UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

SEKI-Report

A Mechanization of Strong Kleene Logic for Partial Functions

Manfred Kerber and Michael Kohlhase

SEKI Report SR-93-20 (SFB)

A Mechanization of Strong Kleene Logic 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 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 resolution calculus that combines the proper treatment of partial functions with the efficiency of sorted calculi.

Keywords: Partial functions, many-valued logic, order-sorted logic, resolution.

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

Contents

1 Introduction **3**

2 Strong Order-Sorted Kleene Logic ($\mathcal{SKL}$) **4**
Syntax 5
Semantics 5
Extended Example 8
Relativization into Truth-Functional Logic 8
Extended Example (continued) 12

3 Simple Resolution **12**
Clause Normal Form 13
Resolution Calculus ($\mathcal{RPF}$) 14
Extended Example (continued) 16
Soundness and Completeness 17

4 Resolution with Order-Sorted Unification **20**
Unification 20
Resolution ($\mathcal{RPF}(\mathcal{D})$) 22
Extended Example (continued) 24

5 Conclusion **25**

1 Introduction

Many practical applications of deduction systems in mathematics and computer science rely on the proper treatment of partial functions. Although there are workarounds for most concrete situations, there has been a considerable interest in the community for clean formalizations of partial functions.

One of the key problems to be solved when formalizing partial functions is to decide, what happens if partial functions are applied to arguments not in their domain. In mathematical practice expressions like $\frac{0}{0} = 1$ or $odd(predecessor(0))$ are thought to be neither true nor false. This phenomenon can be handled in the well-known systems for intuitionistic logic, where the law of the excluded middle does not hold, hence $\frac{0}{0} = 1$ can be (and in fact is) neither true nor false, since neither the truth nor the falsehood of this expression can be shown. However, most mathematicians do not want to give up the law of the excluded middle, because it is basic for a strong proof technique, the indirect proof[1]. Another standard way to deal with this situation is to consider expressions like $\frac{0}{0}$ as "meaningless". 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 [6, 7], Hähnle [11], Baaz and Fermüller [2] impossible. Kleene's approach has been utilized by Tichy [19], Lucio-Carrasco and Gavilanes-Franco [14] to give logical systems for partial functions. Both approaches offer unsorted operationalizations of the systems in sequent calculi.

Other authors (cf. [5, 8, 17, 20]) have avoided the problems that accompany treating a third truth value, and simply consider all atomic expressions containing a meaningless term 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 $y \neq 0$ and $z = \frac{x}{0}$ obtains the truth value f which in turn makes the implication true. However, it is mathematical consensus that the equation should only hold provided that y is not 0. It will turn

[1] For example, consider the following problem: are there irrational numbers a and b such that a^b is rational. With the law of the excluded middle this can be easily shown. If $\sqrt{2}^{\sqrt{2}}$ is rational, $a = b := \sqrt{2}$ solve the problem, since $\sqrt{2}$ is irrational. Is, however, $\sqrt{2}^{\sqrt{2}}$ irrational, then $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$ solve the problem, since $a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = 2$ is rational. You don't have to know whether $\sqrt{2}^{\sqrt{2}}$ is rational or not and indeed this question is not easily answered. (Compare [13, p.160])

out (cf. example 3.11) that the formula is not a theorem in our formalization, since the case $y = 0$ is a counterexample.

We formalize Kleene's ideas for partial functions in an order-sorted three-valued logic, called $\mathcal{SKL}$, that uses the Kleene's strong interpretation of connectives and quantifiers and adapts techniques from Weidenbach's logic [20] to handle definedness information. We furthermore present two versions $\mathcal{RPF}$ and $\mathcal{RPF}(\mathcal{D})$ of a resolution calculus for partial functions.

We would like to thank Christian Fermüller and Ortwin Scheja for stimulating discussions.

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

In [12] 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 2.7. 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 we declare all functions and predicates to be 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 [12]. 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 [20, 21] give us the bounded quantification). We will call formulations of $\mathcal{SKL}$ where $\mathfrak{D}$ is the only sort symbol in the signature *strong unsorted Kleene logic.* The further use of sorts gives the well-known advantages of sorted logics for the conciseness of representation and reduction of search spaces.

Syntax

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 *sort symbols* 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 symbol.

Definition 2.2 (Well-formed Terms and Formulae) We define the set of *well-formed terms* to be the set of variables together with $f(t^1, \ldots, t^k)$ for well-formed 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 well-formed atom*. If t is a term and S a sort then $t \lessdot S$ is a *well-formed sort atom*. The set of *well-formed formulae* contains all well-formed atoms and with formulae A and B the formulae $A \wedge B$, $\neg A$, $!A$, and $\forall x_S.\, A$. Here the intended meaning of the classical connectives is the usual, whereas the intended meaning of $!A$ is that A is defined.

Semantics

In this section we will define the three valued semantics for $\mathcal{SKL}$ by extending the universe of discourse with $\perp$ for the undefined. Note that this is similar to the classical flat CPO construction [18], but Kleene's interpretation of truth values does not make u minimal. Since we are not interested in least fix-points, monotonicity does not play a role in this paper.

Definition 2.3 (Partial Σ-Algebra) Let Σ be a signature, then a *partial Σ-algebra* consists of a

1. non-empty carrier set $\mathcal{A}$,
2. an *interpretation function* $\begin{aligned} \mathcal{I} &: \mathcal{F}^k \longrightarrow \mathcal{F}_p(\mathcal{A}^k; \mathcal{A}) \\ \mathcal{I} &: \mathcal{P}^k \longrightarrow \mathcal{F}_p(\mathcal{A}^k; \{\mathsf{f}, \mathsf{t}\}) \\ \mathcal{I} &: \mathcal{S}^* \longrightarrow \mathcal{F}(\mathcal{A}; \{\mathsf{f}, \mathsf{t}\})^2. \end{aligned}$

   where $\mathcal{F}_p(A; B)$ is the set of partial functions form A into B, and $\mathcal{F}(A; B)$ is

[2] For defined individuals the membership to a sort is not undefined.

that of total functions. Partial functions are defined as right-unique relations. 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. This fact will require special treatments in the transformation to clause normal form and for instantiations in the resolution calculus.

The partial Σ-algebra is an algebraic account of the standard interpretation in mathematics, where partiality of functions is directly modelled by right-unique relations. To be able to use standard methods from predicate logics, we close the universe with a bottom element $\bot$ and model partial functions as strict total function. Obviously these notions of algebras have a one-to-one correspondence, so both approaches are equivalent.

Definition 2.4 (Strict Σ-Algebra) Let Σ be a signature and $(\mathcal{A},\mathcal{I})$ a partial Σ-algebra then we obtain the *strict* Σ*-algebra* $(\mathcal{A}^\bot,\mathcal{I}^\bot)$ *for* $(\mathcal{A},\mathcal{I})$ by the following extensions

1. $\mathcal{A}^\bot := \mathcal{A} \cup \{\bot\}$, where we assume that $\bot$ is not already a member of $\mathcal{A}$
2. The interpretation function $\mathcal{I}^\bot$ is defined to be
   (a) $\mathcal{I}^\bot(f) := [\mathcal{I}(f)]^\bot$, where $h^\bot$ is the strict extension of a function, that is, $h^\bot(a_1,\ldots,a_k) = h(a_1,\ldots,a_k)$, if $(a_1,\ldots,a_k) \in \mathbf{Dom}(h)$ and $h^\bot(a_1,\ldots,a_k) = \bot$ otherwise.
   (b) $\mathcal{I}^\bot(P) := [\mathcal{I}(P)]^\bot$, where $Q^\bot$ is the strict extension of a predicate, that is, $Q^\bot(a_1,\ldots,a_k) = Q(a_1,\ldots,a_k)$, if $(a_1,\ldots,a_k) \in \mathbf{Dom}(Q)$ and $Q^\bot(a_1,\ldots,a_k) = \mathsf{u}$ otherwise.
   (c) $\mathcal{I}$ is extended to $\mathcal{I}^\bot$ for sorts in $\mathcal{S}^*$ just as in the predicate case.
   (d) $\mathcal{I}^\bot(\mathfrak{D})(a) := \mathsf{t}$, if $a \in \mathcal{A}$ and $\mathcal{I}^\bot(\mathfrak{D})(\bot) := \mathsf{f}$.

