CHECKING PROOFS IN THE METAMATHEMATICS OF FIRST ORDER LOGIC

Mario Aiello
Istituto di Scienze dell 'Informazione, Universita di Pisa,
Corso Italia 40, 56100 Pisa Italy

and
Richard Weyhrauch
A.T. Lab., Computer Science Department, Stanford University,
Stanford, California 94305 U.S.A.

Abstract

First order theories not only can be used in proving properties of programs, but have also relevance in representation theory. The desire to represent first order theories in a computer in a feasible way requires the facility to discuss meta mathematical notions. Using metamathematics will eventually allow to construct systems which can formally discuss how they reason. In this paper we present two different first order axiomatizations of the metamathematics of the logic which POL (First Order Logic proof checker) checks and show several proofs using each one. The difference between the axiomatizations is that one defines the metamathematics in a many sorted logic and the other does not. Proofs are then compared and used to discuss the adequacy of some FOL features.

Section 1 Introduction

This paper represents a first attempt at axiomatizing the metamathematics of a first order

(First Order Logic). The logic whieh FOL checks is described in detail in the user manual for this program, Weyhrauch and Thomas 1974. It is based on a system of natural deduction described in Prawitz 1965, 1970.

Our motivation in axiomatizing the metamathematics of FOL was the desire to work on an example which could he used as *a* case study for projected features of FOL and, at the same time, had independent interest with respect to representing the proofs of significant mathematical results to a computer.

The eventual ability to clearly express the theorems of mathematies to a computer will require the *facility* to state and prove theorems of metamathematics. There are several clear examples:

a. Axiom schemas. How exactly do we express that

$$P(0) \land \forall n.(P(n) \supset P(n+1)) \supset \forall n.P(n)$$

is an axiom schema? We need to say: "if for any first order sentence *V* with one free variable y we denote by P(n) the formula obtained from P by substituting n for y assuming n is free for y in P, then the sentence

$$P(0) \land \forall n.(P(n) \supset P(n+1)) \supset \forall n.P(n)$$

is an axiom of arithmetic."

b. Theorem schemas. The following kind of "theorem" is sometimes seen in set theory books

$$\forall x1...xn \exists S. \exists T. \forall u. (<x1,...,xn> \in T \equiv$$
$$\exists y.(<x1,...,xn,y> \in S)).$$

It asserts the existence of some particular projection of n+1-tuples. In its usual formulation this is not a theorem of set theory at all, but a metatheorem which states that, for each n, the above sentence is a theorem. We do not know of any machine implementation of first order logic capable of expressing the above notion in a straightforward way.

c. Subsidiary deduction rules. Below we show how to prove that if there is a proof of Vx y. WFF then there is also a proof of Vy x. WFF, where WFF is any well formed formula. We chose this task because it seemed simple enough to do, and is a theorem which may actually be used. The use of metatheorems as rules of inference by means of a reflection principle will be discussed in a future memo by Richard Weyhrauch. Eventually we hope to check some more substantial metamathematical theorems.

d. Interesting mathematical theorems. We present two examples. The first is any theorem about finite groups. The notion of finite group cannot be defined in the usual first order language of group theory. Thus many "theorems" are actually metatheorems, unless you axiomatize groups in set theory. The second theorem is the "duality principle" in projective geometry.

Finally, from the viewpoint of A.I. and representation theory the ability to state and prove theorems of metamathematies can be very helpful in answering the questions of how we "reflect" on the reasoning we are doing and if a proper axiomatization of the metamathematics of an FOL language together with some sort of computationally realizable reflection principle allows us to discuss in an adequate way our reasonings.

This paper is divided into two sections. In the first one, we present the two axiom systems and the proof of the metatheorem: "for all variables x,y and well formed formulas f, "$\forall x\ y.f$" is a theorem also "$\forall y\ x.f$" a theorem". In the second section we look at proofs appearing in the appendices in order to explore the features of FOL that need improving and their use in earry ing out formal proofs.

