Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 123 (2005) 151–163
www.elsevier.com/locate/entcs

Living with Paradoxes

Manfred Kerber

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

Abstract

A good knowledge representation system has to find a balance between expressive power on the one hand and efficient reasoning on the other. Furthermore it is necessary to understand its limitations and problems. A logic which contains strings is very expressive and allows for very natural representations, which in turn allow for appropriate reasoning patterns. However, such a system has the feature that it is possible to formulate self-referential paradoxes in it. This can be considered as a strength and as a weakness at the same time. On the one hand it is a positive aspect that it is possible to represent paradoxes, which can be formulated in natural language. On the other hand it is necessary to be careful and not to trivialise the logical system. In the paper different aspects of knowledge representation which allows self-referentiality will be discussed. A system will be presented which is a pragmatic compromise between expressive power on the one hand and simplicity and efficiency of the reasoning process on the other hand. It is built on a three-valued system that makes it possible to use reasoning techniques from classical first-order logic.

Keywords: Strings, paradoxes, three-valued logic

1 Introduction

Logical paradoxes have been known at least back to the Cretan prophet Epimenides who said “All Cretans are liars.” Under some additional assumptions this leads to a paradoxical situation. In its most concise form, the paradox can be formulated by the sentence: “This sentence is false.” Let’s assume it is true, then it must be false, since it says that it is false. Hence it must be false. This leads to a contradiction as well, since if it is false, its content can’t be true, hence the sentence can’t be false either.

1571-0661 © 2005 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2004.04.046

The problems with paradoxical sentences have been known for very long, but were initially thought to be of no relevance for the development of set theory and logic around the end of the 19th/start of the 20th century. But then paradoxes in set theory and logic seemed to endanger the progress in the field and for a while it wasn't clear whether "the paradise of sets" built by Cantor (as Hilbert called it) and "the paradise of logic" built by Frege weren't inhabited by terrible lions that could attack and kill at any time. Zermelo presented in [20] the first axiomatisation of set theory, and Russell [16] in the same year the theory of types. This way it was (apparently) possible to fence the lions out.

While to a certain extend the constructs of axiomatic sets and types may be considered as adequate for mathematical reasoning, there are quite a number of examples in the areas of language understanding and knowledge representation in a more general sense, where we need a more powerful language, where we can't and/or do not want to exclude self-referentiality a priori on mere syntactic grounds. If we did, we would end up with a system which is difficult to use. Actually quite a number of complicated formalisms have been developed and are currently in the main stream of investigations in AI, although some simpler system may do the job better. Perlis argued in [14] that we "can have everything in first-order logic." He investigated a system which is close to the one we propose here, it's first-order logic plus strings.

Our system will be made more formal in the next section. In section 3 we make it clear what we mean by a paradox. Section 4 is devoted to three-valued Kleene logic, the system that will be used to deal with paradoxes. Section 5 describes an efficient form of reasoning for the three-valued approach. Some related work is presented in section 6. Finally a summary is given.

2 Syntactic Theory

Before we introduce a formal syntax for syntactic theory we introduce some examples of expressions we want to be able to formulate. Then we fix a syntax, and discuss problems.

Let us look now at some typical examples of pieces of knowledge that can be represented and reasoned with in syntactic theory:

(i) John knows that the morning star is the evening star.
$\mathbf{K}(\mathsf{John}, \text{“evening\_star = morning\_star”})$

(ii) This sentence is false. $\mathbf{L} :\equiv \neg\mathsf{True}(\text{“}\mathbf{L}\text{”})$

(iii) If this proposition is true, then $\mathbf{A}$. $\mathbf{L\ddot{o}b} :\equiv (\mathsf{True}(\text{“}\mathbf{L\ddot{o}b}\text{”}) \rightarrow \mathbf{A})$

The relationship between modal logic and syntactic theory corresponds to indirect vs. direct speech. Davis stresses that direct speech is more powerful, since in particular non-syntactic expressions can be expressed directly, e.g., ("John said 'Arglebargle glumph' ", formally **said**(John, "Arglebargle glumph"), but not indirectly. A similar property holds for modal logic versus syntactic theory. Konolige [11, p.114] mentions two main advantages of syntactic theory over hierarchical approaches to knowledge representation, firstly "a lessening of the notational burden associated with hierarchical languages" and "that self-referential languages are more expressive than hierarchical languages, and this difference might be important in representing belief."