Since strict Σ-algebras are the intended semantics of $\mathcal{SKL}$, we will often drop the explicit reference to $\bot$ in our notation. Note that $\bot \notin \mathcal{A}_S$ for any $S \in \mathcal{S}$.

Definition 2.5 (Σ-assignment) Let $(\mathcal{A},\mathcal{I})$ be a strict Σ-algebra, then we call a total mapping $\varphi: \mathcal{V} \longrightarrow \mathcal{A}^\bot$ 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 2.6 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 well-formed 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. $\mathcal{I}_\varphi(f(t^1,\ldots,t^k) := \mathcal{I}(f)(\mathcal{I}_\varphi(t^1),\ldots,\mathcal{I}_\varphi(t^k))$

4. $\mathcal{I}_\varphi(t{\Leftarrow}S) = \mathcal{I}(S)(\mathcal{I}_\varphi(t))$

Note that this definition applies to $\mathcal{P}$ and $\mathcal{F}$ alike, thus we have given the semantics of all atomic formulae. The semantic status of sorts is that of total unary predicates; in particular in $\mathcal{A}^\perp$ we have $\mathcal{I}_\varphi(t{\Leftarrow}S) = \mathsf{u}$, iff $\mathcal{I}_\varphi(t) = \perp$.

Definition 2.7 The semantics of composed formulae is obtained from the values of the atomic subformulae 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

$\neg$	
f	t
u	u
t	f

!	
f	t
u	f
t	t

Kleene does not use the ! operator as a connective but treats it on the meta-level. Note 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 negation and conjunction, ! is normal.

The semantics of the universal quantifier is defined with the help of a function $\tilde{\forall}$ from the non-empty subsets of the truth values in the truth values. We define

$$\mathcal{I}_\varphi(\forall x_S.\, A) := \tilde{\forall}(\{\mathcal{I}_{\varphi,[a/x]}(A) \mid a \in \mathcal{A}_S\}) \quad \text{with} \quad \tilde{\forall}(T) := \begin{cases} \mathsf{t} & \text{for } T = \{\mathsf{t}\} \\ \mathsf{u} & \text{for } T = \{\mathsf{t},\mathsf{u}\} \text{ or } \{\mathsf{u}\} \\ \mathsf{f} & \mathsf{f} \in T \end{cases}$$

Note that with this definition quantification is separated into a truth functional part $\tilde{\forall}$ and an instantiation part that only considers members of $\mathcal{A}_S$.

Using the classical definitions the other connectives and the existential quantifier can be defined in terms of $\neg$, $\wedge$, and $\forall$, e.g., $A \vee B := \neg(\neg A \wedge \neg B)$.

Definition 2.8 (Σ-Model) Let A be a well-formed 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 φ. With this notion we can define the notions of ***validity***, *(un)-satisfiability*, and *entailment* in the usual way.

Remark 2.9 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 resolution calculus.

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 formula $A := (\mathrm{A1} \wedge \mathrm{A2} \wedge \mathrm{A3} \wedge \mathrm{A4} \wedge \mathrm{A5}) \Rightarrow \mathrm{T}$ with

$$\mathrm{A1}\ \forall x_{\mathbb{R}}\ x \neq 0 \Rightarrow x < \mathbb{R}^*$$

$$\mathrm{A2}\ \forall x_{\mathbb{R}^*}\ \tfrac{1}{x} < \mathbb{R}^*$$

$$\mathrm{A3}\ \forall x_{\mathbb{R}^*}\ x^2 > 0$$

$$\mathrm{A4}\ \forall x_{\mathbb{R}}\ \forall y_{\mathbb{R}}\ x - y < \mathbb{R}$$

$$\mathrm{A5}\ \forall x_{\mathbb{R}}\ \forall y_{\mathbb{R}}\ x - y = 0 \Rightarrow x = y$$

$$\mathrm{T}\ \forall x_{\mathbb{R}}\ \forall y_{\mathbb{R}}\ x \neq y \Rightarrow \left(\tfrac{1}{x-y}\right)^2 > 0$$

An informal mathematical argumentation why T is entailed by $\mathrm{A1} \wedge \ldots \wedge \mathrm{A5}$ can be as follows:

Let x and y be arbitrary elements of $\mathbb{R}$. If $x = y$, the premise of T is wrong, 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.

However, if we analyze the justification of this argumentation, we see that there is a hidden assumption, namely 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. There are two solutions of this problem, namely adding further formulae Ai, in which the definiteness of the predicates are specified, or – what is normally done in mathematics – to start with a formula where the Ai are assumed to be true, that is neither false nor undefined. We will discuss the alternatives later, when we give a formal proof for the example.

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 an order-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 [2, 6, 7, 11]. 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 2.7, the semantics of the quantifier uses unrestricted instantiation, that is,

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

Definition 2.10 We define transformations $\mathbf{Rel}^{\mathcal{S}}$ and $\mathbf{Rel}^{\mathfrak{D}}$, that map $\mathcal{SKL}$-sentences to unsorted $\mathcal{SKL}$-sentences and further into $\mathbf{K}^3$-sentences. $\mathbf{Rel}^{\mathcal{S}}$ is the identity on terms and atoms and homomorphic on connectives and

$$\mathbf{Rel}^{\mathcal{S}}(\forall x_S.\, \Phi) := \forall x_{\mathfrak{D}}.\, x{\leqslant}S \Rightarrow \mathbf{Rel}^{\mathcal{S}}(\Phi)$$

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: $\forall x_{\mathfrak{D}}.\, !S(x)$. The set of all these axioms is denoted by $\mathbf{Rel}^{\mathcal{S}}(\Sigma)$.

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

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

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

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

for any predicate symbol $P \in \mathcal{P}^n$, such that $P \neq \mathfrak{D}$ and for any function symbol $f \in \mathcal{F}^n$, together with the axioms

$$\forall x.\, \mathfrak{D}(x) \vee \neg\mathfrak{D}(x) \qquad \text{and} \qquad \exists x.\, \mathfrak{D}(x)$$

These axioms axiomatize the $\mathcal{SKL}$ notion of definedness in $\mathbf{K}^3$. In particular the last axioms state that the predicate $\mathfrak{D}$ is two-valued and non-empty, in contrast to all other sort predicates which are strict and thus three-valued and may be empty. The other axioms force all functions and predicates to be interpreted strictly with respect to the $\mathfrak{D}$ predicate.

Note that in the case of nullary function symbols (constants) the signature axioms have the form $\mathfrak{D}(c^o)$.

Theorem 2.11 (Sort Theorem) *Let Φ be a set of sentences, then the following are equivalent*

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

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 $\mathbf{Rel}^{\mathfrak{D}}(\Phi)$. 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 $\mathcal{M}^3 \models^{\mathbf{K}^3} \mathbf{Rel}^{\mathfrak{D}}(\Sigma)$, since $\mathcal{M}$ is a Σ-model, where all functions are strict and the carrier $\mathcal{A} = \mathrm{Im}(\mathcal{I}^3(\mathfrak{D}))$ is nonempty.

