Mechanising Partiality without Re-Implementation

Manfred Kerber[1] and Michael Kohlhase[2]

[1] The University of Birmingham, School of Computer Science
Birmingham, B15 2TT, England
e-mail: M.Kerber@cs.bham.ac.uk
WWW: http://www.cs.bham.ac.uk/~mmk

[2] Universität des Saarlandes, FB Informatik
D-66041 Saarbrücken, Germany
e-mail: kohlhase@cs.uni-sb.de
WWW: http://jswww.cs.uni-sb.de/~kohlhase

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. This approach allows rejecting certain unwanted formulae as faulty, which the simpler two-valued ones accept. We have developed resolution and tableau calculi for automated theorem proving that take the restrictions of the three-valued logic into account, which however have the severe drawback that existing theorem provers cannot directly be adapted to the technique. Even recently implemented calculi for many-valued logics are not well-suited, since in those the quantification does not exclude the undefined element. In this work we show, that it is possible to enhance a two-valued theorem prover by a simple strategy so that it can be used to generate proofs for the theorems of the three-valued setting. By this we are able to use an existing theorem prover for a large fragment of the language.

1 Introduction

Many practical applications of deduction systems in mathematics, philosophical logic and computer science rely on the correct and efficient treatment of partiality. For instance, in order to describe formally the semantics of computer programs, the logic has to be able to model that real programs may crash (i.e, are only partial functions from inputs to outputs). For example, one would like to distinguish the faulty type description of the tail function "$l\colon list \Rightarrow \mathsf{tail}(l)\colon list$" from the correct one "$l\colon list\ l \neq [\,] \Rightarrow \mathsf{tail}(l)\colon list$". Such differences can be made formal in the VDM language (see for instance [Jon90, p.68ff] or [BFL*94, p.3]). Unfortunately up to now there is no efficient mechanisation of reasoning with partiality in VDM.

There are different approaches – ranging from workarounds for concrete situations to a proper general treatment – to model partiality. For an overview[i], we

[i] For a more detailed discussion of the different approaches compare [Far90].

will introduce the main approaches and exemplify their advantages and disadvantages by some trivial examples from arithmetic. We have chosen this domain for its clarity, even though for mathematics a logical treatment of partiality might successfully be replaced by a workaround.

We will recall the four main options of treating partiality and then advocate the fourth one. In the first approach, undefined expressions like $1/0$ are syntactically excluded, for instance by using a sorted logic. In the second approach, partiality is either disregarded or bypassed, for instance, a value is assigned to $1/0$, 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, $1/0 = 0$, or in the second case from $0 \cdot x = 0$ the instance $0 \cdot 1/0 = 0$. This approach is not satisfying if such theorems are unwanted and that is normally the case in mathematics.

In the third and fourth, partiality is taken seriously and this is reflected in the semantics and the calculus. While the third considers undefined terms only, but atomic formulae are evaluated either to false or true, in the fourth, atomic formulae can be undefined too, that is, be evaluated to a third truth value "undefined". Concretely, in the third approach terms of the form $1/0$ are treated as undefined and all atomic formulae containing such a meaningless term are evaluated to 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 un-intuitive to the working mathematician. For instance in elementary arithmetic the following sentence

$$\forall x_{\mathbb{R}}, y_{\mathbb{R}}, z_{\mathbb{R}\bullet}\ z = \frac{x}{y} \Rightarrow x = y * z$$

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 $z = x/0$ 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. In the fourth approach, which has, in particular, been investigated by Kleene in [Kle52], this is not a theorem. In this approach atomic formulae containing meaningless terms are evaluated to undefined. In particular, the example above is not a theorem in the three-valued approach, since for the instantiation $y = 0$ the formula evaluates to undefined.