In this contribution, we shall see in more detail, a three-valued approach to deal with paradoxes; formulae like the liar sentence are admitted, but they are neither true nor false, but paradoxical. In order to do so, a third truth value, nn (standing for "neither-nor") is added. As we shall see, just adding such a truth value does not solve the problem, but the system has to fulfil an additional constraint. However, before we go into a semantic analysis, let's first fix the syntax and then clarify what we mean by a paradox.

The language we want to investigate is a standard first-order sorted language plus strings as terms. In addition to the standard setting, we use strings in the language. To this end we take characters as constant symbols in the language. Following the description in [3] we prefix characters by a colon, e.g., $:C$ is the character C, $:\wedge$ the character standing for the conjunction connective. Furthermore we assume a binary function symbol, pair, on strings, which takes a character and a string as input. From the empty string ε we build up all strings, e.g., $\mathsf{pair}(:C, \mathsf{pair}(:a, \mathsf{pair}(:t, \varepsilon)))$. Since these expressions play a major role, we introduce as syntactic sugar for expressions like the one above the notation "*Cat*".

It is possible to define and use *syntactic* predicate symbols. Variable("x") and Objconstant("John") denote, for instance, that x and John are a variable and an object constant, respectively. Details of such a construction can be found in [5, Chapter 10]. Sorts stand for unary predicates. That is, a term t is of sort S (written as $t{<}S$) iff $S(t)$. Sorted quantification like $\forall x_S \centerdot \mathbf{A}$ semantically means that the x's are not taken from the whole universe of discourse, but only from the subset of those elements which are in the interpretation of S. The sort system adopted here has been introduced by Weidenbach [19].

The so-defined language is reflective. For instance, it is possible to define the syntax of the language within the language. Although it is intended that strings and objects they stand for are closely related, we must carefully distinguish between them. There is an intuitive difference between long(John) and long("John") where long is a polymorphic predicate symbol that is true for persons over 195cm and strings consisting of at least eight characters.

In addition to the syntactic function and predicate symbols there are *semantic* predicate symbols such as True. We shall discuss in particular this predicate symbol in detail in the sequel. For this special predicate symbol we will also have to give a special semantic account (as one does for first-order logic plus equality, where the equality sign gets a fixed interpretation).

Traditionally at this point the semantics of the language is discussed. In our case this is not so straightforward and the meaning of syntactic theory is one of the major problems with this language. Hence we shall first try to get an intuitive idea of the semantics, then we discuss in more detail how such a system can be build up.

3 Paradoxes

The existence of problematic expressions in syntactic theory has been known back to the start of the 20th century. If we assume a standard first-order semantics there is no problem with the liar sentence, since the strings like "**L**" and "**A**" are ordinary ground terms in the language and have nothing to do with their counterparts **L** and **A**. However, as such they are not particularly meaningful and do not capture the intended meaning. Of course we like the predicate True to mean truth. Tarski [17] gave a famous definition of truth:

$$\mathsf{True}(\text{“}\mathbf{A}\text{”}) \equiv \mathbf{A}$$

as axiom schema for arbitrary formulae **A**. If we add this schema the liar example is turned into a paradox, since we get: $\mathbf{L} \equiv \neg\mathsf{True}(\text{“}\mathbf{L}\text{”}) \equiv \neg\mathbf{L}$.

The last example, sentence, **Löb**, is not a paradox in a strict sense, but an example that violates the conservativity of definitions. It can be stated as "If this proposition is true, then **A**." where **A** is an arbitrary formula. Formally we get $\mathbf{L\ddot{o}b} :\equiv \mathsf{True}(\text{“}\mathbf{L\ddot{o}b}\text{”}) \rightarrow \mathbf{A}$. With the definition of truth this simplifies to $\mathbf{L\ddot{o}b} \equiv (\mathbf{L\ddot{o}b} \rightarrow \mathbf{A})$. This formula is problematic in a two-valued setting since it allows to prove **A** for arbitrary **A**. The only consistent way to assign truth values is one, in which **A** must be true. That is, defining **Löb** potentially changes the semantic status of sentences.