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

$$\mathcal{I}^3_\varphi(\mathbf{Rel}^{\mathfrak{D}}(t{\leftarrow}\mathfrak{D})) = \mathcal{I}^3_\varphi(\mathfrak{D}(t)) = \mathcal{I}^3(\mathfrak{D})(\mathcal{I}^3_\varphi(t)) = \mathcal{I}(\mathfrak{D})(\mathcal{I}_\varphi(t)) = \mathcal{I}_\varphi(t{\leftarrow}\mathfrak{D})$$

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

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

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

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

Now $\mathcal{I}^3_\varphi(\mathbf{Rel}^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathcal{I}^3_\varphi(\forall x.\, \mathfrak{D}(X) \Rightarrow \mathbf{Rel}^{\mathfrak{D}}(\Psi))$
$= \tilde{\forall}(\{\mathcal{I}^3(_{\varphi,[a/x]}(\mathfrak{D}(X) \Rightarrow \mathbf{Rel}^{\mathfrak{D}}(\Psi))) \mid a \in \mathcal{A}^3\})$,
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 $\mathcal{I}^3_\psi(\mathfrak{D}(x) \Rightarrow \mathbf{Rel}^{\mathfrak{D}}(\Psi)) = \mathsf{t}$. If $a \neq \bot$, then by inductive hypothesis $\mathcal{I}^3_\varphi(\mathbf{Rel}^{\mathfrak{D}}(\Psi)) = \mathcal{I}_\varphi(\Psi)$ and therefore $\Theta^3 = \Theta \cup \{\mathsf{t}\}$.

$$\begin{array}{llll}
\mathcal{I}^3_\varphi(\mathbf{Rel}^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathsf{t} & \text{iff} & \Theta^3 = \Theta = \{\mathsf{t}\} & \text{iff} \quad \mathcal{I}_\varphi(\forall x_{\mathfrak{D}}.\, \Psi) = \mathsf{t} \\
\mathcal{I}^3_\varphi(\mathbf{Rel}^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathsf{u} & \text{iff} & \Theta^3 = \Theta = \{\mathsf{u}, \mathsf{t}\} \text{ or } \{\mathsf{u}\} & \text{iff} \quad \mathcal{I}_\varphi(\forall x_{\mathfrak{D}}.\, \Psi) = \mathsf{u} \\
\mathcal{I}^3_\varphi(\mathbf{Rel}^{\mathfrak{D}}(\forall x_{\mathfrak{D}}.\, \Psi)) = \mathsf{f} & \text{iff} & \mathsf{f} \in \Theta^3 = \Theta \cup \{\mathsf{t}\} & \text{iff} \quad \mathcal{I}_\varphi(\forall x_{\mathfrak{D}}.\, \Psi) = \mathsf{f}
\end{array}$$

Since $\mathbf{Rel}^{\mathfrak{D}}$ is homomorphic for connectives, we have completed the induction, thus $\mathcal{M}^3 \models^{\mathbf{K}^3} \mathbf{Rel}^{\mathfrak{D}}(\Phi)$ 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 $\mathcal{M}^3 \models \mathbf{Rel}^{\mathfrak{D}}(\Phi) \cup \mathbf{Rel}^{\mathfrak{D}}(\Sigma)$. 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 $\forall x.\ \mathfrak{D}(x) \vee \neg\mathfrak{D}(x) \in \mathbf{Rel}^{\mathfrak{D}}(\Sigma)$ and $\mathcal{A} \neq \emptyset$ as required in the definition, since $\exists x.\ \mathfrak{D}(x) \in \mathbf{Rel}^{\mathfrak{D}}(\Sigma)$. If $\mathcal{A}_\perp = \emptyset$ it is easy to see that $(\mathcal{A}^3, \mathcal{I}^3)$ is already a partial Σ-algebra and the assertion is trivial for the corresponding strict Σ-algebra. 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 $\mathcal{M}^3 \models \mathbf{Rel}^{\mathfrak{D}}(\Sigma)$, 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 $\mathcal{M}^3 \models \mathbf{Rel}^{\mathfrak{D}}(\Sigma)$ and gives us $\mathcal{I}_{\pi\circ\varphi}(A) = \mathcal{I}^3(\mathbf{Rel}^{\mathfrak{D}}(A))$ for all atoms A. From this we obtain the general result $\mathcal{I}_{\pi\circ\varphi}(\Phi) = \mathcal{I}^3_\varphi(\mathbf{Rel}^{\mathfrak{D}}(\Phi))$ 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 $\mathcal{M}^3 \models \mathbf{Rel}^{\mathfrak{D}}(\Phi)$. □

As a consequence of the sort theorem, the standard operationalization for many-valued logics [2, 6, 7, 11] can be utilized to mechanize strong order-sorted Kleene logic and in fact the system of Lucio-Carrasco and Gavilanes-Franco [14] can be seen as a standard many-valued tableau operationalization [11, 3] 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 more concise than the relativized version and as we will see the theory of definedness is treated goal-driven by the $\mathcal{RPF}$ calculus (cf. section 3). Thus the $\mathcal{RPF}$ calculus is closer to informal practice than the relativization in this respect.

Extended Example (continued)

The relativization $\mathbf{Rel}^{\mathcal{S}}(\mathbf{Rel}^{\mathfrak{D}}(A))$ of the formula A in the extended example is the $\mathbf{K}^3$-formula (R1 $\wedge$ R2 $\wedge$ R3 $\wedge$ R4 $\wedge$ R5) $\Rightarrow$ RT.

R1 $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathbb{R}(x) \Rightarrow (x \neq 0 \Rightarrow \mathbb{R}^*(x)))$

R2 $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathbb{R}^*(x) \Rightarrow \mathbb{R}^*(\frac{1}{x}))$

R3 $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathbb{R}^*(x) \Rightarrow x^2 > 0)$

R4 $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathbb{R}(x) \Rightarrow (\forall y.\, \mathfrak{D}(y) \wedge \mathbb{R}(y) \Rightarrow \mathbb{R}(x - y)))$

R5 $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathbb{R}(x) \Rightarrow (\forall y.\, \mathfrak{D}(y) \Rightarrow (\mathbb{R}(y) \Rightarrow (x - y = 0 \Rightarrow x = y))))$

RT $\forall x.\, \mathfrak{D}(x) \Rightarrow (\mathbb{R}(x) \Rightarrow (\forall y.\, \mathfrak{D}(y) \Rightarrow (\mathbb{R}(y) \Rightarrow (x \neq 0 \Rightarrow \left(\frac{1}{x}\right)^2 > 0))))$

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

$\mathrm{R}^{=}$ $\forall x, y.\, (x = y \vee x \neq y) \Rightarrow \mathfrak{D}(x) \wedge \mathfrak{D}(y)$

$\mathrm{R}^{>}$ $\forall x, y.\, (x > y \vee x \not> y) \Rightarrow \mathfrak{D}(x) \wedge \mathfrak{D}(y)$

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^{0} $\mathfrak{D}(0)$

R^{2} $\forall x.\, \mathfrak{D}(x^2) \Rightarrow \mathfrak{D}(x)$

$\mathfrak{D}^{!}$ $\forall x.\, \mathfrak{D}(x) \vee \neg\mathfrak{D}(x)$

$\mathfrak{D}^{\emptyset}$ $\exists x.\, \mathfrak{D}(x)$

3 Simple Resolution

In this section we present a resolution calculus with dynamic sorts that is a generalization of Weidenbach's work [20, 21] with ideas from [2, 11]. The concept of dynamic sorts is essential to our program, since definedness cannot in general be decided by syntactic means only, but is usually given in the form of logical axioms

that have to be reasoned about in the calculus itself. Thus static sort methods like those in [16, 10] are not sufficient for our purposes.

Clause Normal Form

Definition 3.1 Let A be a well-formed formula, then we call A^α (the formula A indexed with the intended truth value $\alpha \in \{\mathsf{f}, \mathsf{u}, \mathsf{t}\}$), a *labelled formula.* We will call a labelled atom A^α a *literal* and a set of literals $\{A_1^{\alpha_1}, \ldots, A_n^{\alpha_n}\}$ a *clause.* We say that a Σ-model $\mathcal{M}$ satisfies a clause C, iff it satisfies one of its literals L^α, that is, $\mathcal{I}_\varphi(L^\alpha) = \alpha$. $\mathcal{M}$ satisfies a set of clauses iff it satisfies each clause. In order to conserve space, we employ the "," as the operator for the disjoint union of sets, so that C, L^α means $C \cup \{L^\alpha\}$, in particular L^α is not a member of C. Furthermore we adopt Hähnle's notion of multi-labels in the form $C, A^{\alpha\beta}$ to mean C, A^α, A^β.

Now we are in the position to give a set of transformations that take a set of labelled formulae to an equivalent set of clauses.