Now we address the question which price has to be paid for the proper treatment in the three-valued approach. Indeed in unsorted mechanisations of Kleene's approach by Tichy [Tic82], Lucio-Carrasco and Gavilanes-Franco [LG89], it is necessary to pay a high computational price. In [KK94,KK96] we have developed a sorted three-valued logic $\mathcal{SKL}^3$ and corresponding resolution and tableau calculi $\mathcal{RPF}^3$ and $\mathcal{TPF}^3$ carefully integrating ideas from sorted dynamic logics as introduced by Weidenbach [WO90,Wei95] and from many-valued truth-functional logics as mechanised by Hähnle [Häh94] as well as by Baaz and Fermüller [BF95]. In these logics the additional computations are relatively modest and in many cases proofs in the two-valued logic can be the structurally isomorphically transformed into proofs in the three-valued logic.

The main contribution of this paper is the result that for a large class of $\mathcal{SKL}^3$-theorems (which are also classical theorems by construction) the $\mathcal{TPF}^3$

and $\mathcal{RPF}^3$ proofs can be transformed into classical sorted tableau and resolution proofs and vice versa conserving the structure and size of the proofs. Furthermore we can show that by adding a simple strategy in proof search for two-valued theorems, it is possible to use a two-valued theorem prover for proving $\mathcal{SKL}^3$-theorems. However, unlike to the first of the four above-mentioned approaches, ours does not trivialise undefinedness information in a way that it would become decidable.

2 Strong Sorted Kleene Logic ($\mathcal{SKL}^3$)

In [Kle52] Kleene presents a logic, which he calls *strong three-valued logic* for reasoning about partial recursive predicates[ii] 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 1. 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).

In [KK94] we have made Kleene's meta-level discussion of defined and undefined individuals explicit and presented a formal syntax and semantics that we will now present informally.

The universe of discourse is structured into the sort Δ for all defined individuals and an error element $\perp$; 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 respectively. Just as in Kleene's system, our quantifiers only range over individuals in Δ, that is, individuals that are not undefined. Since $\mathcal{SKL}^3$ needs the sort Δ for bounded quantification anyway, it is no further effort to give the full sorted system. 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.

Terms in $\mathcal{SKL}^3$ are ordinary first-order terms. Atomic formulae are defined as usual, in addition, there are atomic formulae of the kind $t{\leqslant}S$, where t is a term and S a sort symbol. Here, $t{\leqslant}S$ stands for "t has sort S". Formulae are built up from atomic formulae by the usual connectives, and a unary connective ! with the intended meaning that $!A$ is true, whenever the value of A is not u. Furthermore, all quantifications are bound by a sort S (i.e., are of the form $\forall x_S.\, A$ or $\exists x_S.\, A$).

The three-valued semantics for $\mathcal{SKL}^3$ has a "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

[ii] Most logic-based accounts of partiality only treat partiality for functions corresponding to the mathematical notion of a partial function, defined as a right-unique relation opposed to a total function which is left-total *and* right-unique. Indeed, at first glance there seems to be no need for having partial relations as well, since relations are defined as subsets of Cartesian products. However, most mathematicians would agree that the relation $x > y$ does not make much sense for arbitrary complex numbers (rather than saying that it is false for most complex numbers), while $x > y$ is perfectly well-defined for real numbers.

minimal. The standard notion of value function, Σ-algebra and assignments directly carry over to the partial-function case. The only interesting part is the non-classical truth functions for the connectives and quantifiers.

Definition 1. 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

As usual the semantics of formulae with respect to an interpretation $\mathcal{I}$ and an assignment φ is defined recursively. The atomic formulae of the form $t \Leftarrow S$ are treated like expressions of the form $S(t)$. For the quantifiers it 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

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

where $\mathsf{Q} \in \{\forall, \exists\}$ and $\varphi, [a/x]$ coincides with φ away from x and maps x to a. Furthermore we define

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

Note that with this definition quantification is separated into a truth-functional part $\widetilde{\mathsf{Q}}$ and an instantiation part that considers members of the universe according to the sort S (that is, those members for which $\mathcal{I}(S)(a) = \mathsf{t}$). Note furthermore, that although there is no semantical difference between sorts and unary predicates, by the definition of the semantics of the quantifiers, only those elements are considered where the sort is defined and evaluates to t. According to this semantics, the relativisation $\Re(\forall x_S.\ A)$ is $\forall x.\ S(x) \wedge !S(x) \Rightarrow \Re(A)$ and not just $\forall x.\ S(x) \Rightarrow \Re(A)$.