* This research was carried out while the author was visiting at the A.I. Lab., Computer Science Department, Stanford University.

Section 2 The Axiom System

In this HOClion we present two axiomatiza-Uons of the mctamalhcmatics of first order logic. The IIUJi11 difference between Lhcm is that one is done iu a many sorted first order logic and the other not. These axioinilizalions represent an attempt at experimenting with proots about properties of formulas and deductions. No effort has been spent on guaranteeing that the axioms are independent. It would not only have been uninteresting but also contrary to our basic philosophy. We wish to find axioms which naturally reflect the relevant notions. At the moment this axiomatization is far from being in its final form. Neither the extent of the notions involved nor the best way of expressing them is considered settled.

Strings and sequences of strings have been axiotnatized and used to define metamathematical notions. For instance, well formed formulas are represented as strings of symbols which satisfy the predicate' FORM defining which combinations of constants, variables predicates and functions symbols represent a wff.; deductions are then represented as sequences of wff's satisfying the predicate PROOFTKKK.

2.1 The Sorts

The sorts we have defined correspond to the basic notions ol the metamathematics i.e. terms, formulas, individual variables, logical symbols, function symbols etc. and the notions of the domains (strings and sequences of strings) in which the axiotnatization has been defined. KOI, (see Weyhrauch and Thornas 1974) allows the declaration of variables to be ot a certain sort. In the formulas appearing in this paper, each variable is declared to be of some particular sort. For instance f,fl,f2,.,. are of sort well formed formulas t,tl,t2... of sort terms etc. For the complete set of FOL declarations see Aiello and Wcyhrauch 1974.

2.2 The Domain of Representation of the metamathemat ics.

The basic notions of the metamathematics of first order logic have been axiomatized in terms of strings and sequences of strings. The primitive functions on them are concatenation (c for strings, cc for sequences) and selectors (car, cdr for strings and scar, scdr for sequences), c and cc are infix operators.

2.2.1 Formulas and terms

Formulas and terms arc represented by the string of symbols appearing in them. Terms are defined recursively as strings which either represent an individual variable or can be decomposed into n+l substrings representing a function symbol of arity n, followed by n terms. The two predicates defining terms arc:

```
TERMSEQ(Ø,LAMBDA)
∀s.(TERM(s) INDVAR(s)∨ ∃n fn.(fn=car(s)∧
        n=arity(fn) ∧ TERMSEQ(n,cdr(s))))

∀n s.(TERMSEQ(n,s) ((car(s)=LPARSYM)∧((len(s)gt s)=
        RPARSYM∧ ∃nl.(TERM(substring(s,2,nl))∧
        TERMSEQ(n-1,substring(s,nl+1,len(s)-1)))))
```

where the function substring(s,n,n) returns the substring of s starting from its m-th element and ending with the n-th. len(s) computes the length of s and (n gP s))selecls the n-th element of s.

Well formed formulas (wffs) are represented as strings which either are elementary formulas (defined by the predicate KM-') or can be partition^ ed into substrings for formulas and logical connectives. Formulas are defined by:

```
∀s.(ELF(s) (s=FALSESYM ∨ PREDPARØ(s) ∨ ∃n P.(P=
   =car(s) ∧ n=arity(P)∧ TERMSEQ(n,cdr(s))))),

∀s.(FORM(s)_(ELF(s)∨ ∃x f.(s=(x gen f) ∨ s=
   =(x ex f))∨ ∃fl f2.(s=(fl dis f2)∨ s=
   =(fl con f2) ∨ s=(fl impl f2)) ∨ ∃f.s=neg(f)))
```

gen is the infix operator that maps its arguments x and f into the string "(FOKALLSYM c x) c f" representing the well formed formula Vx.f. The oper ator ex is used for the existential quantifier, dis, con and impl are the infix operators for the disjunction, conjunction and implication of two formulas,Finally neg is the operator which maps a formula into its negation.