Definition 3.2 (Transformations for Clause Normal Form)

$$\frac{C, (A \wedge B)^{\mathsf{t}}}{C, A^{\mathsf{t}} \quad C, B^{\mathsf{t}}} \qquad \frac{C, (A \wedge B)^{\mathsf{u}}}{C, A^{\mathsf{ut}} \quad C, B^{\mathsf{ut}} \quad C, A^{\mathsf{u}}, B^{\mathsf{u}}} \qquad \frac{C, (A \wedge B)^{\mathsf{f}}}{C, A^{\mathsf{f}}, B^{\mathsf{f}}}$$

$$\frac{C, (\neg A)^{\mathsf{t}}}{C, A^{\mathsf{f}}} \qquad \frac{C, (\neg A)^{\mathsf{u}}}{C, A^{\mathsf{u}}} \qquad \frac{C, (\neg A)^{\mathsf{f}}}{C, A^{\mathsf{t}}}$$

$$\frac{C, (\forall x_S.\, A[x_S])^{\mathsf{t}}}{C, A[x_S]^{\mathsf{t}}} \qquad \frac{C, (\forall x_S.\, A[x_S])^{\mathsf{u}}}{C, A[f(y^1, \ldots, y^n)]^{\mathsf{u}} \quad C, A[x_S]^{\mathsf{ut}} \quad C, (f(y^1, \ldots, y^n) \lessdot S)^{\mathsf{t}}}$$

$$\frac{C, (\forall x_S.\, A[x_S])^{\mathsf{f}}}{C, A[f(y^1, \ldots, y^n)]^{\mathsf{f}} \quad C, (f(y^1, \ldots, y^n) \lessdot S)^{\mathsf{t}}}$$

$$\frac{C, (!A)^{\mathsf{t}}}{C, A^{\mathsf{tf}}} \qquad \frac{C, (!A)^{\mathsf{u}}}{C} \qquad \frac{C, (!A)^{\mathsf{f}}}{C, A^{\mathsf{u}}}$$

$$\frac{C, (t \lessdot \mathfrak{D})^{\mathsf{u}}}{C} \qquad \frac{C, (t \lessdot S)^{\mathsf{u}}}{C, (t \lessdot \mathfrak{D})^{\mathsf{f}}}$$

where $\{x_S, y^1, \ldots, y^n\} = \mathbf{Free}(A)$ and f is a new function symbol of arity n. Here $\mathbf{Free}(A)$ denotes the set of free variables of A.

For any set Φ of well-formed labelled sentences we will denote the set of clauses resulting from a total reduction of Φ by the above transformations with $\mathbf{CNF}(\Phi)$.

General Assumption 3.3 The clause normal form transformations as presented above are not complete, *i.e.* they do not transform every given labelled formula into clause form, since the rules for quantified formulae insist that the bound variable occurs in the scope. In fact the handling of degenerate quantifications poses some problems in the presence of possibly empty sorts, as quantification over empty sets are vacuously true. In this situation we have three possibilities, either to forbid degenerate quantifications, or empty sorts, or treat degenerate quantifications in the clause normal form transforamtions. For this paper we chose the first, since degenerate quantifications do not make much sense mathematically and do not appear in informal mathematics. Thus we will asssume that in all formulae in this paper the bound variables of quantifications occur in the scopes.

Remark 3.4 For treating degenerate quantifications in the clause normal form transformation we have to add the rules

$$\frac{C, (\forall x_S.\ A)^{\mathsf{t}}}{C, A^{\mathsf{t}}, (x \leqslant S)^{\mathsf{f}}} \qquad \frac{C, (\forall x_S.\ A)^{\mathsf{f}}}{C, (c \leqslant S)^{\mathsf{t}} \quad C, A^{\mathsf{f}}} \qquad \frac{C, (\forall x_S.\ A)^{\mathsf{u}}}{C, (c \leqslant S)^{\mathsf{t}} \quad C, A^{\mathsf{u}}}$$

In the context of mathematics it is often natural to assume the sorts to be non-empty. In this case the the quantifier rules can be simplified, by changing the clause declaring the Skolem constant from $C, (f(y^1, \ldots, y^n) \leqslant S)^{\mathsf{t}}$ to the unit clause $(f(y^1, \ldots, y^n) \leqslant S)^{\mathsf{t}}$. Naturally the resolution calculus has to be changed accordingly, as we will see below.

Furthermore the discussion above is obsolete and the assumption 3.3 can be taken back.

Remark 3.5 Some transformation rules for multi-labels look more natural and symmetric than those for single truth values. For instance we have the rule:

$$\frac{C, (A \wedge B)^{\mathsf{fu}}}{C, A^{\mathsf{fu}}, B^{\mathsf{fu}}}$$

As usual the reduction to clause normal form conserves satisfiability.

Theorem 3.6 *Let Φ be a set of labelled sentences, then the clause normal form* $\mathbf{CNF}(\Phi)$ *is satisfiable, iff Φ is.*

Resolution Calculus ($\mathcal{RPF}$)

Now proceed to give a simple resolution calculus, which utilizes unsorted unification. However despite its name the calculus still utilizes the sort information present in the clause set and therefore gives considerably improved search behavior over unsorted methods as in [14]. In the next section, we will further improve the calculus by using

sorted unification algorithm, which delegates parts of the search into the unification algorithm.

For unsorted substitutions the naive resolution rule is unsound. Therefore we have to add a residual (the sort constraint) that ensures the well-sortedness of the unifier.

Definition 3.7 (Sort Constraints) Let $\sigma = [t^1/x^1_{S_1}], \ldots, [t^n/x^n_{S_n}]$ be a substitution, then we define the *sort constraint for* σ to be the clause

$$\mathcal{SC}(\sigma) := \{[t^1 \leqslant S_1]^{\mathsf{fu}}, \ldots, [t^n \leqslant S_n]^{\mathsf{fu}}\}$$

Definition 3.8 (Resolution Inference Rules ($\mathcal{RPF}$))

$$\frac{L^\alpha, C \qquad M^\beta, D}{\sigma(C), \sigma(D), \mathcal{SC}(\sigma)}\, Res \qquad \frac{L^\alpha, M^\alpha, C}{\sigma(L^\alpha), \sigma(C), \mathcal{SC}(\sigma)}\, Fac$$

$$\frac{(t \leqslant \mathfrak{D})^{\mathsf{f}}, C \qquad L^\gamma, D}{\rho(C), \rho(D), \mathcal{SC}(\rho)}\, Strict$$

where $\alpha \neq \beta$ and $\gamma \in \{\mathsf{t}, \mathsf{f}\}$. For *Res* and *Fac* the substitution σ is the most general (unsorted) unifier of L and M and for *Strict* there exists a subterm s of L, such that ρ is a most general unifier of t and s.

Remark 3.9 Note that clauses containing A^{fut} are tautologous and can therefore be deleted in the generation of the clause normal form as well as in the deduction process. The calculus can be extended by the usual subsumption rule, allowing to delete clauses that are subsumed (super-sets).

In the case where we have assumed non-empty sorts we have to provide declarations (unit clauses) of the form $(c^S \leqslant S)^{\mathsf{t}}$ with new constants c^S for all sorts $S \in \mathcal{S}^*$ in order to obtain a complete calculus.

Definition 3.10 Let A be a sentence and Φ be the clause normal form of the set $\{\{A^{\mathsf{f}}\}, \{A^{\mathsf{u}}\}\}$ then we say that A can be derived in $\mathcal{RPF}$ ($\vdash A$), iff there is a derivation of the empty clause $\square$ from Φ with the inference rules above.

Example 3.11 Now we can come back to the example from the exposition. The assertion is not a theorem of $\mathcal{SKL}$, since the clause normal form of the instance $\{\{(1 = \frac{1}{0} \Rightarrow 1 = 0 * 1)^{\mathsf{f}}\}, \{(1 = \frac{1}{0} \Rightarrow 1 = 0 * 1)^{\mathsf{u}}\}\}$:

$$(1 = \tfrac{1}{0})^{\mathsf{u}}, (1 = \tfrac{1}{0})^{\mathsf{t}}$$
$$(1 = 0 * 1)^{\mathsf{u}}, (1 = 0 * 1)^{\mathsf{f}}$$

is satisfiable. In fact in any reasonable formalization of elementary algebra $1 = \frac{1}{0}$ is undefined, whereas $1 = 0 * 1$ is false. Thus, since $\mathcal{RPF}$ is sound (cf. 3.13), the example cannot be a theorem.

