f}}} \qquad \frac{(A \wedge B)^{\mathsf{ut}}}{\begin{array}{c} A^{\mathsf{ut}} \\ B^{\mathsf{ut}} \end{array}} \qquad \frac{(A \wedge B)^{\mathsf{fu}}}{A^{\mathsf{fu}} \mid B^{\mathsf{fu}}}$$

Since we have special rules for the multi-indices ut and fu, we only need a splitting rule reflecting the definition of multi-indices as disjunctions for the remaining multi-index ft. Note that the multi-index fut gives rise to tautologies, which can never contribute to refutations.

$$\frac{A^{\mathsf{ft}}}{A^{\mathsf{f}} \mid A^{\mathsf{t}}}$$

The negation rules and those for ! just flip the labels in the intuitive way.

$$\frac{(\neg A)^{\mathsf{t}}}{A^{\mathsf{f}}} \qquad \frac{(\neg A)^{\mathsf{u}}}{A^{\mathsf{u}}} \qquad \frac{(\neg A)^{\mathsf{f}}}{A^{\mathsf{t}}} \qquad \frac{(\neg A)^{\mathsf{ut}}}{A^{\mathsf{fu}}} \qquad \frac{(\neg A)^{\mathsf{fu}}}{A^{\mathsf{ut}}}$$

The ! rule for the u case closes the branch (we use an explicit symbol $*$ for that), since $(!A)^{\mathsf{u}}$ is unsatisfiable in $\mathcal{SKL}$.

$$\frac{(!A)^{\mathsf{t}}}{A^{\mathsf{ft}}} \qquad \frac{(!A)^{\mathsf{u}}}{*} \qquad \frac{(!A)^{\mathsf{f}}}{A^{\mathsf{u}}} \qquad \frac{(!A)^{\mathsf{ut}}}{A^{\mathsf{ft}}} \qquad \frac{(!A)^{\mathsf{fu}}}{A^{\mathsf{u}}}$$

In order to simplify the presentation of the examples we also (redundantly) present the rules for disjunction.

$$\frac{(A \vee B)^{\mathsf{t}}}{A^{\mathsf{t}} \mid B^{\mathsf{t}}} \qquad \frac{(A \vee B)^{\mathsf{u}}}{\begin{array}{c} A^{\mathsf{fu}} \\ B^{\mathsf{fu}} \\ A^{\mathsf{u}} \mid B^{\mathsf{u}} \end{array}} \qquad \frac{(A \vee B)^{\mathsf{f}}}{\begin{array}{c} A^{\mathsf{f}} \\ B^{\mathsf{f}} \end{array}} \qquad \frac{(A \vee B)^{\mathsf{ut}}}{A^{\mathsf{ut}} \mid B^{\mathsf{ut}}} \qquad \frac{(A \vee B)^{\mathsf{fu}}}{\begin{array}{c} A^{\mathsf{fu}} \\ B^{\mathsf{fu}} \end{array}}$$

The quantifier rules for the classical truth values and multi-indices are very similar to the standard rules[1] ($\{x_S, y^1, \ldots, y^n\}$ are the free variables of A and f is a new function symbol of arity n), with the exception that the sort of the Skolem function has to be specified. The rule for the case u has a mixed existential and universal character: for y_S the value of A is undefined or true

[1] We employ the liberalized δ-rule of [HS94].

(that is there is no instance, which makes the formula false) *and* there is at least one witness for the undefinedness.

$$\frac{(\forall x_S{\scriptstyle\blacksquare}\, A)^{\mathsf{t}}}{[y_S/x_S]A^{\mathsf{t}}} \qquad \frac{(\forall x_S{\scriptstyle\blacksquare}\, A)^{\mathsf{u}}}{\begin{array}{c}[f(y^1,\ldots,y^n)/x_S]A^{\mathsf{u}}\\ [y_S/x_S]A^{\mathsf{ut}}\\ (f(y^1,\ldots,y^n){\leqslant}S)^{\mathsf{t}}\end{array}} \qquad \frac{(\forall x_S{\scriptstyle\blacksquare}\, A)^{\mathsf{f}}}{\begin{array}{c}[f(y^1,\ldots,y^n)/x_S]A^{\mathsf{f}}\\ (f(y^1,\ldots,y^n){\leqslant}S)^{\mathsf{t}}\end{array}}$$

$$\frac{(\forall x_S{\scriptstyle\blacksquare}\, A)^{\mathsf{ut}}}{[y_S/x_S]A^{\mathsf{ut}}} \qquad \frac{(\forall x_S{\scriptstyle\blacksquare}\, A)^{\mathsf{fu}}}{\begin{array}{c}[f(y^1,\ldots,y^n)/x_S]A^{\mathsf{fu}}\\ (f(y^1,\ldots,y^n){\leqslant}S)^{\mathsf{t}}\end{array}}$$

The rules for connectives and quantifiers above can now be used to reduce complex labeled formulae to literals. Some sort literals can further be reduced, due to the fact that sorts are defined on all defined individuals and the predicate $\mathfrak{D}$ is defined everywhere. (These rules have to be slightly generalized for multi-indices. We only display the interesting case.)

$$\frac{(t{\leqslant}\mathfrak{D})^{\mathsf{u}}}{*} \qquad \frac{(t{\leqslant}S)^{\mathsf{u}}}{(t{\leqslant}\mathfrak{D})^{\mathsf{f}}}$$