We could possibly represent wffs as structured objects (lists, trees, etc.) which contain all the information about the structure of the formula and do not require any parsing. This approach amounts to axiomatizing metamathematics in terms of the abstract syntax of first order logic, instead of strings of symbols. Both of these possibilities should be explored. We have chosen the first alternative because:

1) It is the most traditional, i.e. luetanuith ematics, as it appears in logic books, is usually stated in terms of strings.

2) Axioms in terras of abstract syntax are simply theorems of the theory expressed in terms of strings. Thus the two representations look sub stantially the same with respect to "high level" theorems.

3) Ill-formed formulas can be mentioned. This is of course impossible in an axiomatization in terms of the abstract syntax.

The properties of wffs relevant to our theory have been defined by the predicates FR, FRN, GEB and SBT, FR(x,f) is true iff the variable x has at least one free occurrence in the wff f, while FRN(x,n,f) and CEB(x,n,f) are respectively true when the variable x occurs free or bound at the place n in the formula f. In addition to these predicates, some generalized selector functions are defined, which evaluate the first or the k-th free occurrence of a variable in a wff, or the number of its free occurrences. The predicate SBT is then defined. It axiotnatizes the notion of sub stitution of a term for any free occurrence of a variable in a wff.

```
∀x t fl f2.(SBT(x,t,fl,f2)≡ ∀nl n2.((n2=
        =(numbfreeocc(x,nl,fl)*(len(t)-1))+nl) ⊃
        ((¬INDVAR(nl gl fl) ⊃ (nl gl fl)=
        =(n2 gl f2))∧(INDVAR(nl gl fl) ⊃
        ((FRN(x,nl,fl)⊃SUBT(t,f2,n2))∧
        (¬FRN(x,nl,fl) ⊃ INVART(nl,fl,n2,f2)))) )))
```

∀t f2 n2.(SUBT(t,f2,n2) ∀x2 k.((k gℓ t)=x2 ⊃
FRN(x2,n2-(len(t)-k),f2)))),

∀n f1 n1 f2.(INVART(n,f1,n1,f2) ((GEB(n1 gℓ f2,
n1,f2) GEB(n gℓ f1,n,f1))∧(FRN(n1 gℓ f2,n1,f2)≡
FRN(n gℓ f1,n,f1))∧(n1 gℓ f2)=(n gℓ f1)))

In the previous definition, nl is any position in the string fl and n2 is the corresponding position in f2. The axiliary predicate SHUT states that the variables appearing in the term t substituted for a tree occurrence of the variable x are still free. INVART defines which properties of fl are still true for 12. If the term t is a variable, then SBT reduces to SBV:

∀x1 x2 f1 f2.(SBV(x1,x2,f1,f2)≡∀n.((¬INDVAR(
n gℓ f1)⊃(n gℓ f1)=(n gℓ f2)∧(INDVAR(n gℓ f1)⊃
((FRN(x1,n,f1))⊃FRN(x2,n,f2))∧(¬FRN(x1,n,f1)⊃
INVARV(n,f1,f2)))))),

∀n f1 f2.(INVARV(n,f1,f2)≡((GEB(n gℓ f2,n,f2) GEB
(n gℓ f1,n,f1))∧(FRN(n gℓ f2,n,f2) FRN(n gℓ f1,
n,f1))∧(n gℓ f2)=(n gℓ f1))).

The proof of the equivalence of SBT and SBV when t is a variable is very simple. It is based on the fact that n2 coincides with nl when the term t has length 1 (see Aiello and Weyhrauch 1974), The function sbt (sbv) evaluates to the siring representing the result of substituting a term (variable) for every free occurrence of a variable in a gi ven wff. sbt and sbv are defined from the predicates SBT and SBV as follows:

∀x t f1 f2.(SBT(x,t,f1,f2) sbt(x,t,f1)=f2)

∀x1 x2 f1 f2.(SBV(x1,x2,f1,f2) sbv(x1,x2,f1)=f2)

The problem of finding the best way of defining functions in FOL is crucial: in the axiom system given in this paper a uniform way has not been followed. In defining the substitution we are interested in properties of the functions sbt and sbv and in drawing conclusions from the fact that a substitution has been made. It is thus useful to have a predicate which defines the relation between formulas before and after a substitution instead of inferring it from the definition of the functions (stated for example as a system of equations, as in Kleene 1952). One of the motivationT of the present experiment was to explore different ways of defining functions. We do not yet have enough examples of proofs to make a clear statement about this matter.