Of course it is possible to doubt whether these examples are relevant and do really occur; also whether there is a syntactic way to exclude them. Kripke [12] stated the so-called Watergate Paradox by summarising different propositions made by a journalist Jones and by the former US president Nixon about the Watergate affair. Nixon says "Everything Jones says about Watergate is true." and Jones says "Most of Nixon's assertions about Watergate are false." If we assume in addition that Nixon made $2n + 1$ assertions about Watergate altogether of which n are definitively true, n definitively false, plus the one above, we have reproduced the paradox of the liar.

This example suggests that it is not possible to decide just on syntactic grounds whether sentences make up a paradox or not, since this may depend on the real world context. If Nixon, for instance, always told the truth except for the statement about Jones, there wouldn't be a paradox, the two statements would simply be false. There seems not to be a clearcut syntactic borderline of what should be admissible and what not.

We will understand by a paradox, a definition which cannot be evaluated to the truth value true. A *definition* is a sentence which introduces a new concept, that is, a constant symbol, function symbol, or predicate symbol, which has not been in the signature. A property we usually expect from a definition is that it forms a *conservative extension*, that is, that the semantic status of old concepts remains unchanged by the introduction of the new concept.

4 Kleene Logic for Syntactic Theory

In this section three-valued Kleene logic [10] is extended in order to deal with syntactic expressions. As we shall see this allows us to treat the liar sentence adequately, but cannot be yet quite the final system, since not all problems with paradoxes can be avoided that way. The treatment in the following corresponds roughly to the development of the mechanisation of Kleene logic developed by Kerber and Kohlhase [8,9]. In the first of these contributions a tableau calculus for sorted three-valued Kleene logic is described, in the second a way is shown how to extend a classical two-valued theorem prover by a simple restriction strategy in a way that it can be used as a theorem prover for an interesting fragment of Kleene logic. I give here only a short summary of the results, for details see the corresponding papers.

In addition to the treatment of Kleene logic in the papers mentioned above, in the following we have to deal with strings (a simple conservative extension as discussed above) and additionally the axiom schema for defining the semantics of the True predicate.

4.1 Syntax and Semantics

As above we assume terms to be variables (labelled by a sort), constant symbols, or functional expressions. In particular strings are terms. Atomic formulae are either k-ary predicate symbols applied to k terms or sort expressions of the form $t \lessdot S$. Formulae in general are built by the connectives and quantifiers as $\neg A$, $\mathbf{D}A$, $A \vee B$, $A \equiv B$, $\forall x_S\centerdot A$.

The semantics is based on three truth values, false, nn, and true. The meaning of the connectives can be defined by the truth tables

¬	
false	true
nn	nn
true	false

D	
false	true
nn	false
true	true

∧	false	nn	true
false	false	false	false
nn	false	nn	nn
true	false	nn	true

$\equiv$ is true if and only if the input truth values are the same and false else. Other connectives $\vee$, $\rightarrow$, and $\leftrightarrow$ can be defined in $\wedge$ and $\neg$ just as in classical two-valued logic. The semantics of the universal quantifier is defined by

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

$$\widetilde{\forall}(T) := \begin{cases} \mathsf{true} & \text{for } T = \{\mathsf{true}\} \text{ or } T = \emptyset \\ \mathsf{nn} & \text{for } T = \{\mathsf{true}, \mathsf{nn}\} \text{ or } T = \{\mathsf{nn}\} \\ \mathsf{false} & \text{for } \mathsf{false} \in T \end{cases}$$

The semantics of the True predicate is defined as

$$\mathcal{I}(\mathsf{True}(x)) = \begin{cases} \mathcal{I}(\mathbf{A}) & \text{if } x = \text{“}\mathbf{A}\text{” and } \mathbf{A} \text{ is a wff} \\ \mathsf{nn} & \text{else} \end{cases}$$