Now we only need tableau closure rules: The *cut* rule and the *strict* rule

$$\frac{\begin{array}{c}A^{\alpha}\\ B^{\beta}\end{array}}{*\;\big|\;\mathscr{C}(\sigma)}\;\sigma \qquad \frac{\begin{array}{c}C^{\gamma}\\ (t{\leqslant}\mathfrak{D})^{\mathsf{f}}\end{array}}{*\;\big|\;\mathscr{C}(\sigma)}\;\sigma$$

where $\alpha \cap \beta = \emptyset$, $\gamma \subseteq \{\mathsf{ft}\}$, and $\sigma = [t_1/x^1_{S_1}], \ldots, [t_n/x^n_{S_n}]$ is the most general unifier of A and B or the most general unifier of the term t and a subterm s of C, respectively. In both cases the *sort constraint* $\mathscr{C}(\sigma) = ((t_1{\leqslant}S_1){\wedge}\ldots{\wedge}(t_n{\leqslant}S_n))^{\mathsf{fu}}$ insures the correctness (in terms of the sorts) of the instantiations. We have employed the notation of writing the substitution σ next to the tableau schema, to indicate that the whole tableau is instantiated by σ during the application of the rule.

A tableau is built up by constructing a tree with the tableau rules starting with an initial tree without branchings. We call a tableau *closed*, iff all of its branches end in $*$. Note that the disjunct $*$ in the succedent of the rules above is only needed, if the set of sort constraints is empty. Then this rule closes the branch without residuating.

Remark 3.2 We could also have used a generalization of the cut rule of the form

$$\frac{\begin{array}{c}A^{\alpha}\\ B^{\beta}\end{array}}{A^{\alpha\cap\beta}\;\big|\;\mathscr{C}(\sigma)}\;\sigma$$

where we employ the convention that $A^{\emptyset} = *$, since this corresponds to the empty disjunction, which is unsatisfiable. However, it is not straightforward to see in which cases this variant of the cut rule is more efficient.

Definition 3.3 (Tableau Proof) A *tableau proof for a formula* A is a closed tableau constructed from the initial tree consisting of the labelled formula A^{fu}. A *tableau proof for a consequent* $\Phi \models A$ is a closed tableau constructed form $\Phi^{\mathsf{t}} \cup \{A^{\mathsf{fu}}\}$.

Remark 3.4 The tableau proof of a consequent $\Phi \models A$ essentially refutes the possibility that A can be undefined or false under the assumption of all formulae in Φ. By the quartum non datur rule, we can then conclude that A is entailed by Φ.

3.1 Soundness and Completeness

The soundness of the $\mathcal{TPF}$ rules can be verified by a tedious recourse to the semantics of the quantifiers and connectives. Completeness is proven by the standard argument using the model existence theorem for $\mathcal{SKL}$. For this, we first have prove a lifting theorem for $\mathcal{TPF}$

Theorem 3.5 (Tableau Lifting) *Let* $\Phi \models A$ *be a consequent and* θ *a substitution, then* $\Phi \models A$ *has a closed* $\mathcal{TPF}$*-tableau provided* $\theta(\Phi) \models \theta(A)$ *has one.*

Proof: Let $\mathcal{T}_\theta$ be a closed tableau for $\theta(\Phi) \models \theta(A)$, the claim is proven by an induction on the construction of $\mathcal{T}_\theta$ constructing a tableau $\mathcal{T}$ for $\Phi \models A$ that is tableau-isomorphic to $\mathcal{T}$. Concretely we have a tree-isomorphism $\omega : \mathcal{T} \longrightarrow \mathcal{T}_\theta$ between $\mathcal{T}_\theta$ and $\mathcal{T}$ that respects labels and is compatible with θ, that is, for any node $\mathcal{N}$ in $\mathcal{T}$ with labeled formula A^α we have $\omega_{\mathcal{N}}(A) = \theta(A)$.

This induction is straightforward for all $\mathcal{TPF}$ rules except for the cut and the strict rules that residuate a sort constraint. In both cases, we can use a standard argument which we will only carry out for the cut case: σ is a most general unifier of $\theta(A)$ and $\theta(B)$ in $\mathcal{T}_\theta$, so $\sigma \circ \theta$ unifies A and B in $\mathcal{T}$. So there exists a most general unifier ρ of A and B, and a substitution τ with $\sigma \circ \theta = \tau \circ \rho$. Now we have $\tau(\mathcal{C}(\rho)) = \mathcal{C}(\tau \circ \rho) = \mathcal{C}(\sigma \circ \theta)$, so we obtain the assertion by the inductive hypothesis for $\rho(\mathcal{T})$ and $\tau \circ \rho(\mathcal{T}) = \sigma(\mathcal{T}_\theta)$. □

Theorem 3.6 (Completeness) $\mathcal{TPF}$ *is refutation complete, that is, if* $\Phi \models A$ *is a valid consequent, then there is a closed tableau for* $\Phi^{\mathsf{t}} \cup A^{\mathsf{fu}}$.

Proof: Completeness of $\mathcal{TPF}$ can be proven using the model existence theorem 2.26 by verifying that the class ∇ of sets Φ that do not have closed $\mathcal{TPF}$ tableaux is an abstract consistency class. This can be achieved with the usual techniques of e.g. [Fit90]: It is obvious that the $\mathcal{TPF}$ rules for the connectives, quantifiers and sorts directly correspond the clauses of 2.20. We have treated the only case where this correspondence is nontrivial (the quantifier case) in the tableau lifting theorem above. □

Example 3.7 (continuing 2.12) Taking the above example we give a proof for

$$\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}, \mathrm{A4}, \mathrm{A5}\} \models \mathrm{T}$$

using the above tableau rules. The proof is shown in figure 1. Applying the closure rule in the case of non-empty sort constraints, we omit the $*$ branch for simplicity reasons. Note that the unsorted unifiers $[c - d/u_{|\mathsf{R}}]$, $[c/x_{|\mathsf{R}}]$, and $[d/y_{|\mathsf{R}}]$ have to be applied to the whole tableau. For display reasons, however, we only add the relevant formulae to the tableau instead of replacing them, that is, correctly (F8) has to replace (F3) and (F13) to replace (F9).