Finally, quantification never considers undefined values and therefore cannot be truth-functional even for the unsorted case. As a consequence, we cannot directly use the methods developed for truth-functional many-valued logics from [Häh94,BF95].

Finally, 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 calculi.

While in classical logic, the consequence relation is directly connected to the implication by the deduction theorem, in $\mathcal{SKL}^3$ things are a little bit more difficult, since the classical deduction theorem is not valid. 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. Actually, the $\mathcal{SKL}^3$ deduction theorem has the form $\Phi \cup \{A\} \models B$ iff $\Phi \models A \wedge !A \Rightarrow B$.

Now, the definedness connective ! for formulae does not have an explicit counterpart in informal mathematical practice, instead definedness assumptions are implicitly made in the assumptions. Hence for mathematical applications we will consider so-called *consequents*, that is, pairs consisting of a set of formulae Φ and a formula A, in which all formulae in Φ are assumed to be defined. We call a consequent $\Phi \models A$ valid if A is entailed by Φ in all Σ-models.

In fact the tautologies in the !-free fragment of $\mathcal{SKL}^3$, i.e., valid consequents of the form $\emptyset \models A$, where A does not contain any !, are very limited. The only atoms that are defined in an empty context are of the form $t{\lessdot}\Delta$. Therefore the set of tautologies can be generated by adding disturbances to classical propositional tautologies, where the propositional variables have been replaced by such atoms, for instance $(t{\lessdot}\Delta \Rightarrow t{\lessdot}\Delta) \vee A$ for arbitrary formulae A.

Now we can come back to the example from the exposition. The assertion is not a theorem of $\mathcal{SKL}^3$, since the instance $1 = \frac{1}{0} \Rightarrow 1 = 0 \cdot 1$ is not a valid formula (in any reasonable axiomatisation of elementary arithmetic). While the antecedent of the implication evaluates to u, the succedent evaluates to f, hence the whole expression to u.

Example 2 (Extended Example). We will formalise an extended example from elementary algebra that shows the basic features of $\mathcal{SKL}^3$. 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 the formula A2 is taken as an axiom. Naturally, we give only a reduced formalisation of real number arithmetic that is sufficient for our example. Consider the consequent $\{\mathrm{A1}, \mathrm{A2}, \mathrm{A3}, \mathrm{A4}, \mathrm{A5}\} \models \mathrm{T}$ with

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

In an informal mathematical argumentation why T is entailed by $\{\mathrm{A1}, \ldots, \mathrm{A5}\}$, 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). For $x \neq y$ the conclusion $\left(\frac{1}{x-y}\right)^2 > 0$ can be derived from A1 through A5.

3 Tableau

In our tableau calculus, a labeled formula A^α means that A has the truth value α. For the purposes of this paper, it is essential to make use of multi-indices [Häh94] (semantically $A^{\alpha\beta}$ is equivalent to $A^\alpha \vee A^\beta$, however syntactically, on the calculus level it is treated specially.). This not only gives us notational conciseness, but also drastically improves our calculus over a single-index variant, since we can introduce special rules for their treatment. So in general we think of the labels α

as truth value sets, which may be singletons. Note that we normally do not have to treat triple-indices as in A^{fut}, since that would correspond to a three-valued tautology, which cannot contribute to a refutation.

Definition 3 (Tableau Rules). The tableau rules consist of the traditional tableau rules for the propositional connectives, augmented by the case of the label u.

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

The negation rules 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 $\mathcal{SKL}^3$.

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

The last rule is a splitting[iii] rule reflecting the definition of multi-index ft as a disjunction. We only need this one splitting rule, since we have treated the multi-indices ut and fu explicitly in the rules.