Remark 4.1 Note that this is (purposefully) not a proper definition of the True predicate, since it doesn't unambiguously fixes the meaning of True. Let us assume, for instance, a knowledge base $\Gamma := \{\mathbf{T}, \mathbf{T} \equiv \mathsf{True}(\text{“}\mathbf{T}\text{”})\}$. Nothing can be derived if someone protests his sincerity. The sentence has three fixpoints, false, nn, and true. In the semantics we fix the meaning of True just to the extent that it mirrors Tarski's definition of truth and build a calculus which exactly corresponds to this. If we assume this semantics it is possible to consistently assign truth values to the liar sentence, namely with $\mathcal{I}(\mathbf{L}) = \mathsf{nn}$ the whole equivalence is true (and hence not causing any trouble).

4.2 Calculus and Properties

In the following a refutation tableau calculus for three-valued Kleene logic is presented, the details can be found in [8]. Let $\mathbf{A}$ be a formula, then $\mathbf{A}^\alpha$ is a labelled formula if $\emptyset \neq \alpha \subseteq \{\mathsf{false}, \mathsf{nn}, \mathsf{true}\}$.

In order to show that a formula $\mathbf{B}$ follows from a formula set $\{\mathbf{A}_1, \ldots, \mathbf{A}_n\}$ we can prove that the set of labelled formulae $\{\mathbf{A}_1^{\mathsf{true}}, \ldots, \mathbf{A}_n^{\mathsf{true}}, \mathbf{B}^{\mathsf{nn},\mathsf{false}}\}$ is unsatisfiable.[1] That is, it is shown that the assumption that the $\mathbf{A}_i$ hold and $\mathbf{B}$ is nn or false is inconsistent (“quartum non datur”-principle).

[1] The formulae in the formula set are implicitly conjuncts. Formulae like $\mathbf{B}^{\mathsf{nn},\mathsf{false}}$ stand for $\mathbf{B}$ is nn or $\mathbf{B}$ is false.

Tableau Rules: Tableau rules for $\wedge$ and $\forall$ are:

$$\frac{(\mathbf{A}\wedge\mathbf{B})^{\mathsf{true}}}{\begin{array}{c}\mathbf{A}^{\mathsf{true}}\\ \mathbf{B}^{\mathsf{true}}\end{array}} \qquad \frac{(\mathbf{A}\wedge\mathbf{B})^{\mathsf{nn}}}{\begin{array}{c}\mathbf{A}^{\mathsf{nn,true}}\\ \mathbf{B}^{\mathsf{nn,true}}\\ \mathbf{A}^{\mathsf{nn}} \mid \mathbf{B}^{\mathsf{nn}}\end{array}} \qquad \frac{(\mathbf{A}\wedge\mathbf{B})^{\mathsf{false}}}{\mathbf{A}^{\mathsf{false}} \mid \mathbf{B}^{\mathsf{false}}}$$

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

Now we only need tableau closure rules: These are a *cut* rule, which allows to close a tableau which contains formulae with complementary labels and the *strict* rule which allows to close a tableau if a defined expression contains an undefined term. Where possible we would like to make use of most general unifiers, of course. In the case of string expressions, these may not be so easy to compute, however, and a unification algorithm would have to make use of techniques developed in higher-order unification. In both cases so-called *sort constraints* insure the correctness (in terms of the sorts) of the instantiations. We quote the soundness and completeness theorem from [8]:

Theorem 4.2 (Soundness and Completeness)
The tableau calculus is sound and complete for Kleene logic.

Remark 4.3 In order to ensure soundness and completeness for the syntactic theory add axioms that exactly capture the meaning of the semantic predicates, i.e., in the case of True add $\mathsf{True}(\text{“}\mathbf{A}\text{”}) \equiv \mathbf{A}$ either as axiom schema or as simplification rules (plus the reverse rules):

$$\frac{(\mathsf{True}(\text{“}\mathbf{A}\text{”}))^{\alpha}}{\mathbf{A}^{\alpha}} \text{ if } \mathbf{A} \text{ wwf} \qquad \frac{(\mathsf{True}(\text{“}\mathbf{A}\text{”}))^{\alpha}}{(\mathsf{True}(\text{“}\mathbf{A}\text{”}))^{\alpha\cap\{\mathsf{nn}\}}} \text{ else}$$

4.3 Problems

Now let us take a closer look whether the three-valued system is adequate for dealing with paradoxes. We have already seen that it is adequate for treating the liar paradox. But can it do so for all paradoxes?