The tableau proof can roughly be divided into three different parts, first the representation of the problem, displayed above the first line, second some initial simplification by eliminating quantifiers and connectives displayed between the first and the second line, and third the final refutation, below the second line.

Remark 3.8 The proof in figure 1 shows an interesting feature, namely it corresponds in length and structure exactly to a proof of the theorem in classical two-valued logic. By replacing all truth-value sets fu by the truth value f you get the corresponding two-valued proof. This correspondence is due to the correspondence of the tableau rules R^{α} and $R^{\alpha\mathsf{u}}$ for $\alpha \in \{\mathsf{f}, \mathsf{t}\}$ and $R \in \{\wedge, \vee, \neg, \forall\}$. In other words using rules for truth-value sets provide proofs as short as in the two-valued case. If, however, truth-value sets are not used, certain parts of the proofs must be duplicated. This relationship can only hold for so-called *normal problems* of course, that is, problems which do not contain any ! connective, since formulae containing a ! do not make any sense in classical two-valued logic.

Theorem 3.9 (Correspondence Theorem) *Each tableau proof for a normal problem $\Phi \models A$ in $\mathcal{SKL}$ can be isomorphically transformed into a tableau proof in $\mathcal{FOL}$.*

Proof: Let us prove the assumption by a case analysis on the rules applied in the proof. At a certain formula in the $\mathcal{SKL}$-tableau, its label set either contains the u value or not. If the formula does not contain u then it is labeled by t, by f, or by ft and will be treated by the same rule R^{α} with $R \in \{\wedge, \vee, \neg, \forall\}$ and $\alpha \in \{\mathsf{f}, \mathsf{t}\}$ or the splitting rule. Note the corresponding tableau rules are the same for $\mathcal{FOL}$ and $\mathcal{SKL}$.

In the case that α contains the truth value u, just eliminate u from the set. Since the initial problem formulation contains only the labels t and fu, for normal problems it inductively follows that no formula with the label u can occur in a tableau. The procedure of just eliminating the truth value u is correct, since for all connectives (with the exception of !, which may not occur in normal problems), all quantifier and all truth values holds, if R is a rule in $\mathcal{SKL}$ with a truth value set containing the truth value u, then a tableau rule of $\mathcal{FOL}$ can be constructed from R by eliminating the truth value u in the rule. For instance

$$\frac{(A \wedge B)^{\mathsf{ut}}}{\begin{array}{c} A^{\mathsf{ut}} \\ B^{\mathsf{ut}} \end{array}} \qquad \rightsquigarrow \qquad \frac{(A \wedge B)^{\mathsf{t}}}{\begin{array}{c} A^{\mathsf{t}} \\ B^{\mathsf{t}} \end{array}}$$