The quantifier rules for the classical truth values and multi-indices are very similar to the standard rules ($\{x_S, y^1, \ldots, y^n\}$ 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 (that is there is no instance, which makes the formula false) *and* there is at least one defined witness for the undefinedness.

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

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

[iii] Note that the inverse rule that merges literals A^{α} and A^{β} into a multi-literal $A^{\alpha \cup \beta}$ is not present in the calculus and merging is also not carried out implicitly.

The rules for connectives and quantifiers above can now be used to reduce complex labeled formulae to literals.

Now we only need tableau closure[iv] rules: Undefined definedness literals can be used to close the tableau due to the fact that the predicate Δ is defined everywhere. In the rules *total*, *cut* and *strict*

$$\frac{(t{\leftarrow}\Delta)^{\mathsf{u}\alpha}}{(t{\leftarrow}\Delta)^{\alpha}} \qquad \frac{\begin{array}{c}A^{\alpha}\\ B^{\beta}\end{array}}{A^{\alpha\cap\beta} \mid \mathcal{SC}(\sigma)}\ \sigma \qquad \frac{\begin{array}{c}C^{\gamma}\\ (t{\leftarrow}\Delta)^{\mathsf{f}}\end{array}}{* \mid \mathcal{SC}(\sigma)}\ \sigma$$

we require that $\gamma \subseteq \{\mathsf{ft}\}$ 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. Note that the *cut* rule is only non-redundant if $\alpha \cap \beta$ is a proper subset of both α and β; we will assume this in the following.

In both cases the *sort constraint* $\mathcal{SC}(\sigma) = ((t_1{\leftarrow}S_1) \wedge \ldots \wedge (t_n{\leftarrow}S_n))^{\mathsf{fu}}$ insures the correctness 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.

Definition 4 (Tableau Proof). A *tableau proof for a formula* A is a closed tableau constructed from the initial tree consisting of the labeled formula A^{fu}. A *tableau proof for a consequent* $\Phi \models A$ is a closed tableau constructed from $\Phi^{\mathsf{t}} \cup \{A^{\mathsf{fu}}\}$.

The tableau proof of a consequent $\Phi \models A$ essentially refutes the possibility that A can be undefined or false under the assumption that all formulae in Φ are true. By the quartum non datur rule, we can then conclude that A is entailed by Φ. The soundness of the $\mathcal{TPF}^3$ rules can be verified by a tedious recourse to the semantics of the quantifiers and connectives. Completeness is proved by the standard argument using a model existence theorem for $\mathcal{SKL}^3$. For details see [KK96].

Example 5 (continuing Example 2). 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 Fig. 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 $[c - d/u_{\mathbb{R}}]$, $[c/x_{\mathbb{R}}]$, and $[d/y_{\mathbb{R}}]$ 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) and (F13) have to replace (F3) and (F9) respectively.

[iv] We define that a literal $A^{\emptyset}$ closes a branch of the tableau and denote it with $*$.

The proof in Fig. 1 shows an interesting feature, namely it corresponds in length and structure to a proof of the theorem in a two-valued variant of $\mathcal{TPF}^3$. This observation has a more general background: [Häh94] shows that for so-called regular truth functional logics (all our connectives and quantifiers except for ! are regular), the sets-of-signs method allows a presentation of the tableau system in Smullyan's universal notation, varying only the closure conditions, i.e., the structure is isomorphic to the classical tableau system. However, this result is not directly applicable, since $\mathcal{SKL}^3$ is not truth-functional (we have to consider bounded quantifiers) and we assume strictness.

Definition 6 ($\mathcal{TPF}^2$). Formally this proof system (we will call it $\mathcal{TPF}^2$) can be obtained from $\mathcal{TPF}^3$ by removing all inference rules for the ! connective, the *total* rule and all connective and quantifier rules that contain the label u. This is essentially a tableau variant of [WO90] in the style of [Wei95].