2.2.2 Rules ot inference, deductions and the notion of provability

The rules of inference with one premise, are expressed by means of a binary predicate whose arguments are two sequences of wffs (sq, pf) which satisfy PROOFTREE.
The predicate is true iff pf is the scdr of sq and the first element of sq is a wff obtained by applying that particular deduction rule to the first wff of pf. The rules with more antecedents are defined in a similar way.

Derivations are recursively defined as sequences of wffs which either are a single wff or are obtained from one or more derivations by applying one of the deduction rules. The recursion is implicitly stated by saying that there exist objects of sort PKOOKTKKK which satisfy one of the predicates defining the rules of inference. These sequences represent the linearization of a deduction-tree and are defined as follows:

∀sq.(PROOFTREE(sq) (FORM(sq)∨ ∃pf.(ORI(sq,pf)∨
ANDE(sq,pf)∨FALSEE(sq,pf) ∨NOTI(sq,pf)∨
NOTE(sq,pf)∨IMPLI(sq,pf))∨ ∃pf x t.(GENI(sq,pf,
x,t)∨GENE(sq,pf,x,t)∨EXI(sq,pf,x,t))∨ ∃pf1
pf2.(ANDI(sq,pf1,pf2)∨ FALSEI(sq,pf1,pf2)∨
IMPLE(sq,pf1,pf2))∨ ∃pf1 pf2 x1 x2.EXE(sq,pf1,
pf2,x1,x2)∨ ∃pf1 pf2 pf3.ORE(sq,pf1,pf2,pf3)))

A sequence of wffs is a prooftree it either it consists of a single wff or one of the following alternatives holds: there exists another prooftree and a one premise deduction rule has been applied; there exist two prooftrees and one of the two premises rules has been applied; finally, there are three prooftrees and the predicate defining the V-elimination rule is true. Note that the root of a prooftree is not necessarily a theorem in a given theory. A predicate DEPEND has been defined which is true if a given wff is a dependence for the root of a prooftree. The axioms about DEPEND allow to decide all the dependencies of a prooftree.

Since some of the deduction rules (the implication introduction, for instance) eliminate dependencies, not all the leaves of a prooftree pf are dependencies for a wff f such that f-scar(pf). The predicate DEPEND is true only for those leaves of the prooftree which the formula f actually depends on. The axioms DEPEND state which dependencies do not change by applying the deduction rules and are transferred from one prooftree to the other. The axioms NDEPEND state which rules dicharge dependentcies in a given prooftree.

Using this notion of dependence, the provability of a formula in a theory is defined as follows:

∀f.(BEW(f)≡ ∃sq.(PROOFTREE(sq)∧ f=scar(sq)∧
∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))

A wff.f is a theorem in a given theory if there exists a prooftree whose first element is f and whose only dependencies are axioms in that theory. We have limited our attention to theories in which axioms have no free variables. This property is defined by the axiom: ∀ x f.(AXIOM(f)⊃ FK(x.f)).

2.3 The Main Proof in the "any Sorted Logic

The main theorem we have proved in this axiomatization of the metamathematics states that if ∀x f. wff is provable in some theory, then ∀ y x. wff is also provable. We have chosen this theorem because, even if very simple, it involves basic notions of provability, substitution and universal quantification. Its proof is found in appendix 2. The theorem depends on the first three lines of the proof. The first step is a lemma stating that ∀x wff.sbt(x,x,wff)=wff, i.e. substituting a variable x for any free occourrence of x in wff doesn't change that wff. Steps 2 and 3 give simple facts about sequences. The theorem is then proved by instantiating two other lemmas: a) if ∀x.wff is a theorem, then wff is also a theorem; b) if wff is provable, then x cannot be free in the dependencies of the proof of wff and so ∀x.wff is provable. This is of course true only for teories with no free variables in their axioms.