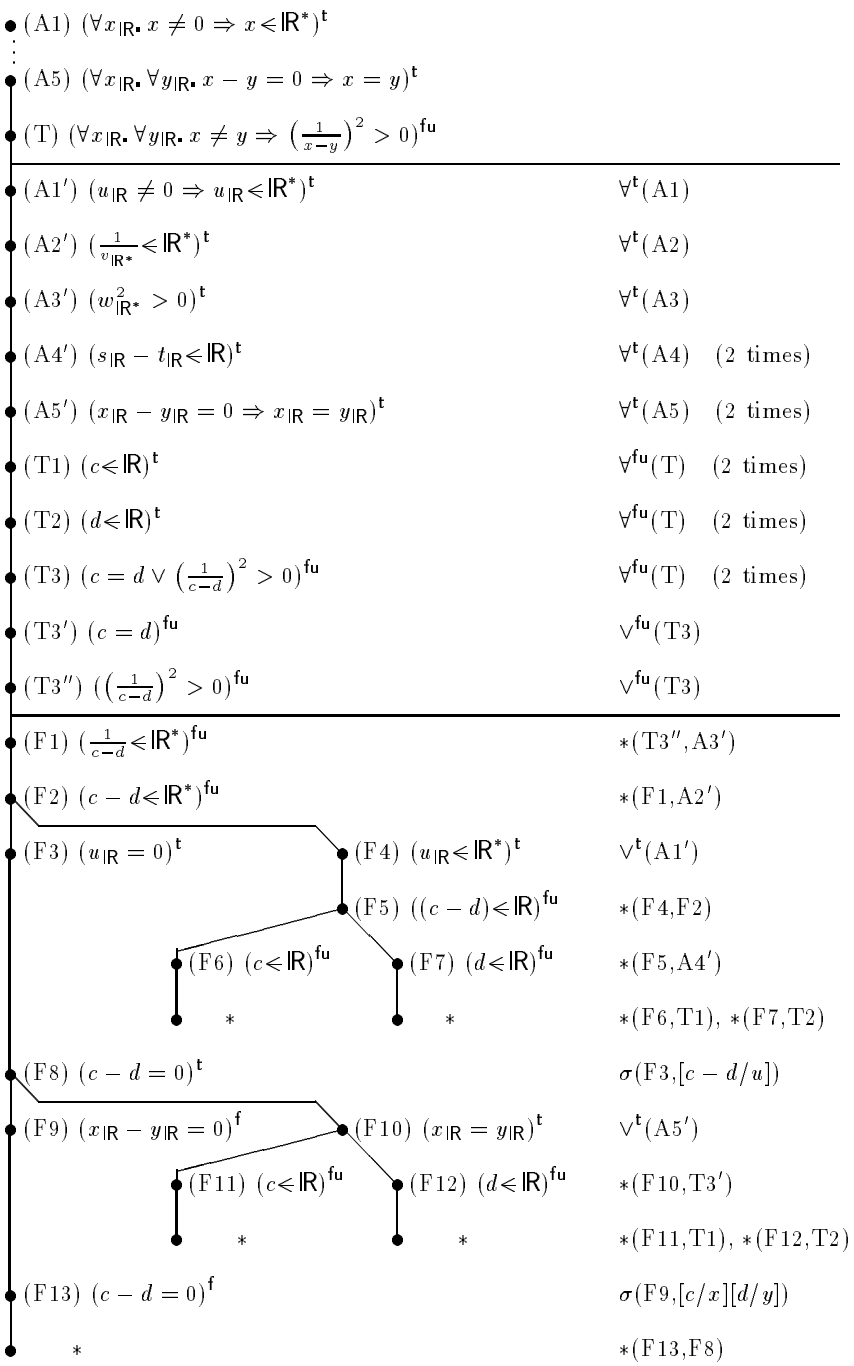


Fig. 1. Tableau proof with unsorted unification, example 3.7

For the other cases check this relation in definition 3.1. This relation holds also for the tableau closure rule.

Thus we get a $\mathcal{FOL}$ proof from the $\mathcal{SKL}$ proof by simply eliminating all truth values u. □

Remarks 3.10 Unfortunately, the converse of the above theorem does not hold. Not each $\mathcal{FOL}$ proof can be transformed into an $\mathcal{SKL}$ proof, even if there is an $\mathcal{SKL}$ proof. Consider for example the relation $\{A\} \models A \vee (B \vee \neg B)$ which holds in $\mathcal{SKL}$ as well as in $\mathcal{FOL}$. An $\mathcal{FOL}$-proof is:

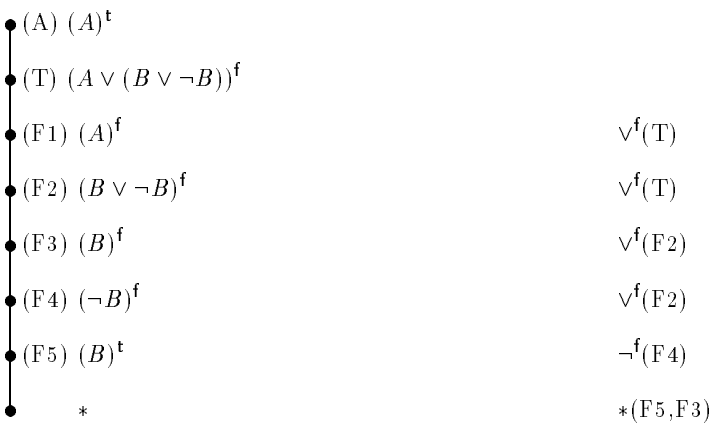


Fig. 2. Counterexample to the converse correspondence theorem

This proof cannot be transferred since in $\mathcal{SKL}$ (T), (F1), (F2), (F3), (F4), and (F5) are labeled by the truth value u in addition, hence the closure rule does not apply to (F5) and (F3). This comes from the fact that $B \vee \neg B$ is not a tautology in $\mathcal{SKL}$. However the other straightforward closure of the tableau by applying the closure rule to (A) and (F1) can be applied in $\mathcal{FOL}$ as well as in $\mathcal{SKL}$.

Of course it would be nice to have the property that for each classical $\mathcal{FOL}$ proof there exists an $\mathcal{SKL}$ proof which is as short as the classical (of course only if the classical theorem is also an $\mathcal{SKL}$ theorem). The example above shows that this property does not hold in general, for instance, replace the assumption set $\{A\}$ by a set from which A can be derived in 20 steps only. On the other hand this example is rather artificial insofar as the theorem would normally not be stated in this form in mathematics, because mathematical theorems are normally not redundant in the way that link two true statements by an "$\vee$", on the contrary *usual* mathematical theorems employ preconditions as weak as possible and consequences as strong as possible. For instance, in a mathematical context we would expect theorems like A, $B \vee \neg B$, $A \wedge (B \vee \neg B)$. While a proof

for the first (from the assumptions A) can be transferred from $\mathcal{FOL}$ to $\mathcal{SKL}$, the latter two are not theorems in $\mathcal{SKL}$. Hence we expect that for usual mathematical theorems the proof effort in $\mathcal{SKL}$ will not be bigger then in $\mathcal{FOL}$.

4 Extensions – Sorted Unification

Even though the $\mathcal{TPF}$ calculus defined above represents a significant computational improvement over a naive tableau calculus for Kleene's strong logic for partial functions, it only makes very limited use of the sorts in $\mathcal{SKL}$. This can be improved by utilizing a rigid sorted unification algorithm that takes into account all the sort information present in the respective branch and uses it as a local sort signature. This measure in effect restricts the set of possible unifiers to those that are well-sorted with respect to this (local) sort signature. This allows to perform some of the reasoning about well-sortedness (and therefore definedness) in the unification in an algorithmic way. This reasoning would otherwise be triggered by the sort constraints in $\mathcal{SKL}$ and would have to be carried out in the proof search. The methods presented in this section are heavily influenced by Weidenbach's work on sorted tableau methods in [Wei94].