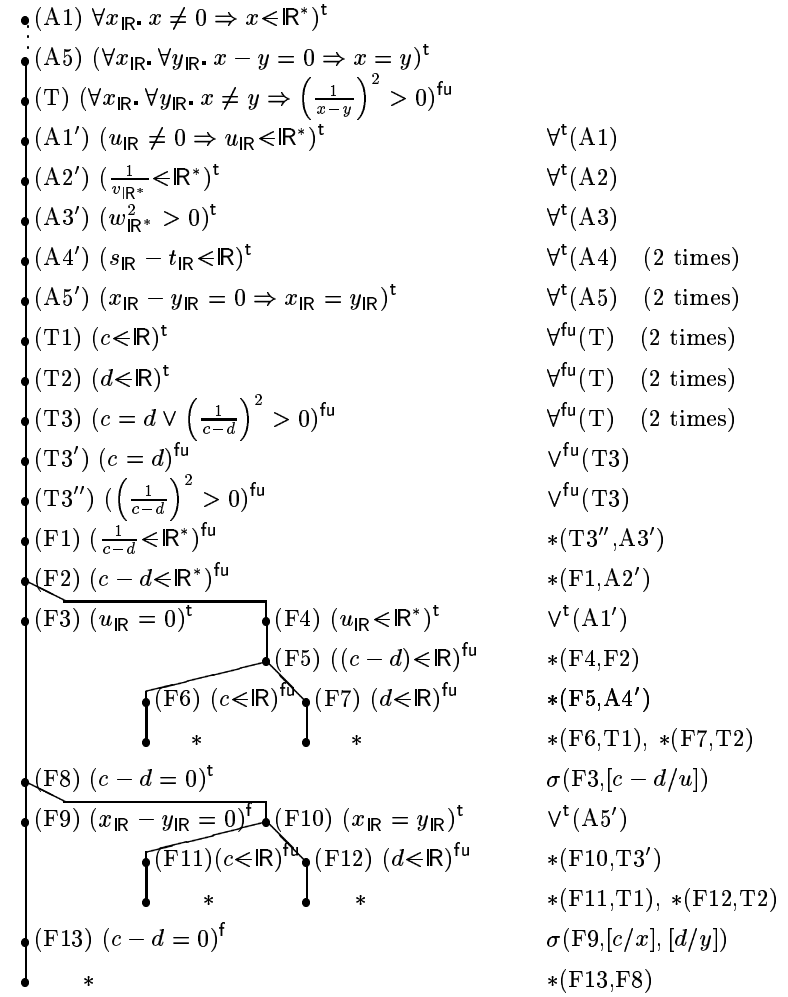

Fig. 1 Tableau proof sketch

The correspondence mentioned above can be realised by replacing all multi-indices fu in $\mathcal{TPF}^3$ by the truth value f in $\mathcal{TPF}^2$. The formal reason that this is possible, lies in the fact that in $\mathcal{TPF}^3$ the tableau rules for R^α and $R^{\alpha\mathsf{u}}$ have exactly the same structure for $\alpha \in \{\mathsf{f}, \mathsf{t}\}$ and $R \in \{\vee, \neg, \forall\}$.

In other words the simple measure of using rules for truth-value sets provides proofs that are 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. Let now $\mathcal{SFL}^2$ be a two-valued sorted logic, that is, the same logic as $\mathcal{SKL}^3$ with two truth values and without the ! connective.

Theorem 7 (Conservativity). *Each $\mathcal{TPF}^3$-tableau proof for a normal problem $\Phi \models A$ in $\mathcal{SKL}^3$ can be transformed into a $\mathcal{TPF}^2$-tableau proof in $\mathcal{SFL}^2$.*

Remark 8. Obviously, the converse of the above theorem does not hold. Not each $\mathcal{TPF}^2$ proof can be transformed into an $\mathcal{TPF}^3$ proof even if there is a $\mathcal{TPF}^3$ proof. Consider for example the relation $\{A\} \models A \vee (B \vee \neg B)$ which holds in $\mathcal{SKL}^3$ as well as in $\mathcal{SFL}^2$. An $\mathcal{TPF}^2$-proof is given in Fig. 2.