2 (x gen f)=(x gen f)∨(x gen f)=(x ex f)

3 ∃x1 f1.((x gen f)=(x1 gen f)∨(x gen f)=(x1 ex f1))

4 FORM(x gen f)

1.2 Printout of the oroof in the second axiomatization

1 FORM(f)∧INDVAR(x1)

2 ∀sq s.((SEQUENCE(sq)∧sq≠SLAMBDA)⊃(STRING(s)⊃(s cc sq)≠SLAMBDA)) (2)

3 ∀s sq.((STRING(s)∧SEQUENCE(sq)). scar(s cc sq)=s) (3)

4 ∀s sq.((STRING(s)∧SEQUENCE(sq)). scdr(s cc sq)=sq) (4)

5 ∀sq.((SEQUENCE(sq)∧sq≠SLAMDA)≡find(1,scar(sq),sq)) (5)

6 ∀f x.((FORM(f)∧INDVAR(x))⊃STRING(x gen f)) (6)

7 ∀s sq.((STRING(s)∧SEQUENCE(sq))⊃SEQUENCE(s cc sq)) (7)

8 ∀x.(INDVAR(x)⊃STRING(x) (8)

9 PORM(f)≡(STRING(f)∧∃sq.(FRR(sq)∧f=scar(sq)))

10 ∃sq.(FRR(sq)∧f=scar(sq)) (1 2 3 4 5 6 7 8)

11 FRR(SQ)∧f=scar(SQ) (11)

12 FRR(SQ)≡(SEQUENCE(SQ)∧(SQ≠SLAMBDA∧(ELF(scar(SQ))∨(FRR(scdr(SQ))∧∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar(SQ)=NEG(s1)∧find(1,s1,scdr(SQ)))∨((scar(SQ)=(s1 dis s2)∧find(2,s1 c x2,scdr(SQ)))∨((scar(SQ)=(s1 con s2)∧find(2,s1 c s2,scdr(SQ)))∨((scar(SQ)=(s1 impl s2)∧find(2,s1 c s2,scdr(SQ)))∨((scar(SQ)=(s1 gen s2)∧(INDVAR(s1)∧find(1,s2,scdr(SQ))))∨(scar(SQ)=(s1 ex s2)∧(INDVAR(s1)∧find(1,s2,scdr(SQ))))))))))))))

13 (SEQUENCE(SQ)∧SQ≠SLAMBDA)⊃(STRING(x1 gen f)⊃((x1 gen f) cc SQ)≠SLAMBDA) (2)

14 (STRING(x1 gen f)∧SEQUENCE(SQ))≡scar((x1 gen f) cc SQ)(x1 gen f) (3)

15 (STRING(x1 gen f)∧SEQUENCE(SQ))≡scdr((x1 gen f) cc SQ)=SQ (4)

16 (SEQUENCE(SQ)∧SQ≠SLAMBDA)≡find(1,scar(SQ),SQ) (5)

17 (FORM(f)∧INDVAR(x1))⊃string(x1 gen f) (6)

18 INDVAR(x1)⊃STRING(x1) (8)

19 FRR(x1 gen f) cc SQ)≡(SEQUENCE((x1 gen f) cc SQ)∧(((x1 gen f) cc U)≠SLAMBDA∧(ELF(scar((x1 gen f) cc SQ))∨(FRR(scdr((x1 gen f) cc SQ))∧∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar((s1 gen f) cc SQ)=NEG(s1)∧find(1,s1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 dis s2)∧find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 con s2)∧find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 con s2)∧find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 impl s2)∧find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 gen s2)∧(INDVAR(s1)∧find(1,s2,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(s1 ex s2)∧(INDVAR(s1)∧find(1,s2,scdr((x1 gen f) cc SQ)))))))))))))))