In the tableau framework the extension with sorted unification is simpler (but perhaps less powerful) than in the resolution framework (see for instance [Wei91, KK93]). The reason for this is the difference in the treatment of the disjunction in resolution and tableau. Tableau calculi use the β rule to analyze disjuncts in different branches, but pay the price with the necessity to instantiate the entire tableau. Consider, for example the formula $t{\leqslant}S \vee t{\leqslant}T$ stating that the term t has sort S or sort T. In the tableau method, we can investigate both situations in separate branches (with different local sets of declarations). In the resolution method, we have to use one of the disjuncts for sorted unification and residuate the other as a constraint, which has to be attached to well-sorted terms, well-sorted substitutions and clauses resulting from resolutions, whenever the other literal is used. On the other hand, the tableau method needs to instantiate all declarations that are used, since they can contain variables that also appear in other branches. Consider, for example, the axiom $\forall x_{|\mathbb{R}_\bullet}\ x > 0 \Rightarrow x{\leqslant}\mathbb{R}^*$, which can be read as a conditional declaration. This axiom will result in branches containing the literal $(x_{|\mathbb{R}} > 0)^{\mathbf{f}}$ and the declaration $(x_{|\mathbb{R}}{\leqslant}\mathbb{R}^*)^{\mathbf{t}}$. If we use the declaration in sorted unification to justify that $1{\leqslant}\mathbb{R}^*$, then we have to refute that $(1 > 0)^{\mathbf{f}}$ in the other branch. This simple example shows that we have to use (not surprisingly in a tableau framework) a *rigid* variant of sorted unification for our extension.

Definition 4.1 (Rigid Sorted Unification) Let $\mathcal{D}$ be a set of declarations, then we call a substitution σ *rigidly well-sorted* with respect to $\mathcal{D}$, iff there is a substitution τ, such that

1. $\sigma \subseteq \tau$ and $\mathbf{Dom}(\tau) \subseteq \mathbf{Free}(\mathcal{D}) \cup \mathbf{Dom}(\sigma)$
2. τ is well-sorted with respect to $\tau(\mathcal{D})$.

For instance the substitution $\sigma = [f(f(x_S))/z_S]$ is well-sorted, but not rigidly so, for the set $\mathcal{D} = \{f(y_S) \leqslant S\}$, since the declaration has to be used twice (in differing instances) to show that $f(f(x_S))$ has sort S. σ is, however, rigidly well-sorted with respect to $\mathcal{D}' = \{f(y_S) \leqslant S, f(v_S) \leqslant S\}$, and the substitution $\tau = [f(f(x_S))/z_S], f(v_s)/y_s]$ is a substitution that instantiates $\mathcal{D}'$ in the appropriate way.

4.1 Rigid Sorted Unification

Sorted unification with term declarations was first considered by Schmidt-Schauß who also presents a sound and complete algorithm in [SS89]. In $\mathcal{SKL}$, term declarations appear as sort atoms of the form $t \leqslant S$, declaring all instances of t to be of sort S. Rigid sorted unification is treated in [Wei94].

Definition 4.2 (Well-Sorted Terms) Let $\mathcal{D}$ be a set of *declarations* (positive sort literals of the form $(t \leqslant S)^{\mathbf{t}}$), then the set $\mathbf{wsT}_S(\mathcal{D})$ of *well-sorted terms of sort* S is inductively defined by

1. variables $x_S \in \mathbf{wsT}_S(\mathcal{D})$
2. if $t \leqslant T \in \mathcal{D}$ then $t \in \mathbf{wsT}_T(\mathcal{D})$
3. if $t \in \mathbf{wsT}_T(\mathcal{D})$ and $s \in \mathbf{wsT}_S(\mathcal{D})$ then $[s/x_S]t \in \mathbf{wsT}_T(\mathcal{D})$.

We call a substitution $[t^1/x^1_{S_1}], \ldots, [t^1/x^n_{S_n}]$ a *well-sorted substitution*, iff $t^i \in \mathbf{wsT}_{S_i}(\mathcal{D})$. Obviously the application of well-sorted substitutions to well-sorted terms yields well-sorted terms, so $\mathbf{wsT}(\mathcal{D})$ is closed under well-sorted substitutions and the set of well-sorted substitutions is a monoid with function composition.

Remark 4.3 The definition above is an inductive one, not in the structure of terms, but in the justification of the well-sortedness. A simple induction on this justification shows that the consequent $\mathcal{D} \models t \leqslant S$ is valid for any term $t \in \mathbf{wsT}_S(\mathcal{D})$. In particular, for any well-sorted term $t \in \mathbf{wsT}_S(\mathcal{D})$ the denotation $\mathcal{I}_\varphi(t)$ is in $\mathcal{A}_S$ and therefore defined.

Furthermore a declaration of the form $x_S \leqslant T \in \mathcal{D}$ entails that $\mathbf{wsT}_S(\mathcal{D}) \subseteq \mathbf{wsT}_T(\mathcal{D})$ and $\mathcal{A}_S \subseteq \mathcal{A}_T$ for any Σ-model $\mathcal{A}$ of $\mathcal{D}$. Therefore we call declarations of the form $x_S \leqslant T \in \mathcal{D}$ *subsort declarations*.