(A) $(A)^{\mathsf{t}}$	
(T) $(A \vee (B \vee \neg B))^{\mathsf{f}}$	
(F1) $(A)^{\mathsf{f}}$	$\vee^{\mathsf{f}}$(T)
(F2) $(B \vee \neg B)^{\mathsf{f}}$	$\vee^{\mathsf{f}}$(T)
(F3) $(B)^{\mathsf{f}}$	$\vee^{\mathsf{f}}$(F2)
(F4) $(\neg B)^{\mathsf{f}}$	$\vee^{\mathsf{f}}$(F2)
(F5) $(B)^{\mathsf{t}}$	$\neg^{\mathsf{f}}$(F4)
$*$	$*$(F5,F3)

Fig. 2. Counterexample to the converse conservativity

This proof cannot be transferred since in $\mathcal{SKL}^3$ (T), (F1), (F2), (F3), (F4), and (F5) are labeled by the truth value u in addition, hence applying the closure to (F5) and (F3) leaves a label u in $\mathcal{SKL}^3$ and does not lead to $*$. This comes from the fact that $B \vee \neg B$ is not a tautology in $\mathcal{SKL}^3$. However, the other straightforward closure of the tableau by applying the closure rule to (A) and (F1) can be applied in $\mathcal{SFL}^2$ as well as in $\mathcal{SKL}^3$.

Certainly it would be nice to have the property that for each classical $\mathcal{SFL}^2$ proof there exists an $\mathcal{SKL}^3$ proof which is as short as the classical (of course only if the classical theorem is also an $\mathcal{SKL}^3$ 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 two true statements are linked 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 $\mathcal{SFL}^2$ to $\mathcal{SKL}^3$, the latter two are not theorems in $\mathcal{SKL}^3$. Hence we expect that for usual mathematical theorems the proof effort in $\mathcal{SKL}^3$ will not be bigger then in $\mathcal{SFL}^2$.

If we look again at the counterexample above, we see a general principle, how it is possible to generate a classical proof that cannot be lifted to a $\mathcal{TPF}^3$ proof, essentially closing the tableau by two complementary formulae, which both stem from the theorem. In such a case the branch can be closed on two formulae, labeled t and f in the two-valued setting, but in the three-valued setting they are labeled by tu and uf, so the closure results in a formula labeled by u only.

In the following, we want to give a formal definition of a control strategy for $\mathcal{TPF}^3$ that avoids these pitfalls. For this, we mark theorem nodes with $\mathcal{U}$ and leave assumption nodes unmarked. Marking the nodes generated by the application of a $\mathcal{TPF}^3$ rule Φ is carried out as follows: Add the truth value u to the labels of the premises and apply Φ, then mark the new nodes with $\mathcal{U}$, iff their formulae contain the truth value u. It is a simple exercise to check, that the labeling of a node in a $\mathcal{TPF}^3$ tableau only depends on its origin (i.e., whether it descends from the theorem or not).

Definition 9 ($\mathcal{U}^3$-Strategy). The applicability of all rules except *cut* remains unrestricted by $\mathcal{U}^3$, while the *cut* rule is restricted to the case, where at most one of the parent nodes is in $\mathcal{U}$.

Lemma 10 (Completeness of $\mathcal{U}^3$). *$\mathcal{U}^3$ is a complete strategy for $\mathcal{TPF}^3$ on the normal fragment of $\mathcal{SKL}^3$.*

Note that in $\mathcal{U}^3$-tableaux no node can have the singleton label u, and that hence the connective and quantifier rules for that label are redundant in $\mathcal{TPF}^3$ for the normal fragment of $\mathcal{SKL}^3$.