20 STRING(x1)∧(STRING(f)∧((scar((x1 gen f) cc SQ)=NEG(x1)∧find(1,x1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 dis f)∧find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 con f)∧find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 impl f)∧find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 gen f)∧(INDVAR(x1)∧find(1,f,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(x1 ex f)∧(INDVAR(x1)∧find(1,f,scdr((x1 gen f) cc SQ))))))))) (1 2 3 4 5 6 7 8 11)

21 ∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar((s1 gen f) cc SQ)=NEG(s1)∧find(1,s1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 dis s2)∧find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 con s2)∧find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 impl s2)∧find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f)cc SQ)=(s1 gen s2)∧(INDVAR(s1)∧find(1,s2,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(s1 ex s2)∧(INDVAR(s1)∧find(1,s2,scdr((x1 gen f) cc SQ)))))))))))))) (1 2 3 4 5 6 7 8 11)

22 (STRING(x1 gen)∧SEQUENCE(SQ))⊃SEQUENCE((x1 gen f) cc SQ) (7)

23 FORM(x1 gen f) (STRING(x1 gen f)∧∃sq.(FRR(sq)∧(x1 gen f)=scar(sq)))

24 FRR(x1 gen f) cc SQ)∧(x1 gen f)=scar((x1 gen f) cc SQ) (1 2 3 4 5 6 7 8 11) TAUTEQ I:23

25 ∃sq.(FRR(sq)∧(x1 gen f)=scar(sq)) (1 2 3 4 5 6 7 8 11)

26 FORM(x1 gen f) (1 2 3 4 5 6 7 8 11)

27 FORM(x1 gen f) (1 2 3 4 5 6 7 8)

28 (FORM(f)∧INDVAR(x1))⊃FORM(x1 gen f) (2 3 4 5 6 7 8)

29 ∀f x1.((FORM(f)∧INDVAR(x1))⊃FORM(x1 gen f)) (2 3 4 5 6 7 8)

<u>Appendix 2</u>

The proof that universal quantifiers can be interchanged Printout of the proof in the many sorted logic

1 ∀x f.sbt(x,x,f)=f (1)

2 ∀f sq.scar(f cc sq)=f (2)

3 ∀f sq.scdr(f cc sq)=sq (3)

4 BEW(x gen f) (4)

5 sbt(x,x,f)=f (1)

6 BEW(x gen f)≡∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))

7 ∃sq.(PROFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)))) (4)

8 PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))) (8)

9 GENE(f cc sq,sq,x,x)≡(scdr(f cc sq)=sq∧(PROOFTREE(sq)∧∃f.(scar(sq)=(x gen f1)∧scar(f cc sq)=sbt(x,x,f1))))

10 scar(f cc sq)=f (2)

11 scdr(f cc sq)=sq (3)

12 scar(sq)=(x gen f)∧scar(f cc sq)=sbt(x,x,f) (1 2 3 4 8)