When we look at the sentence “This sentence is paradoxical or false”, we can define it in this system as $\mathbf{P} :\equiv \neg\mathbf{D}\mathsf{True}(\text{“}\mathbf{P}\text{”}) \vee \neg\mathsf{True}(\text{“}\mathbf{P}\text{”})$. With the definition of truth we can simplify the expression to $\mathbf{P} \equiv \neg\mathbf{DP} \vee \neg\mathbf{P}$. If we look at the possible truth values for $\mathbf{P}$, that is, false, nn, and true, we get an evaluation, which shows that $\mathbf{P}$ cannot be consistently assigned to any of the three truth values that are available. That means, while we have tamed some paradoxes like the liar sentence by assigning a third truth value to them, some are still wild beasts and cause mayhem in form of contradictions.

Neither the attempt to remedy the problems with paradoxes by many-valued logics nor the problems of an ad hoc solution by introducing a further truth value are new. In 1939 Bochvar made such an attempt, but as Church pointed out the same year, the problems can be easily reconstructed in the three-valued setting again (see [18, p.75]). Actually, adding more and more truth values is a solution to the problem, but there is no intuitive meaning of these values any more.

4.4 How to Avoid Self-Referential Paradoxes

The source of any paradoxes of self-referentiality relies on the fact that it is possible to define a formula to which no truth value can be assigned. This can be done in the propositional logic case if and only if it is possible to define a function in the connectives that is *fixpoint free* on the truth values. In a two-valued system such a function is easily defined, it is just negation ($\lambda x_{\blacksquare} \neg x$), that is, as soon as negation is part of a self-referential two-valued system, paradoxes are not far away, or to put it the other way around: in a two-valued logic without negation (explicit or implicit) it is not possible to construct paradoxes. To give up negation in a knowledge representation system is, of course, a serious restriction of the expressive power.

In our three-valued setting above $\lambda x_{\blacksquare} \neg \mathbf{D} x \vee \neg x$ and $\lambda x_{\blacksquare} (x \equiv \neg x) \equiv x$ are fixpoint-free as well. If we apply one of these functions to any of the three truth values false, nn, or true, the result will never be the same as the input of the function. From this the paradoxes are easily constructed.

However, if we restrict the language to a system, in which each function that can be built from the connectives has at least one fixpoint (conveniently, nn often plays that role), it is not possible to construct self-referential paradoxes. This restriction seems not to be a very serious one, since it is rarely desired to communicate paradoxicality. If one really liked to do so one would have to go to a four-valued setting (which would have paradoxes of a higher kind in itself, of course).

In a three-valued setting, no language that contains the connectives $\neg$, $\vee$, and $\mathbf{D}$ and no language with unrestricted use of $\neg$ and $\equiv$ guarantees the existence of fixpoints. But $\neg$ has nn as fixpoint, $\lambda x_{\blacksquare} x \vee x$ has the same fixpoint nn as well (and in addition the fixpoints false and true), and the fixpoint property is conserved under composition.

However, this restriction would not prevent the construction of more subtle paradoxes, in which we construct paradoxes by defining a binary function f (using only $\neg$ and $\vee$) in the truth values such that $x := f(x, y)$ for a fixed truth value y. This would cause a problem if $x \neq f(x, y)$ for all possible truth values x. In that case y can't be nn, since then $x = \mathsf{nn}$ would be a fixpoint. So let us assume that $y \neq \mathsf{nn}$. Is it possible to define a function f in $\neg$ and $\vee$ such that $f(x, y) \neq x$ for $x = \mathsf{true}$, $x = \mathsf{nn}$, *and* $x = \mathsf{false}$? This is not the case, since when restricted to false and true, f has also range $\{\mathsf{false}, \mathsf{true}\}$. In order to be fixpoint free on $\{\mathsf{false}, \mathsf{true}\}$, $f(x, y)$ must be $\neg x$, since this is the only fixpoint free function of all possible 16 such functions. Hence it will have the fixpoint nn on the full set $\{\mathsf{false}, \mathsf{nn}, \mathsf{true}\}$. (Typically there are more fixpoints than just nn.)