Since we are working in a tableau framework and our sorted unification algorithm involves nondeterminism, we utilize the tableau search mechanism for the search for unifiers by representing unification constraints as special disequality literals. This gives us a very uniform presentation of the combined tableau procedure

Definition 4.4 (Tableau Rules for Rigid Sorted Unification)
We assume the existence of a binary predicate symbol $\doteq \in \mathcal{P}^2$ and call a literal $(s \doteq t)^{\mathbf{fu}}$ a *constraint literal* and often abbreviate the $(s \doteq t)^{\mathbf{fu}}$ by $s \not\doteq t$, as usual we do not distinguish between $(s \doteq t)$ and $(t \doteq s)$. We model sorted unification

as a tableau-based constraint simplification calculus with the following set of inference rules: The *decomposition* rule

$$\frac{f(s_1,\ldots,s^n) \not\doteq f(t^1,\ldots,t^n)}{* \mid s^1 \not\doteq t^1 \mid \ldots \mid s^n \not\doteq t^n}$$

is just the traditional decomposition transformation, known from unification theory. Again note that we only need the disjunct $*$, if $n = 0$. The *subsort* rule

$$\frac{\begin{array}{c}(z_T \leqslant S)^{\mathbf{t}} \\ x_S \not\doteq y_T\end{array}}{*} \; [z_T/x_S], [z_T/y_T]$$

allows to eliminate variables, provided that T is a subsort of S, in which case the instantiation $[z_T/x_S]$ is well-sorted. The *intersect* rule

$$\frac{\begin{array}{c}(u_V \leqslant S)^{\mathbf{t}} \\ (v_V \leqslant T)^{\mathbf{t}} \\ x_S \not\doteq y_T\end{array}}{*} \; [u_V/x_S], [u_V/y_T], [u_V/v_V]$$

allows to eliminate a pair of variables that share a common subsort V. Finally a pair of variables can be eliminated for a term t, if t has sorts S and T, even if they do not share a common subsort (we call this situation *irregular*). Therefore the following rule *non-reg* instantiates the variables with the least committed generalization of t.

$$\frac{\begin{array}{c}(f(s_1,\ldots,s_n) \leqslant S)^{\mathbf{t}} \\ (f(t_1,\ldots,t_n) \leqslant T)^{\mathbf{t}} \\ x_S \not\doteq y_T\end{array}}{* \mid s^1 \not\doteq t^1 \mid \ldots \mid s^n \not\doteq t^n} \; [f(s^1,\ldots,s^n)/x_S], [f(t^1,\ldots,t^n)/y_T]$$

Finally we need a rule (the *imitation rule* below) that allows to eliminate a variable x_S for a term t, if it is an instance of a declaration in the branch above.

$$\frac{\begin{array}{c}(f(t^1,\ldots,t^n) \leqslant S)^{\mathbf{t}} \\ x_S \not\doteq f(s^1,\ldots,s^n)\end{array}}{* \mid s^1 \not\doteq t^1 \mid \ldots \mid s^n \not\doteq t^n} \; [f(t^1,\ldots,t^n)/x_S]$$

In contrast to the related set of rules for sorted unification in [Wei91] or [SS89] we only eliminate solved pairs that are known to be well-sorted from the set $\mathcal{D}$ of declarations. Therefore we do not need the explicit failure rules these authors need, since they do not test for well-sortedness of the pair before eliminating. In our system we define failure as irreducibility and non-solvedness, but we could also add explicit failure rules to detect failure early for a practical implementation.

We say that a declaration $(t \leqslant S)^{\mathbf{t}}$ is *used* by a unification inference rule, if it appears in the antecedent of the rule.

Theorem 4.5 *The above set of rules define a sound and complete non-determinist unification algorithm.*

Proof sketch: It is obvious that all inference rules maintain the property of well-sortedness for unification problems, since all new pairs added are from declarations and are therefore well-sorted by definition and the set of well-sorted terms is closed under well-sorted substitutions. Since the set of inference rules is a rigid variant of that given in [SS89, p.98], we refer to the proofs given there. These only have to be modified to account for rigidity. A close inspection of the differences shows that Schmidt-Schauß' rules can be obtained from ours by renaming all declarations that are used by the unification rules before applying the rules, and thus preventing that the declarations are used up in the process. For the proof of completeness, we construct a rigid extension τ from a non-rigid unifier by taking into account the instantiations of the declarations (in the rigid set of rules) that were circumvented in Schmidt-Schauß' rules by renaming. □

4.2 A Tableau Calculus for $\mathcal{SKL}$ using Rigid Sorted Unification

We will now extend $\mathcal{TPF}$ with a variant of the rigid sorted unification alogrithm above. Note that the notion of substitution discussed above is still not appropriate for a refutation calculus, where substitutions need to have ground instances. Otherwise the tableau cut rule becomes unsound: Let S be a sort that does not have ground terms, that is, where $\mathcal{A}_S$ may be empty, then a branch containing the literals $(Px_S)^{\mathbf{t}}$ and $(Py_S)^{\mathbf{f}}$ could be closed using the substitution $[y_S/x_S]$, without being unsatisfiable. A well-sorted term may not have ground instances, if it contains variables of sorts that do not have ground terms. Therefore we are interested in conditions for sorts to be non-empty.

Lemma 4.6 *Let $\mathcal{D}$ be a set of sort declarations, then the problem whether the set of ground terms of sort S is empty is decidable.*

Proof sketch: Let $Ax(\mathcal{D})$ be the set of propositional formulae $(S^1 \wedge \ldots \wedge S^n) \Rightarrow T$, such that $t{<}T \in \mathcal{D}$ and $\{x^1_S,\}$ are the free variables of t. Then the emptiness problem is equivalent to the problem whether $Ax(\mathcal{D}) \models S$ in propositional logic, which is known to be decidable. □

Remark 4.7 Thus we can modify the sorted unification algorithm above by allowing tableau closure $*$ (or equivalently the rule to be appliccable) only iff the sorts of the free variables in the substitutions associated with the rules are non-empty with respect to the set $\mathcal{D}$ of declarations in the branch above. This variant of the sorted unification algorithm only returns sorted unifiers that have well-sorted ground instances.