13 ∃f1.(scar(sq)=(x gen f1)∧scar(f cc sq)=sbt(x,

```
   x,f1)) (1 2 3 4 8)
14 GENE(f cc sq,sq,x,x) (1 2 3 4 8)
15 PROOFTREE(f cc sq)≡(FORM(f cc sq)∨(∃pf.(ORI(f
   cc sq,pf)∨(ANDE(f cc sq,pf)∨(FALSEE(f cc sq,pf)
   ∨(NOTI(f cc sq,pf)∨(NOTE(f cc sq,pf)∨IMPLI(f
   cc sq,pf))))))∨(∃pf x t.(GENI(f cc sq,pf,x,t)
   ∨(GENE(f cc sq,pf,x,t)∨EXI(f cc sq,pf,t)))∨(∃
   pf1 pf2.(ANDI(f cc sq,pf1,pf2)∨(FALSEI(f cc sq,
   pf1,pf2)∨IMPLE(f cc sq,pf1,pf2)))∨∃pf1 pf2 x
   t.EXE(f cc sq,pf1,pf2,x,t)∨∃pf1 pf2 pf3.ORE(f
   cc sq, pf1,pf2,pf3))))))
16 GENI(f cc sq,sq,x,x)∨(GENE(f cc sq,sq,x,x)∨
   EXI(f cc sq,sq,x,x)) (1 2 3 4 8)
17 ∃pf x t.(GENI(f cc sq,pf,x,t)∨(GENE(f cc sq,pf,
   x,t)∨EXI(f cc sq,pf,t))) (1 2 3 4 8)
18 PROOFTREE(f cc sq) (1 2 3 4 8)
19 ∀ f1.(DEPEND(sq,f1)⊃AXIOM(f1)) (8)
20 DEPEND(sq,f1)⊃AXIOM(f1) (8)
21 PROOFTREE(f cc sq)⊃(PROOFTREE(sq)⊃((sq=scdr(f
   cc sq)⊃(DEPEND(f cc sq,f1)≡DEPEND(sq,f1))≡(
   ORI(f cc sq,sq)∨(ANDE(f cc sq,sq)∨(FALSEE(f cc
   sq,sq)∨(∃f.((NOTID(f cc sq,sq,f)∨(NOTED(f cc
   sq,sq,f)∨IMPLID(f cc sq,sq,f)))∧f≠f1)∨∃x t.(
   GENI(f cc sq,sq,x,t)∨(GENE(f cc sq,sq,x,t)∨
   EXI(f cc sq,sq,x,t)))))))))
22 ∃x t.(GENI(f cc sq,sq,x,t)∨(GENE(f cc sq,sq,
   x,t)∨EXI(f cc sq,sq,x,t))) (1 2 3 4 8)
23 DEPEND(f cc sq,f1)⊃AXIOM(f1) (1 2 3 4 8)
24 ∀f1.(DEPEND(f cc sq,f1)⊃AXIOM(f1)) (1 2 3 4 8)
25 f=scar(f cc sq) (2)
26 PROOFTREE(f cc sq)∧(f=scar(f cc sq)∧∀f1.(
   DEPEND(f cc sq,f1)⊃AXIOM(f1))) (1 2 3 4 8)
27 BEW(f)≡∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(
   DEPEND(sq,f1)⊃AXIOM(f1))))
28 ∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,
   f1)⊃AXIOM(f1)))) (1 2 3 4)
29 BEW(f) (1 2 3 4)
30 BEW(x gen f)⊃BEW(f) (1 2 3)
31 BEW(f) (31)
32 ∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,
   f1)⊃AXIOM(f1)))) (31)
33 PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃
   AXIOM(f1))) (33)
34 ∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)) (33)
35 DEPEND(sq,f1)⊃AXIOM(f1) (33)
36 APGENI(x,sq)≡(∀f.(DEPEND(sq,f)⊃¬FR(x,f))∧
   PROOFTREE(sq))
37 AXIOM(f1)⊃¬FR(x,f1)
38 DEPEND(sq,f1)⊃¬FR(x,f1) (31 33)
39 ∀f1.(DEPEND(sq,f1)⊃¬FR(x,f1)) (31 33)
40 APGENI(x,sq) (31 33)
41 GENI((x gen f) cc sq,sq,x,x)≡(scdr(x gen f) cc
   sq)=sq∧(PROOFTREE(sq)∧∃f1.(scar((x gen f) cc
   sq)=(x gen f1)∧(scar(sq)=sbt(x,x,f1)∧APGENI(x,
   sq)))))
42 scar((x gen f) cc sq)=(x gen f) (2)
43 scdr((x gen f) cc sq)=sq (3)
44 scar((x gen f) cc sq)=(x gen f)∧(scar(sq)=sbt