Theorem 4.4 *Any function built from the connectives $\neg$ and $\vee$ contains a fixpoint.*

The fixpoint property has as corollary that definitions form conservative extensions. Let us now have a look why universally quantified formulae like $\mathbf{A} \equiv \forall x_\centerdot \mathbf{B}$ have a fixpoint. This is potentially problematic if x is a string which allows an instantiation with an expression that contains $\mathbf{A}$. If $\mathbf{B}$ can be made false for one instantiation of x, $\mathbf{A}$ is false. If $\mathbf{B}$ can be made true for all instantiations, then $\mathbf{A}$ is true. In other cases (it is not possible to instantiate x to get a false value, but it is not always true), we can select nn as fixpoint. This possibility is given since the propositional system of connectives is *not* complete.

In order to be able to introduce definitions (like the definition of the liar sentence, but of course of more useful sentences as well), the $\equiv$ connective cannot be abandoned altogether, but has to be allowed in definitions, that is, in the form $\mathbf{A} \equiv \ldots$ or $\forall x_\centerdot \mathbf{A}(x) \equiv \ldots$, where $\mathbf{A}$ is a predicate constant. This does not cause problems, since the right hand side will have a fixpoint of $\mathbf{A}$ if $\mathbf{A}$ occurs in it, hence the equivalence as such can always be made true.

5 Consequence Relation and Reasoning

When we abandon the $\mathbf{D}$ connective, we lose most tautologies in the language and the question arises how we can restore a non-trivial system when there are no proper tautologies in it. This is achieved by considering sequents, that is, pairs consisting of a set of formulae Φ and a formula $\mathbf{A}$ (written $\Phi \vdash \mathbf{A}$), in which all formulae in Φ are assumed to be true. The semantic equivalent is the model relation $\Phi \models \mathbf{A}$ that stands for: in all models of Φ, that is, all interpretations that evaluate every formula in Φ to true, $\mathbf{A}$ holds as well, that is, $\mathbf{A}$ is evaluated to true as well. A proof theory that is built on the refutation principle makes use of the fact that all formulae in Φ must be true and refutes the assumptions that $\mathbf{A}$ might be false or nn. Note the asymmetry between Φ and $\mathbf{A}$ in this approach, formulae in Φ can't be nn, while $\mathbf{A}$ might be.

Since we use a restricted language without a $\mathbf{D}$ connective it is possible to make use of the strategy for reusing two-valued theorem proving methods in dealing with Kleene logic as described in [9]. That means, efficient standard two-valued theorem provers can be made to efficient theorem provers for the approach above just by adding a simple restriction strategy.

This forms a calculus for first-order Kleene logic in general. In order to handle quoted expressions, we have to add axiom schemas that describe the semantics of the meta-predicates defined on them. In the case of the truth predicate True, which we have looked at more closely, this means we add the rules from remark 4.3, or corresponding axiom schemas.

The main result from [9] is that it is possible to use the structural similarity between the tableau rules for truth value sets containing nn and those not containing nn. It is possible to just eliminate nn in these rules. Since it is possible to avoid rules for the truth value nn altogether by using combined rules (for $\mathsf{false}, \mathsf{nn}$ and $\mathsf{nn}, \mathsf{true}$), it is possible to establish the following strategy result, which allows to employ standard first-order theorem proving techniques for Kleene logic as well.