Remark 3.12 In practical applications most problems will be of the form $A := (A_1 \wedge \ldots \wedge A_n \Rightarrow C)$ where the A_i are the assumptions and C is the intended conclusion. In contrast to classical first-order predicate logic where it suffices to take the clause normal form of $\{\{A_1^{\mathsf{t}}\}, \ldots, \{A_n^{\mathsf{t}}\}, \{C^{\mathsf{f}}\}\}$ the situation here is more complex, since in $\mathcal{SKL}$ we also have to refute the case that A gets the value $\mathbf{u}$. It is however easy to see, that we can start the calculation of the clause normal form with the set

$$\left\{\{A_1^{\mathsf{ut}}\}, \ldots, \{A_n^{\mathsf{ut}}\}, \{C^{\mathsf{fu}}\}\right\}$$

or with the sets

$$\left\{\{A_1^{\mathsf{ut}}\}, \ldots, \{A_n^{\mathsf{ut}}\}, \{A_1^{\mathsf{u}}, \ldots, A_n^{\mathsf{u}}\}, \{C^{\mathsf{u}}\}\right\} \qquad (*)$$
$$\left\{\{A_1^{\mathsf{t}}\}, \ldots, \{A_n^{\mathsf{t}}\}, \{C^{\mathsf{fu}}\}\right\} \qquad (**)$$

which have to be refuted by the resolution calculus independently. In the second case the refutation can be split in two independent proofs, thus reducing the search space considerably. Nevertheless, the refutation of the set $(*)$ is impractical except for trivial examples. Fortunately in mathematical practice the assumptions A_i often have the status of axioms, which are assumed to be true independently of the theorem[3]. Then the problem is really of the form

$$A' := (A_1 \wedge !A_1 \wedge \ldots \wedge A_n \wedge !A_n \Rightarrow C)$$

The clause normal form of A' is just that of $(**)$, which is close to the classical case in derivational complexity. In particular the background theory formalized by the A_i results in exactly the same clauses as in the classical case.

Extended Example (continued)

Following the discussion above we will continue our extended example with the calculation of the clause normal form $(**)$ of $\mathrm{A1} \wedge !\mathrm{A1} \wedge \ldots \wedge \mathrm{A5} \wedge !\mathrm{A5} \Rightarrow \mathrm{T}$. Since $\mathbb{R}$ and $\mathbb{R}^*$ are not empty, we use the simplified quantification rules of remark 3.4 and provide the declaration $1 \leqslant \mathbb{R}$ and $1 \leqslant \mathbb{R}^*$, which we will not need in the particular refutation. Without this assumption clauses T1 through T5 would have extra literals.

A1 $(x_{\mathbb{R}} = 0)^{\mathsf{t}}, (x \leqslant \mathbb{R}^*)^{\mathsf{t}}$

A2 $(\frac{1}{x_{\mathbb{R}^*}} \leqslant \mathbb{R}^*)^{\mathsf{t}}$

A3 $(x^2_{\mathbb{R}^*} > 0)^{\mathsf{t}}$

A4 $(x_{\mathbb{R}} - y_{\mathbb{R}} \leqslant \mathbb{R})^{\mathsf{t}}$

A5 $(x_{\mathbb{R}} - y_{\mathbb{R}} = 0)^{\mathsf{f}}, (x_{\mathbb{R}} = y_{\mathbb{R}})^{\mathsf{t}}$

[3]This is also the very idea of the set of support strategy in resolution theorem proving.

The price for the formal treatment of three-valued partiality has to be paid in the complicated clause normal form of the formula T with the label fu.

T1 $(c{\leqslant}\mathbb{R})^{\mathsf{t}}$

T2 $(d{\leqslant}\mathbb{R})^{\mathsf{t}}$

T3 $(e{\leqslant}\mathbb{R})^{\mathsf{t}}$

T4 $(f{\leqslant}\mathbb{R})^{\mathsf{t}}$

T5 $(g(y_{\mathbb{R}}){\leqslant}\mathbb{R})^{\mathsf{t}}$

T6 $(c = d)^{\mathsf{f}}, (e = f)^{\mathsf{fu}}$

T7 $(c = d)^{\mathsf{f}}, \left(\left(\frac{1}{e-f}\right)^2 > 0\right)^{\mathsf{fu}}$

T8 $\left(\left(\frac{1}{c-d}\right)^2 > 0\right)^{\mathsf{f}}, (e = f)^{\mathsf{fu}}$

T9 $\left(\left(\frac{1}{c-d}\right)^2 > 0\right)^{\mathsf{f}}, \left(\left(\frac{1}{e-f}\right)^2 > 0\right)^{\mathsf{fu}}$

Eight further clauses resulting from the theorem are not shown here, four are tautologies, four others not needed for the derivation below.

T6 & A5$\longrightarrow$R1 $(c - d = 0)^{\mathsf{f}}, (e = f)^{\mathsf{fu}}, (c{\leqslant}\mathbb{R})^{\mathsf{fu}}, (d{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R1 & A1$\longrightarrow$R2 $(c - d{\leqslant}\mathbb{R}^*)^{\mathsf{t}}, (e = f)^{\mathsf{fu}}, (c - d{\leqslant}\mathbb{R})^{\mathsf{fu}}, (c{\leqslant}\mathbb{R})^{\mathsf{fu}}, (d{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R2 & A4$\longrightarrow$R3 $(c - d{\leqslant}\mathbb{R}^*)^{\mathsf{t}}, (e = f)^{\mathsf{fu}}, (c{\leqslant}\mathbb{R})^{\mathsf{fu}}, (d{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R3 & T1$\longrightarrow$R4 $(c - d{\leqslant}\mathbb{R}^*)^{\mathsf{t}}, (e = f)^{\mathsf{fu}}, (d{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R4 & T2$\longrightarrow$R5 $(c - d{\leqslant}\mathbb{R}^*)^{\mathsf{t}}, (e = f)^{\mathsf{fu}}$

T8 & A3$\longrightarrow$R6 $(e = f)^{\mathsf{fu}}, \left(\left(\frac{1}{c-d}\right){\leqslant}\mathbb{R}^*\right)^{\mathsf{fu}}$

R5 & A2$\longrightarrow$R7 $(e = f)^{\mathsf{fu}}, (c - d{\leqslant}\mathbb{R}^*)^{\mathsf{fu}}$

R7 & R5$\longrightarrow$R8 $(e = f)^{\mathsf{fu}}$

Analogously, clause T7 can be reduced with T9 to R16.

$\ldots$ & $\ldots$ $\longrightarrow$R16 $\left(\left(\frac{1}{e-f}\right)^2 > 0\right)^{\mathsf{fu}}$

R16 & A3 $\longrightarrow$R17 $\left(\frac{1}{e-f}{\leqslant}\mathbb{R}^*\right)^{\mathsf{fu}}$

R17 & A2 $\longrightarrow$R18 $(e - f{\leqslant}\mathbb{R}^*)^{\mathsf{fu}}$

R18 & A1 $\longrightarrow$R19 $(e - f = 0)^{\mathsf{t}}, (e - f{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R19 & A4 $\longrightarrow$R20 $(e - f = 0)^{\mathsf{t}}, (e{\leqslant}\mathbb{R})^{\mathsf{fu}}, (f{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R20 & A5 $\longrightarrow$R21 $(e = f)^{\mathsf{t}}, (e{\leqslant}\mathbb{R})^{\mathsf{fu}}, (f{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R21 & T3 $\longrightarrow$R22 $(e = f)^{\mathsf{t}}, (f{\leqslant}\mathbb{R})^{\mathsf{fu}}$

R22 & T4 $\longrightarrow$R23 $(e = f)^{\mathsf{t}}$

R8 & R23$\longrightarrow$R24 $\square$

Soundness and Completeness

Theorem 3.13 (Soundness) *Let Φ be set of clauses with $\Phi \vdash \square$, then Φ is unsatisfiable.*

Proof sketch: The soundness of the resolution and factoring rules is established in the usual way taking into account that the sort constraints make the substitutions

"well-sorted" and thus compatible with the semantics: The sort constraints add two sort literals $(t{\leqslant}S)^{\mathsf{f}}, (t{\leqslant}S)^{\mathsf{u}}$ per component of the substitution, which only can be refuted if indeed $(t{\leqslant}S)^{\mathsf{t}}$.

The *Strict* rule is sound, because functions and predicates in $\mathcal{SKL}$ are strict and thus undefined subterms of a literal make the literal undefined. □

Definition 3.14 Let $C := \{L_1^{\alpha_1}, \ldots, L_n^{\alpha_n}\}$ be a clause, then the *conditional instantiation* $\sigma{\downarrow}(C)$ *of* σ *to* C is defined by

$$\sigma{\downarrow}(C) := \{\sigma(L_1^{\alpha_1}), \ldots, \sigma(L_n^{\alpha_n})\} \cup \mathcal{SC}(\sigma|_{\mathbf{Free}(C)})$$

The following result from [20] is independent of the number of truth values.