Now we will present an extension $\mathcal{TPF}(\Sigma)$ of $\mathcal{TPF}$ that allows to restrict the calculation to formulae that are well-sorted with respect to the declarations present on the branch above, and thereby prune branches of the proof search that would not lead to refutations, since they contain meaningless objects.

Definition 4.8 (Tableau with Sorted Unification ($\mathcal{TPF}(\Sigma)$))
To obtain the tableau calculus $\mathcal{TPF}(\Sigma)$ with sorted unification from $\mathcal{TPF}$, we modify the tableau closure rules and add the modified (cf. 4.7) constraint variant of the constraint simplification rules of sorted unification. The new *cut* and *strict* rules have for form

$$\frac{\begin{array}{c}A^{\alpha}\\ B^{\beta}\end{array}}{A \not\equiv B} \qquad \frac{C^{\gamma}}{\frac{(t{\ll}\mathfrak{D})^{\mathsf{f}}}{s \not\equiv t}}$$

where $\alpha \cap \beta = \emptyset$, C has a subterm s, and $\gamma \subseteq \{\mathsf{f}, \mathsf{t}\}$. Thus instead of using unsorted unification, these rules residuate a unification constraint can then be processed by the sorted unification algorithm. All other $\mathcal{TPF}$ rules stay unchanged.

Theorem 4.9 *$\mathcal{TPF}(\Sigma)$ is sound and refutation complete.*

Proof sketch: The soundness of $\mathcal{TPF}(\Sigma)$ relies on the soundness of the sorted unification algorithm, which guarantees only well-sorted instantiations. For the completeness proof we can again use the model existence theorem 2.26, where we only have to reconsider the tableau lifting theorem for $\mathcal{TPF}(\Sigma)$. This can be proven with the standard argumentation, since the sorted unification algorithm is complete and we can abstract from the internal structure of the unification derivation. □

As we have seen in remark 4.3 the well-sorted substitutions and therefore well-sorted unifications filter out instantiations of the tableau that contain meaningless objects and therefore cannot contribute to a refutation of the initial consequent. This property yields a significant pruning of the search spaces and therefore in a gain of computational efficiency. However the rigidity of the unification algorithm makes it necessary to guess in advance the number of instances of the declarations needed for a proof, since they are used up during the unification. This is especially bothersome, since in general a great multiplicity of declarations is needed. In order to arrive at a more practical algorithm it will be important to find variants of the unification algorithm that are rigid only on the disjunctive part of the declarations present in a consequent.

Example 4.10 (continuing 3.7) Now we revisit the problem of proving

$$\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}, \mathrm{A4}, \mathrm{A5}\} \models \mathrm{T}$$

using the tableau calculus with sorted unification. While the first two main parts of the proof in figure 1, namely the problem setting and the initial simplification remain the same, the proper refutation will be shorter, in particular only three branches instead of five have to be considered. In figure 3, we display only the last part.

The unification for closing F2 with T3′ is straightforward because of T1 and T2, $(c{\ll}\mathbb{R})^{\mathsf{t}}$ and $(d{\ll}\mathbb{R})^{\mathsf{t}}$, while the sorted unification for closing F3 and F5 makes use of T1, T2, and the term declaration A4′. For the closure of T3″ and A3′ the unification algorithm must derive that $\frac{1}{c-d}$ has the sort $\mathbb{R}^*$, this is done by F6 with the term declaration A2′.

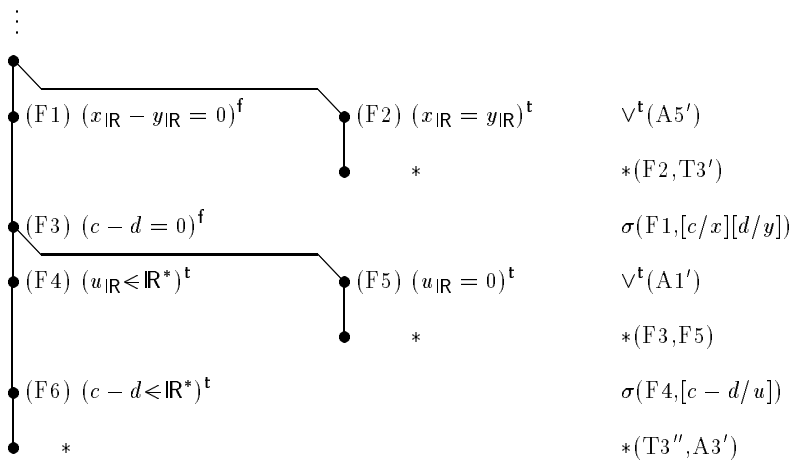


Fig. 3. Tableau proof with sorted unification

5 Conclusion

We have developed a sorted three-valued logic for the formalization of informal mathematical reasoning with partial functions. This system generalizes the system proposed by Kleene in [Kle52] for the treatment of partial functions over natural numbers to general first-order logic. In fact we believe that the unsorted version of our system without the ! operator is a faithful formalization of Kleene's ideas.

If we compare $\mathcal{SKL}$ to the three other approaches mentioned in the introduction, we see that the truth conditions coincide on valid mathematical statements, but that $\mathcal{SKL}$ properly excludes statements that a mathematician would reject as having problems with definedness. While the first approach has not the necessary expressiveness, the second and fourth approaches legitimate unwanted statements as theorems.