Theorem 5.1 (Strategy) *Let $\mathcal{S}$ be the control strategy not to close a tableau on two formulae both stemming from the theorem, then each two-valued tableau proof found using $\mathcal{S}$ can be lifted to a three-valued tableau proof for the theorem.*

Remark 5.2 The result from [9] leaves us with the question how to treat the $\equiv$ connective efficiently. We could use tableau rules for $\equiv$. However, we make only restricted use of the $\equiv$ connective, namely for definitions ("definiendum $\equiv$ definiens"). For that reason only the first of the three tableau rules ($(\mathbf{A} \equiv \mathbf{B})^{\mathsf{true}}$) is relevant at all. But even this rule splits the tableau into three tableaux, hence it would be rather inefficient. Even more seriously, the strategy result mentioned above could not be conserved. Instead of using this rule, we simply assume that in proofs all definitions can be expanded, that is, we take a definition expansion rule into the calculus (for arbitrary truth value sets α).

$$\frac{\begin{array}{c}\mathbf{A}^{\alpha}\\ (\mathbf{A} \equiv \mathbf{B})^{\mathsf{true}}\end{array}}{\mathbf{B}^{\alpha}}$$

This rule (as well as the rule for handling $\mathsf{True}(\text{“}\mathbf{A}\text{”})$ if we assume $\mathbf{A}$ to be a well-formed formula) fulfils the conditions of the strategy theorem. Hence the corresponding theorem carries over to our case as well.

6 Related Work

There is an abundance of literature on paradoxes, the reasons for their existence as well as on ways how to deal with them. It would be easy to fill books just with an overview and it will be impossible to give a fair account here.

The first solution to treat paradoxes goes back to Russell and it is as easy as brilliant, namely simply to exclude them. Russell introduced types such that in any expression of the form $P(Q)$, P has to have a higher type than Q. In particular P can't express anything about itself or about anything that can make statements about P. This approach has been adapted by Tarski [17] to hierarchical meta-systems, in which it is possible to speak explicitly about the truth of statements, but not in the language itself, but in some meta-language. If one wants to make statements about the meta-language, this is possible as well, but then one needs a meta-meta-language. This approach has been formalised and implemented in the multi-level meta-systems FOL and Getfol [6]. While this approach is appropriate for mathematics, examples like the Watergate paradox point to the conclusion that excluding self-referentiality altogether is not appropriate for (at least) some applications.

An alternative to totally forbidding self-reference is restricting it, putting it in the right context. This approach is discussed by Barwise and Etchemendy in detail in [1] as the second approach, which they call the Austinian approach. The idea is to restrict self-referential expressions in an adequate way. Applied to Russell's example of the barber this means, the definition of the barber as "the man who shaves all and only the men who don't shave themselves" is problematic. But, for instance, to define the barber as "the man who shaves all and only the men who live in Oxford and who don't shave themselves" means only that the barber does not live in Oxford. While I agree that many situations which seem to be paradoxical on the first view can be clarified so that the paradox goes away, certain sentences are deliberately paradoxical.

Kripke [12] and others attacked the problem by changing the semantic construction of the truth values of formulae. They go beyond two-valued logic in a way that they assume a third truth value, which stands for the paradoxical expressions. There are different ways, either to assume truth value gaps [4] or a third truth value, which states undefinedness explicitly [10, p.332–340]. In both systems, Tarski's original definition of truth can be incorporated, that is, $\mathsf{True}(\text{“}\mathbf{A}\text{”}) \equiv \mathbf{A}$ and the liar sentence, $\mathbf{L} :\equiv \neg\mathsf{True}(\text{“}\mathbf{L}\text{”})$ can be dealt with as well. In the first system with truth value gaps, no truth value is assigned to the liar sentence $\mathbf{L}$. In the second, $\mathbf{L}$ is evaluated to the truth value nn, in particular $\mathsf{True}(\text{“}\mathbf{L}\text{”})$ is nn as well. The question arises how to determine the truth values in such expressions. To this end Kripke proposes a fixpoint iteration, a process in which in each iteration truth values are assigned to more and more formulae. Kripke's approach is an appropriate general semantic account. Such a general approach is necessary for a general approach. It has been made more formal and in particular it has been given a semantic interpretation by Barwise and Etchemendy [1]. They use Aczel's theory of hypersets (sets which do not fulfil the foundation axiom) to model self-referentiality. A more detailed description is given in [2]. A similar approach has been developed in parallel by McGee [13]. In Perlis' approach [14] Tarski's definition of truth is changed to a system in which $\mathbf{L}$ and $\neg\mathsf{True}(\text{“}\mathbf{L}\text{”})$ hold at the same time.