Lemma 3.15 *Conditional instantiation is sound: for any clause* C*, substitution* σ *and* Σ*-model* $\mathcal{M}$ *we have that* $\mathcal{M} \models \sigma{\downarrow}(C)$*, whenever* $\mathcal{M} \models C$*.*

Definition 3.16 Let A be a well-formed sentence and $\mathbf{CNF}(A)$ be the clause normal form of A, then we define the *Herbrand set of clauses* $\mathbf{CNF}_H(A)$ *for* A to be

$$\mathbf{CNF}_H(A) := \{\sigma{\downarrow}(C) \mid C \in \mathbf{CNF}(A), \sigma \text{ ground substitution}, \mathbf{Dom}(\sigma) = \mathbf{Free}(C)\}$$

Definition 3.17 We will call two literals L^α and L^β *complementary*, if $\alpha \neq \beta$ and literals L^γ and $(t{\leqslant}\mathfrak{D})^{\mathsf{f}}$ $\perp$*-complementary*, if t is a subterm of L and $\gamma \in \{\mathsf{t}, \mathsf{f}\}$.

Definition 3.18 (Herbrand Model) Let Φ be a set of clauses, then the *Herbrand base* $\mathcal{H}(\Phi)$ *of* Φ is defined to be the set of all ground atoms containing only function symbols that appear in the clauses of Φ. If there is no constant in Φ, we add a new constant c. A *valuation* ν is a function $\mathcal{H}(\Phi) \longrightarrow \{\mathsf{f}, \mathsf{u}, \mathsf{t}\}$, such that for all atoms $L, M \in \mathcal{H}(\Phi)$ the literals $L^{\nu(L)}$ and $M^{\nu(M)}$ are not $\perp$-complementary. Note that these literals are not complementary since ν is a function. The Σ*-Herbrand Model* $\mathcal{H}$ *for* Φ *and* ν is the set $\mathcal{H} := \{L^\alpha \mid \alpha = \nu(L), L \in \mathcal{H}(\Phi)\}$.

We say that a Σ-Herbrand model $\mathcal{H}$ *satisfies a clause set* Φ iff for all ground substitutions σ and clauses $C \in \Phi$ we have $\sigma{\downarrow}(C) \cap \mathcal{H} \neq \emptyset$. A clause set is called Σ*-Herbrand-unsatisfiable* iff there is no Σ-Herbrand-model for Φ.

Theorem 3.19 (Herbrand Theorem) *Let* A *be a well-formed formula, then the clause normal form* $\mathbf{CNF}(A)$ *has a* Σ*-model iff* $\mathbf{CNF}_H(A)$ *has a* Σ*-Herbrand-model.*

Proof: Let $\mathcal{M} = (\mathcal{A}, \mathcal{I})$ be a Σ-model for $\Phi := \mathbf{CNF}(A)$. We will see that

$$\mathcal{H} := \{L^\alpha \mid L \in \mathcal{H}(\Phi), \alpha = \mathcal{I}_\varphi(L)\}$$

is a Σ-Herbrand model for $\Psi := \mathbf{CNF}_H(A)$ if φ is an arbitrary Σ-assignment. It is immediately clear that $\mathcal{I}_\varphi$ is a valuation, therefore $\mathcal{H}$ is Σ-Herbrand model. Assume that it is not a Σ-Herbrand model for Ψ, that is, there is a clause $C \in \Psi$, such that

$\mathcal{H} \cap C = \emptyset$. Since $C \in \Psi$ there is a substitution $\sigma = [t^i/x^i_{S_i}]$ and a clause $D \in \Phi$, such that $C = \sigma\downarrow(D) = \sigma(D) \cup \mathcal{SC}(\sigma)$.

Without loss of generality we can assume that $\mathcal{I}(S_i)(\mathcal{I}_\varphi(t^i)) = \mathsf{t}$, since otherwise $\mathcal{I}_\varphi(t^i \lessdot S^i) \in \{\mathsf{f}, \mathsf{u}\}$, and therefore $(t^i \lessdot S^i)^\gamma \in \mathcal{H}$ for $\gamma \in \{\mathsf{f}, \mathsf{u}\}$, which contradicts the assumption. Thus the mapping $\psi := \varphi, [\mathcal{I}_\varphi(t^i)/x^i]$ is a Σ-assignment.

Note that since $\mathcal{M}$ is a model of Φ, we have that $\mathcal{M} \models D$ and therefore there is a literal $L^\alpha \in D$, such that $\alpha = \mathcal{I}_\psi(L) = \mathcal{I}_\varphi(\sigma(L))$, hence $\sigma(L) \in \mathcal{H}$, which contradicts the assumption.

For the converse direction let $\mathcal{H}$ be a Σ-Herbrand model for Ψ. To construct a Σ-model $\mathcal{M}$ for Φ we first construct a partial Σ-algebra $(\mathcal{A}, \mathcal{I})$. Let

$$\mathcal{A} := \{t \mid \exists L^\alpha \in \mathcal{H} \text{ where } \alpha \in \{\mathsf{f}, \mathsf{t}\} \text{ and } t \text{ subterm of } L\}$$

and let $\mathcal{I}(S)$, $\mathcal{I}(f^n)$ and $\mathcal{I}(P^n)$ be partial functions, such that

$$\begin{aligned} \mathcal{I}(S)(t) = \mathsf{t} \quad &\text{iff} \quad (t \lessdot S)^{\mathsf{t}} \in \mathcal{H} \\ \mathcal{I}(f^n)(t^1, \ldots, t^n) := f^n(t^1, \ldots, t^n) \quad &\text{iff} \quad f^n(t^1, \ldots, t^n) \in \mathcal{A} \\ \mathcal{I}(P^n)(t^1, \ldots, t^n) := \alpha \quad &\text{iff} \quad (P^n(t^1, \ldots, t^n))^\alpha \in \mathcal{H} \end{aligned}$$

Now let $\mathcal{M}$ be the strict Σ-algebra corresponding to the partial Σ-algebra $(\mathcal{A}, \mathcal{I})$. We proceed by convincing ourselves that $\mathcal{M} \models \Phi$. Let $C \in \Phi$ and $\varphi := [t^i/x^i_{S_i}]$ be an arbitrary Σ-assignment. Since $\mathcal{A}$ is a set of ground terms φ is also a ground substitution and moreover $(t^i \lessdot S_i)^{\mathsf{t}} \in \mathcal{H}$ by construction of $\mathcal{I}$.

Since $\mathcal{H}$ is a Σ-Herbrand model for Ψ we have $\varphi\downarrow(C) \cap \mathcal{H} = (\varphi(C) \cup \mathcal{SC}(\varphi)) \cap \mathcal{H} \neq \emptyset$. Since $\mathcal{H}$ cannot contain complementary literals we must already have a literal $\varphi(L^\alpha) \in \varphi(C) \cap \mathcal{H}$. Now let ν be the valuation associated with $\mathcal{H}$. Since $\varphi(L^\alpha) \in \mathcal{H}$ we have $\alpha = \nu(\varphi(L)) = \mathcal{I}_\varphi(L)$, which implies $\mathcal{M} \models_\varphi L^\alpha$. Since we have taken C and φ arbitrary, we get the assertion. □

Corollary 3.20 *A set Φ of ground unit clauses is unsatisfiable iff it contains two complementary or $\bot$-complementary literals.*

Theorem 3.21 (Ground Completeness) *Let Φ be an unsatisfiable set of ground clauses, then there exists a $\mathcal{RPF}$ derivation of the empty clause from Φ.*

Proof: The proof is analogous to the standard k-parameter proof of Anderson and Bledsoe [1]. We show be induction on $k := \sum_{C \in \Phi}(\mathrm{card}(C) - 1)$ that there exists a refutation for Φ.

If $k = 0$ then Φ is a set of ground unit clauses. Therefore by Lemma 3.20 and the assumed unsatisfiability there has to be a pair of complementary or $\bot$-complementary literals in Φ. Thus a single application of the rule *Res* or *Strict* yields the empty clause.