Since, the $\mathcal{TPF}^3$ and $\mathcal{TPF}^2$ rules are identical in structure, and the marks $\mathcal{U}$ only depend on the origin of formulae, they can also be computed for $\mathcal{TPF}^2$-tableaux, if we leave formulae of the form $t{:}\Delta$ unmarked, irrespective of their origin. This move imitates an application of the *total* rule that does not exist in $\mathcal{TPF}^2$. Let $\mathcal{U}^2$ be that strategy for $\mathcal{TPF}^2$-tableaux that forbids *cut* on formulae marked with $\mathcal{U}$.

Theorem 11 (Lifting). *Each $\mathcal{U}^2$-tableau can be lifted to an isomorphic $\mathcal{U}^3$-tableau.*

Obviously, the strategy $\mathcal{U}^2$ is not complete for $\mathcal{SFL}^2$, and indeed we do not want it to be, since $\mathcal{SKL}^3$ was developed to eliminate formulae from the set of formulae that are provable in $\mathcal{SFL}^2$ and thus in classical first-order logic, but that are generally not considered as *mathematical* theorems (like $1/0 = 0 \vee 1/0 \neq 0$).

Actually, we show the adequacy of $\mathcal{U}^2$-tableau for the normal fragment of $\mathcal{SKL}^3$. To see that $\mathcal{U}^2$ is sound let $\mathcal{T}$ be a closed $\mathcal{U}^2$-tableau for a consequent $\Phi \models A$, then it can be lifted to a closed $\mathcal{U}^3$-tableau by Theorem 11, by the soundness of $\mathcal{TPF}^3$ the consequent must be unsatisfiable. The completeness of $\mathcal{U}^3$-tableau, Lemma 10, for the normal fragment of $\mathcal{SKL}^3$, directly entails the

completeness of $\mathcal{U}^2$ with respect to the three-valued (mathematical) semantics by conservativity, Theorem 7.

Theorem 12. *The $\mathcal{TPF}^2$ calculus with the $\mathcal{U}^2$ restriction strategy is an adequate calculus for the normal fragment of $\mathcal{SKL}^3$.*

Now, we can ask, what is lost by restricting ourselves to the normal fragment of $\mathcal{SKL}^3$. It is not that we cannot specify definedness assumptions for the mathematical objects. This is always possible in the assumption part of consequents, even without an explicit !-connective since $(!A)^{\mathsf{t}}$ is equivalent to $(A \vee \neg A)^{\mathsf{t}}$. In the theorem part, this is not possible, since the presence of the label u blocks the equivalence. Thus it is not possible to prove theorems about the undefinedness of formulae such as $1/0 \not\in \Delta \models \neg !P(1/0)$. Note that we can still prove assertions about the definedness of terms, like in $1/0 \not\in \Delta \models f(1/0) \not\in \Delta$. So in fact $\mathcal{SFL}^2$ with $\mathcal{U}^2$-tableau is a very good approximation of $\mathcal{SKL}^3$.

As discussed in Remark 8 proofs in $\mathcal{SKL}^3$ may be inevitably longer than proofs in $\mathcal{SFL}^2$. This, however, does not mean that short proofs are excluded by the $\mathcal{U}^2$-strategy if they exist in the three-valued calculus, since the truth value set tf in $\mathcal{TPF}^3$ proofs does not occur for the normal fragment of $\mathcal{SKL}^3$.

In the resolution calculus $\mathcal{RPF}^3$ [KK94], we have a similar conservativity and strategy result. One reason for that is that the clause normal form transformations directly correspond to the analytic tableau rules for connectives and quantifiers. For details see [KK97].

4 Conclusion

In this paper we have refuted the common assumption, that a three-valued treatment of partial functions following the ideas of Kleene is impractical, since it requires a fundamental redesign of the current theorem proving technology. This tacit assumption has led to a practical preference of the simpler (but less adequate) two-valued treatment of partial functions. The results in this paper show a simple way towards a practical implementation: In an existing theorem prover for dynamic sorts like SPASS [WGR96], only the strategy $\mathcal{U}^2$ has to be imposed[v].