The approach presented in this paper can be viewed as a restricted version of what Barwise and Etchemendy call the Russellian approach. It builds up on earlier work by the author [7], and is closely related to Ramsay's approach [15], which is built up on a constructive λ-calculus.

7 Summary

We have seen in this contribution a three-valued approach to allow self-referential sentences in a formal system almost without any restrictions. The only restriction that we have to impose on such a system, is to guarantee that any definition gives us a fixpoint. For predicative definitions this can be guaranteed by definitions which make use only of the standard connectives, while excluding connectives which would allow to speak about paradoxicality. On the first view this seems to trivialise the system, since by this restriction we have eliminated almost all tautologies from the system. However, when we assume a background theory, with respect to which reasoning takes place, this is not a serious restriction at all. All formulae in the background theory are assumed to be true (this is equivalent to saying, they are not false and not paradoxical). If we assume a paradoxical formula in the background theory as true, the system becomes inconsistent (and not paradoxical), just as a classical system becomes inconsistent by assuming a formula and its negation. Interestingly the restriction on the connectives allows to transfer standard two-valued theorem proving techniques to the three-valued case.

Summarising, we have presented a system for stating facts about truth. By adding further axiom schemas it is possible to extend the framework to deal with knowledge and belief as well. No unnatural restrictions on the expressive power are made in order to avoid self-referential statements. As shown in section 5 standard calculi can be used. In this work we have not produced a formal fixpoint construction for the True predicate, but only given an indication why such fixpoints exist. Building on the work of Kripke, Barwise, Etchemendy, and Moss this should be possible, but is left as future work.

References

[1] Jon Barwise and John Etchemendy. *The Liar – An Essay on Truth and Circularity.* Oxford University Press, New York, USA, 1987.

[2] Jon Barwise and Lawrence Moss. *Vicious Circles.* CSLI Publications, Stanford, California, USA, 1996.

[3] Ernest Davis. *Representations of Commonsense Knowledge.* Morgan Kaufmann, San Mateo, California, USA, 1990.

[4] Bas C. van Fraassen. Singular terms, truth-value gaps, and free logic. *The Journal of Philosophy*, LXIII(17):481–495, 1966.

[5] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, San Mateo, California, USA, 1987.

[6] Fausto Giunchiglia. Getfol manual – Getfol version 2.0. IRST-Technical Report 9107-01, IRST, Trento, Italy, 1994.

[7] Manfred Kerber. On knowledge, strings, and paradoxes. In J. Dix, L. Fariñas del Cerro, and U. Furbach, editors, *Proceedings of JELIA'98, Logics in Artificial Intelligence*, pages 342–354. Springer Verlag, Berlin, 1998. LNAI 1489.

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

[9] Manfred Kerber and Michael Kohlhase. Mechanising partiality without re-implementation. In G. Brewka, C. Habel, and B. Nebel, editors, *Proceedings of KI'97*, pages 123–134, Freiburg, 1997. Springer Verlag, Berlin, LNAI 1303.

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

[11] Kurt Konolige. *A Deduction Model of Belief.* Pitman, London, 1986.

[12] Saul Kripke. Outline of a theory of truth. *The Journal of Philosophy*, LXXII:690–716, 1975.

[13] Vann McGee. *Truth, Vagueness, and Paradox – An Essay on the Logic of Truth.* Hackett Publishing Company, Indianapolis, USA, 1990.

[14] Donald Perlis. Languages with self-reference I: Foundations (or: We can have everything in first-order logic!). *AI*, **25**:301–322, 1985.

[15] Allan M. Ramsay. Theorem proving for untyped constructive λ-calculus: Implementation and application. *Logic Journal of the IGPL*, **9**(1):83–100, 2001.

[16] Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, XXX:222–262, 1908.

[17] Alfred Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia philosophia*, **1**:261–405, 1936.

[18] Alasdair Urquhart. Many-valued logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, chapter III.2, pages 71–116. Reidel, Dordrecht, The Netherlands, 1986. Volume III: Alternatives to Classical Logic.

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

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