If $k > 0$, then there is a non-unit clause $C =: C_1 \cup C_2 \in \Phi$. If $\Phi = \Phi' \cup \{C\}$ then the k parameters for $\Phi_1 := \Phi' \cup \{C_1\}$ and $\Phi_2 := \Phi' \cup \{C_2\}$ are smaller than k and

therefore by inductive hypothesis there are refutations for Φ_1 and Φ_2 which can be combined to a refutation for Φ, since Φ is ground. □

Theorem 3.22 (Completeness) *The calculus consisting of the rules Res, Fac, and Strict is refutation complete.*

Proof: For the proof of this assertion we combine the completeness result from the ground case with a lifting argument. It turns out that the lifting property can be established by methods from [20], since they are independent of the number of truth values. □

4 Resolution with Order-Sorted Unification

The calculus defined above can still be improved by introducing an order-sorted unification.

Definition 4.1 (Conditional Objects) A pair $o^c = @\|C$ is called a *conditional declaration (a conditional term, a conditional substitution)*, if C is a set of literals and @ is a declaration $t\leqslant S$ (a term t, a substitution σ). We will call a conditional object ground, iff @ is ground. We define the application of a conditional substitution $\sigma^c := \sigma\|D$ to a conditional term $o^c := @\|C$ (denoted by $\sigma^c(o^c)$) to be $\sigma(@)\|\sigma(C)\cup D$.

Definition 4.2 Let $\mathcal{D}$ be a set of conditional declarations, then the set $\mathbf{wsT}_S(\Sigma,\mathcal{D})$ of *well-sorted conditional terms of sort S* is inductively defined by

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

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

Unification

Definition 4.3 (Unification Problem) Let $\Gamma := \{s^1 = t^1,\ldots,s^n = t^n\}$ be a set of two-element multi-sets of terms (called *disagreement set*), then we will call a pair $\Gamma\|D$ a *unification problem*, that is, an unification problem is a conditional disagreement set. It is called well-sorted, iff $t^i\|C_i$ and $s^i\|D_i \in \mathbf{wsT}(\Sigma,\mathcal{D})$ for some sets C_i, D_i of literals, such that $\bigcup_i C_i \cup \bigcup_i D_i = D$. We will call a unification problem *solved*, if

- it is of the form $x^1_{S_1} = t^1, \ldots, x^n_{S_n} = s^n, y^1 = y^1, \ldots, y^m = y^m \| C$, and
- the x^i are distinct and do not occur on the right hand sides of equations.
- $t^i \in \mathbf{wsT}_{S_i}(\Sigma, \mathcal{D})$ for all $1 \leq i \leq n$.

Now we will present a set of transformations for a nondeterministic unification algorithm that computes complete sets[4] of unifiers for well-sorted unification problems. The nondeterministic unification algorithm starts with a well-sorted unification problem Γ and enumerates the set of irreducible unification problems from Γ. Such a unification problem is called a *success node*, if it is in solved form and a *failure node* otherwise. The set of substitutions corresponding to the success nodes is the output of the algorithm.

Definition 4.4 (Order-sorted Unification) The following inference system gives a non-deterministic algorithm for order-sorted unification in $\mathcal{SKL}$.

elim-var $\dfrac{x_S = y_T, \Gamma \| C}{x = y, [y/x]\Gamma \| [y/x]C \cup D}$ if $z_T \lessdot S \| D \in \mathcal{D}$ or $S = T$ (then D is empty).

decompose $\dfrac{f(s^1, \ldots, s^n) = f(t^1, \ldots, t^n), \Gamma \| C}{s^1 = t^1, \ldots, s^n = t^n, \Gamma \| C}$

imitate $\dfrac{x_S = f(s^1, \ldots, s^n), \Gamma \| C}{x_S = f(t^1, \ldots, t^n), s^1 = t^1, \ldots, s^n = t^n, \Gamma \| C \cup D}$ if $f(t^1, \ldots, t^n) \| D \in \mathcal{D}$ or $S = \mathfrak{D}$ (then D is empty) and furthermore $x_S \notin \mathbf{Free}(f(t^1, \ldots, t^n))$.

intersect $\dfrac{x_S = y_T, \Gamma \| C}{x_S = z_V, y_T = z_V, \Gamma \| C \cup D^1 \cup D^2}$ if $z^1_V \lessdot S \| D^1$ and $z^2_V \lessdot S \| D^2 \in \mathcal{D}$

non-reg $\dfrac{x_S = y_T, \Gamma \| C}{x_S = (f(s^1, \ldots, s^n), y_T = (f(t^1, \ldots, t^n), s^1 = t^1, \ldots, s^n = t^n, \Gamma \| C \cup D^1 \cup D^2}$
if $f(s^1, \ldots, s^n) \lessdot S \| D^1$ and $f(t^1, \ldots, t^n) \lessdot T \| D^2 \in \mathcal{D}$.

For the rules **imitate**, **intersect**, and **non-reg** we assume the sets of variables in the declarations in $\mathcal{D}$ and the unification problem to be disjoint. If this is not the case the variables have to be renamed in the declarations. For all newly introduced $x = t$, these rules are directly followed by applications of the rule

elim-new $\dfrac{x = t, \Gamma \| C}{x = t, [t/x]\Gamma \| [t/x]C}$

[4]Here the instantiation ordering for completeness is just that for ordinary substitutions, since this set of transformations is clearly not complete for the instantiation ordering derived from the definition of composition for conditional substitutions.

Remark 4.5 Note that we have to keep trivial variable pairs in our solved forms, since we do not postulate transformation rule for deleting trivial pairs (constants and function symbols can be deleted by the decompose rule). This trick prevents the loss of already used variables from the unification problem and eases the freshness conditions (we only have to consider the free variables of the current unification problem) in the rules **imitate**, **intersect** and **non-reg**.

In contrast to the related set of rules for order-sorted unification in [21] or [16] 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.

Theorem 4.6 *The above set of rules define a sound and complete non-deterministic 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 we also record the respective conditions) and are therefore well-sorted by definition and the set of well-sorted terms is closed under well-sorted substitutions.

The rest of the soundness and completeness proof is independent of the conditions, since we have chosen the instantiation ordering independently. In particular solved forms are independent of the conditions. Since without conditions the set of inference rules corresponds to that given in [16, p.98], we refer to the proofs given there. □

Resolution ($\mathcal{RPF}(\mathcal{D})$)

The notion of substitution discussed above is not yet the one appropriate for a resolution calculus, where substitutions are required to have ground instances. otherwise the resolution rule becomes unsound: Let S be a sort that does not have ground terms, *i.e.* where $\mathcal{A}_S$ may be empty, then the clause set $\{\{(Px_S)^{\mathsf{t}}\}, \{(Py_S)^{\mathsf{f}}\}\}$ would be refutable, 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 4.7 *Let $\mathcal{D}$ be a set of conditional sort declarations, then the problem whether the set of conditional ground terms of sort S is empty is decidable. Furthermore the set of conditions for the nonemptyness with respect to $\mathcal{D}$ is effectively computable.*

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

In particular the set $Ax(\mathcal{D})$ is a Horn clause set, therefore propositional SLD-resolution with an appropriate strategy is a decision algorithm for the problem $Ax(\mathcal{D}) \models S$. From a SLD proof $Ax(\mathcal{D}) \vdash S$, the set of conditions for S to have ground formulae can be computed by identifying the declarations corresponding to the clauses in the proof and collecting the suitably instantiated versions of their conditions. □

Definition 4.8 Let S be a sort, then we define $\nu(S)$ to be the set of conditions computed by the algorithm sketched in 4.7, if S is nonempty and $\nu(S)$ a tautology if S is empty. For a substitution $\sigma := [t^1/x^1], \ldots [t^n/x^n]$ let $\mathcal{S}(\sigma)$ the set of all sorts of variables free in $t^1, \ldots, t^n$, then we define $\nu(\sigma) := \bigcup_{S \in \mathcal{S}(\sigma)} \nu(S)$.