We have presented a sound and complete tableau calculus with dynamic sorts for our logic $\mathcal{SKL}$, which uses the sort mechanism to capture the fact that in Kleene's logic quantification only ranges over defined individuals. Our calculus can be seen as an extension of classical logic that combines methods from many-valued logics (cf. [BF92, Häh92]) for a correct treatment of the undefined and sorted logics (see [Wei89, Wei91]) for an adequate treatment of the defined. It differs from the sequent calculus in [LCGF89] in that the use of dynamic sort techniques greatly simplifies the calculus, since most definedness preconditions can be taken care of in the unification. Thus we believe that our system is not only more faithful to Kleene's ideas (definedness inference is handled in the unification at a level below the calculus) but also more efficient for the sort techniques involved.

In an earlier work [KK93, KK94] we had represented a resolution calculus of strong sorted Kleene logic. In this work, we not only have transferred the methods developed there to the tableau framework, but have also shown that normally proofs that keep track of the definedness conditions are not more complex than those in the classical two-valued logic. In some sense it is surprising that in spite of the advantages mentioned above, the complexity of proof search can be preserved by the treatment of multi-indices.

Of course further extensions of the system described here have to be considered in order to be feasible for practical mathematics. In particular this calculus does not address the question of the mechanization of higher-order features for the formalization of mathematical practice. Higher-order logics are especially suitable for formalizing partial functions, since functions are first class objects of the systems, that can even be quantified over. In this direction the work of Farmer et al. [Far90, FGT93] has shown that partial functions are a very natural and powerful tool for formalizing mathematics. We expect that our three-valued approach, which remedies some problems of their simpler two-valued approach (see the discussion in the introduction and in example 2.11) can be generalized in much the same manner and will be a useful tool for formalizing mathematics.

Finally, the authors believe that the merit of the idea of generalizing first-order logic with respect to both, the number of truth values, and the domain of quantification is not confined to the application to partial functions. In particular there seem to be no obstacles against the extension of many multi-valued logics in artificial intelligence (such as Belnap's four-valued paraconsistent logic) that have only been investigated for the propositional fragment to the first-order case using our techniques.

References

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Academic Press, 1986.

[Bee85] Michael J. Beeson. *Foundations of Constructive Mathematics.* Springer Verlag, 1985.

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

[BF92] Matthias Baaz and Christian G. Fermüller. Resolution for many-valued logics. In A. Voronkov, editor, *Proceedings of LPAR*, pages 107–118, St. Petersburg, Russia, 1992. Springer Verlag, LNAI 624.

[BFZ93] Matthias Baaz, Christian G. Fermüller, and Richard Zach. Dual systems of sequents and tableaux for many-valued logics. Technical Report TUW-E185.2BFZ.2-92, Technische Universität Wien, 1993.

[Car87] Walter A. Carnielli. Systematization of finite many-valued logics through the method of tableaux. *The Journal of Symbolic Logic*, 52:473–493, 1987.

[Car91] Walter A. Carnielli. On sequents and tableaux for many-valued logics. *Journal of Non-Classical Logic*, 8(1):59–76, 1991.

[Far90] William M. Farmer. A partial functions version of Church's simple theory of types. *The Journal of Symbolic Logic*, 55(3):1269–1291, 1990.

[FGT93] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11(2):213–248, October 1993.

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

[Häh92] Reiner Hähnle. *Automated Theorem Proving in Multiple Valued Logics*. PhD thesis, Fachbereich Informatik, Universität Karlsruhe, Karlsruhe, Germany, March 1992. revised version: Automated Deduction in Multiple-Valued Logics, Oxford University Press, 1994.

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

[HS94] Reiner Hähnle and Peter H. Schmitt. The liberalized δ-rule in free variable tableaux. *Journal of Automated Reasoning*, 12(2):211–222, 1994.

[KK93] Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. SEKI Report SR-93-20, Fachbereich Informatik, Universität des Saarlandes, Im Stadtwald, Saarbrücken, Germany, 1993.

[KK94] Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. In Alan Bundy, editor, *Proceedings of the 12th CADE*, pages 371–385, Nancy, France, 1994. Springer Verlag, LNAI 814.

[Kle52] Stephen C. Kleene. *Introduction to Metamathematics*. Van Nostrand, 1952.

[LCGF89] Francisca Lucio-Carrrasco and Antonio Gavilanes-Franco. A first order logic for partial functions. In *Proceedings STACS'89*, pages 47–58. Springer Verlag, LNCS 349, 1989.

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

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

[Sch68] R. Schock. *Logics without Existence Assumptions*. Almquist & Wisell, 1968.

[Sco70] Dana S. Scott. Outline of a mathematical theory of computation. Technical Monograph PRG-2, Oxford University Computing Laboratory, 1970.

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

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

[SS89] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, Springer Verlag, LNAI 395, 1989.

[Tic82] Pawel Tichy. Foundations of partial type theory. *Reports on Mathematical Logic*, 14:59–72, 1982.

[Wei89] Christoph Weidenbach. A resolution calculus with dynamic sort structures and partial functions. SEKI Report SR-89-23, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1989. Short version in ECAI'90, p. 688–693.

[Wei91] Christoph Weidenbach. A sorted logic using dynamic sorts. Technical Report MPI-I-91-218, Max-Planck-Institut für Informatik, Im Stadtwald, Saarbrücken, Germany, 1991. Short version in IJCAI'93, p. 60–65.

[Wei94] Christoph Weidenbach. First-order tableaux with sorts. In Krysia Broda and Marcello D'Agostino et.al., editors, *TABLEAUX-'94, 3rd Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 247–261. Imperial College of Science Technology and Medicine, TR-94/5, April 1994. To appear in the Bulletin of the IGPL.