As we have stated in Remark 8, for most mathematical theorems, this does not even result in a loss of efficiency (proof length). However, the experiment in SPASS should not be regarded as a full-blown implementation of the normal fragment of $\mathcal{SKL}^3$, since the interaction with equality and the interaction of the $\mathcal{U}^3$-strategy with the other strategies (e.g. reduction) has not been addressed in this paper. We leave this problem to future work.

From another perspective, the $\mathcal{U}^3$ strategy can be seen as a substitute (that is easier to implement) for the third truth value, whose presence is adequately represented by marking a two-valued literal by $\mathcal{U}$. Note that this underlines the intuition, that strong Kleene logic is a variant of classical first-order logic that only adds a definedness check for the application of partial functions to the logic without changing the logic proper. In particular, it is plausible that results as those presented in this paper will not in general hold for multi-valued logics.

[v] Christoph Weidenbach has added the necessary extensions to the pure resolution part of SPASS [WGR96] in a matter of a few hours.

Note that it is essential for the theorem prover to be able to treat dynamic sorts, for the conservativity results break down with most relativisation techniques. The only counterexample is the technique of term relativisation [Sti86], where in the case of static tree-ordered sorts a conservativity theorem holds. If this could be extended to dynamic sorts, then any existing theorem prover could (without loss of efficiency) be augmented by partial functions by a term relativisation pre-process and $\mathcal{U}^2$.

References

[BF95] Matthias Baaz and Christian G. Fermüller. Resolution-based theorem proving for many-valued logics. *Journal of Symbolic Computation*, 19(4):353–391, April 1995.

[BFL*94] Juan C. Bicarregui, John S. Fitzgerald, Peter A. Lindsay, Richard Moore, and Brian Ritchie. *Proof in VDM: A Practitioner's Guide*. Springer, London, United Kingdom, 1994.

[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.

[Häh94] Reiner Hähnle. Automated Deduction in Multiple-Valued Logics, Oxford University Press, 1994.

[Jon90] Cliff B. Jones. *Systematic Software Development using VDM*. Prentice Hall, New York, USA, second edition, 1990.

[KK94] Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. In *Proc. CADE-12*, pp. 371–385, 1994. Springer LNAI 814.

[KK96] Manfred Kerber and Michael Kohlhase. A tableau calculus for partial functions. *Collegium Logicum – Annals of the Kurt Gödel Society*, 2:21–49, 1996.

[KK97] Manfred Kerber and Michael Kohlhase. Mechanising Partiality without Re-Implementation. Technical Report CSRP-97-10, School of Computer Science, The University of Birmingham, Birmingham, England, 1997 `ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1997/CSRP-97-10.ps.gz`.

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

[LG89] Francisca Lucio-Carrasco and Antonio Gavilanes-Franco. A first order logic for partial functions. In *Proceedings STACS'89*, pages 47–58. Springer, LNCS 349, 1989.

[Sco70] Dana Scott. Outline of a mathematical theory of computation. In *Proc. Fourth Annual Princeton Conference on Information Sciences and Systems*, pages 169–176. Princeton University, 1970.

[Sti86] Mark E. Stickel Schubert's Steamroller Problem: Formulations and Solutions, *Journal of Automated Reasoning*, 2:89–101, 1986.

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

[WO90] Christoph Weidenbach and Hans Jürgen Ohlbach. A resolution calculus with dynamic sort structures and partial functions. *Proceedings of the 9th ECAI*, pages 688–693, 1990. Pitman.

[Wei95] Christoph Weidenbach. First-order tableaux with sorts. *Journal of the Interest Group in Pure and Applied Logics, IGPL*, 3(6):887–906, 1995.

[WGR96] Christoph Weidenbach, Bernd Gaede and Georg Rock. Spass & Flotter, Version 0.42, In *Proc. CADE-13* pages 141–145, 1996. Springer LNAI 1104.