Now we can adapt the resolution calculus to order-sorted unification:

Definition 4.9 (Resolution with Order-Sorted Unification ($\mathcal{RPF}(\mathcal{D})$))
Let $\mathcal{D}$ be a set of conditional declarations

$$\frac{L^\alpha, C \qquad M^\beta, D}{\sigma(C), \sigma(D), E, \nu(\sigma)} \, Res(\mathcal{D}) \qquad \frac{L^\alpha, M^\alpha, C}{\sigma(L^\alpha), \sigma(C), E, \nu(\sigma)} \, Fac(\mathcal{D})$$

$$\frac{(t \lessdot \mathfrak{D})^{\mathrm{f}}, C \qquad L^\gamma, D}{\rho(C), \rho(D), F, \nu(\rho)} \, Strict(\mathcal{D})$$

where $\alpha \neq \beta$ and $\gamma \in \{\mathrm{t}, \mathrm{f}\}$. For $Res(\mathcal{D})$ and $Fac(\mathcal{D})$ the substitution $\sigma \| E$ is the most general well-sorted unifier of L and M and for $Strict(\mathcal{D})$ there exists a subterm s of L, such that $\rho \| F$ is a most general well-sorted unifier of t and s.

Note that by residuating $\nu(\sigma)$ in our calculus, we have not prohibited inferences with substitutions that do not have ground instances. We have merely rendered the generated clauses tautologous. A practical implementation would not add such clauses, since they can never contribute to a refutation.

Let Ψ be a clause set and

$$\mathcal{D}^M(\Psi) := \{t \lessdot S \| C \mid C, (t \lessdot S)^{\mathrm{t}} \in \Psi\}$$

and $\mathcal{D}^C(\Psi) \subseteq \mathcal{D}^M(\Psi)$ a subset, such that for any clause of the form $C, (t \lessdot S)^{\mathrm{t}}$ in Ψ there is exactly one conditional declaration $t \lessdot S \| C$ in $\mathcal{D}^C(\Psi)$.

These definitions give us two variants of the resolution calculus, $\mathcal{RPF}(\mathcal{D}^M)$ if we take $\mathcal{D}$ to be $\mathcal{D}^M(\Psi)$ or $\mathcal{RPF}(\mathcal{D}^C)$, if we take $\mathcal{D} := \mathcal{D}^C(\Psi)$. For the latter variant we have to reevaluate the set $\mathcal{D}^C(\Psi)$ after each inference step to obtain a complete calculus.

Remark 4.10 In the case of non-empty sorts we can simplify the inference rules by deleting the non-emptyness conditions $\nu(\sigma)$ and $\nu(\rho)$ in the definition of $\mathcal{RPF}(\mathcal{D})$. Furthermore, unlike in the simple resolution calculus we do not need to add new constant declarations for completeness.

Remark 4.11 At first glance the use of order-sorted unification in the calculus is not a great improvement over that with unsorted unification, since in the refined calculus residuation is also required. The difference in the calculi is that each use of a conditional declaration in the unification algorithm of $\mathcal{RPF}(\mathcal{D})$ has to be imitated by a resolution step in $\mathcal{RPF}$. The conditions residuated in $\mathcal{RPF}(\mathcal{D})$ are only those, that are not yet present as positive information in the clause set, whereas those of $\mathcal{RPF}$ are all that are needed for well-sortedness irrelevant of the sort information already present in the clause set.

Theorem 4.12 *Both variants $\mathcal{RPF}(\mathcal{D}^C)$ and $\mathcal{RPF}(\mathcal{D}^M)$ are sound and complete.*

Proof: The methods from [21] apply to $\mathcal{SKL}$, since unification is only concerned with terms and the difference in the number of truth values does not affect the term structure. □

Extended Example (continued)

Following the discussion above we will continue our extended example and show a proof using order-sorted unification. The refined calculus uses the same clause normal form as in the unsorted case.

$$
\begin{array}{lll}
\text{T6} \;\&\; \text{A5}\longrightarrow\text{R1} & (c-d=0)^{\mathsf{f}}, (e=f)^{\mathsf{fu}} \\
\text{R1} \;\&\; \text{A1}\longrightarrow\text{R2} & (c-d{\leqslant}\mathbb{R}^*)^{\mathsf{t}}, (e=f)^{\mathsf{fu}} \\
\text{T8} \;\&\; \text{A3}\longrightarrow\text{R3} & (e=f)^{\mathsf{fu}} \\
\text{T7} \;\&\; \text{A5}\longrightarrow\text{R4} & (c-d=0)^{\mathsf{f}}, \left(\left(\frac{1}{e-f}\right)^2 > 0\right)^{\mathsf{fu}} \\
\text{R4} \;\&\; \text{A1}\longrightarrow\text{R5} & (c-d{\leqslant}\mathbb{R}^*)^{\mathsf{t}}, \left(\left(\frac{1}{e-f}\right)^2 > 0\right)^{\mathsf{fu}} \\
\text{T9} \;\&\; \text{A3}\longrightarrow\text{R6} & \left(\left(\frac{1}{e-f}\right)^2 > 0\right)^{\mathsf{fu}} \\
\text{R6} \;\&\; \text{A3}\longrightarrow\text{R7} & (e-f=0)^{\mathsf{t}} \\
\text{R7} \;\&\; \text{A5}\longrightarrow\text{R8} & (e=f)^{\mathsf{t}} \\
\text{R3} \;\&\; \text{R8}\longrightarrow\text{R9} & \square
\end{array}
$$

Note that clauses R2 and R5 are conditional declarations that have been added to our set of declarations, these additional declarations have made resolution steps R3 and R7 possible. Now we see the improvement of the refined calculus, where we need 9 steps as compared to 24 steps in the unsorted case. One can easily imagine the magnitude of the search space and the proof for the relativized formulation.

5 Conclusion

We have developed an order sorted three-valued logic for the formalization of informal mathematical reasoning with partial functions. This system generalizes the system proposed by Kleene in [12] 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. Furthermore we have presented a sound and complete resolution calculus with dynamic sorts for our system, 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. [2, 11]) for a correct treatment of the undefined and order-sorted logics (see [20, 21]) for an adequate treatment of the defined. It differs from the sequent calculus in [14] 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.

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 efficient mechanization of equality, here paramodulation (cf. [15]) or even superposition ([4]) methods would be interesting to study. However, we believe that this endeavor will mainly involve the development of the sort aspects for these calculi, because we think that the aspects of three-valuedness will not be critical.

On the other hand, the mechanization of higher-order features is essential 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. [8, 9] 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 3.11) can be generalized in much the same manner and will be a useful tool for formalizing mathematics.

References

[1] R. Anderson and W.W. Bledsoe. A linear format for resolution with merging and a new technique for establishing completeness. *Journal of the ACM*, **17**:525–534, 1970.

[2] Matthias Baaz and Christian G. Fermüller. Resolution for many-valued logics. In A. Voronkov, editor, *Proceedings of International Conference on Logic Programming and Automated Reasoning*, pages 107–118, St. Petersburg, Russia, 1992. Springer Verlag, Berlin, Germany. LNAI 624.

[3] 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, Institut für Computersprachen, 1993. Short version in *Proceedings of the 23rd International Symposium on Multiple Valued Logic*, Sacramento, California, 1993. IEEE Press.

[4] Leo Bachmair and Harald Ganzinger. Non-clausal resolution and superposition with selection and redundancy criteria. In A. Voronkov, editor, *Proceedings of International Conference on Logic Programming and Automated Reasoning*, pages 273–284, St. Petersburg, Russia, 1992. Springer Verlag, Berlin, Germany. LNAI 624.

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

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

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

[8] William M. Farmer. A partial functions version of Church's simple theory of types. Technical Report M88-52, Revision 1, The MITRE Corporation, Bedford, Massachusetts, USA, February 1990.

[9] 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.

[10] Alan M. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, **49**:161–198, 1991.

[11] 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, Oxford, England, 1994.

[12] Stephen Cole Kleene. *Introduction to Metamathematics.* Van Nostrand, Amsterdam, The Netherlands, 1952.

[13] Lothar Kreiser, Siegfried Gottwald, and Werner Stelzner, editors. *Nichtklassische Logik.* Akademie Verlag, Berlin, Germany, 1990.

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

[15] Arthur Robinson and Larry Wos. Paramodulation and TP in first order theories with equality. *Machine Intelligence*, 4:135–150, 1969.

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

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

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

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

[20] 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.668–693.

[21] 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.