   (x,x,f)∧APGENI(x,sq)) (1 2 3 31 33)
45 ∃f1.(scar((x gen f) cc sq)=(x gen f1)∧(scar(sq)
   =sbt(x,x,f1)∧APGENI(x,sq))) (1 2 3 31 33)
46 GENI((x gen f) cc sq,sq,x,x) (1 2 3 31 33)
47 PROOFTREE((x gen f) cc sq)≡(FORM((x gen f) cc
   sq)∨(∃pf.(ORI((x gen f) cc sq,pf)∨(ANDE((x gen
   f) cc sq,pf)∨(FALSEE((x gen f) cc sq,pf)∨(NOTI
   ((x gen f) cc sq,pf)∨(NOTE((x gen f) cc sq,pf)
   ∨IMPLI((x gen f) cc sq,pf))))))∨(∃pf x1 t.(GENI
   ((x gen f) cc sq,pf,x1,t)∨(GENE((x gen f) cc
   sq,pf,x1,t)∨EXI((x gen f) cc sq,pf,x1,t)))∨
   (∃pf1 pf2.(ANDI((x gen f) cc sq,pf1,pf2)∨(
   FALSEI((x gen f) cc sq,pf1,pf2)∨IMPLE((x gen f)
   cc sq,pf1,pf2)))∨(∃pf1 pf2 x1 t.EXE((x gen f)
   cc sq,pf1,pf2,x1,t)∨∃pf1 pf2 pf3.ORE((x gen f)
   cc sq,pf1,pf2,pf3))))))
48 GENI((x gen f)cc sq,sq,x,x)∨(GENE((x gen f) cc
   sq,sq,x,x)∨EXI((x gen f) cc sq,sq,x,x)) (1 2
   3 31 33)
49 ∃pf x1 t.(GENI((x gen f) cc sq,pf,x1,t)∨(GENE
   ((x gen f) cc sq,pf,x1,t)∨EXI((x gen f) cc sq,
   pf,x1,t))) (1 2 3 31 33)
50 PROOFTREE((x gen f) cc sq) (1 2 3 31 33)
51 PROOFTREE(x gen f) cc sq)⊃(PROOFTREE(sq)⊃((sq=
   scdr((x gen f) cc sq)⊃(DEPEND((x gen f) cc sq,
   f1) (DEPEND(sq,f1))) (ORI((x gen f) cc sq,sq)∨
   (ANDE((x gen f) cc sq,sq)∨(FALSEE((x gen f) cc
   sq,sq)∨(∃f.((NOTID((x gen f) cc sq,sq,f)∨(NOTED
   ((x gen f) cc sq,sq,f)∨IMPLID((x gen f) cc sq,
   sq,f)))∧f≠f1)∨∃x1 t.(GENI((x gen f) cc sq,sq,
   x1,t)∨(GENE((x gen f) cc sq,sq,x1,t)∨EXI(x gen
   f) cc sq,sq,x1,t)))))))))
52 ∃x1 t.(GENI((x gen f) cc sq,sq,x1,t)∨(GENE((x
   gen f) cc sq,sq,x1,t)∨EXI((x gen f) cc sq,sq,
   x1,t))) (1 2 3 31 33)
53 DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1) (1 2 3
   31 33)
54 ∀f1.(DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1))
   (1 2 3 31 33)
55 (x gen f)=scar((x gen f) cc sq) (2)
56 PROOFTREE((x gen f) cc sq)∧((x gen f)=scar((x
   gen f) cc sq)∧∀f1.(DEPEND((x gen f) cc sq,f1)⊃
   AXIOM(f1))) (1 2 3 31 33)
57 BEW(x gen f)≡∃sq.(PROOFTREE(sq)∧((x gen f)=
   scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))
58 ∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(
   DEPEND(sq,f1)⊃AXIOM(f1)))) (1 2 3 31)
59 BEW(x gen f) (1 2 3 31)
60 BEW(f)⊃BEW(x gen f) (1 2 3)
61 BEW(x gen f)≡BEW(f) (1 2 3)
62 ∀x f.(BEW(x gen f)≡BEW(f)) (1 2 3)
63 BEW(x1 gen(x2 gen f))≡BEW(x2 gen f) (1 2 3)
64 BEW(x2 gen f)≡BEW(f) (1 2 3)
65 BEW (x1 gen f)≡BEW(f) (1 2 3)
66 BEW(x2 gen (x1 gen f))≡BEW(x1 gen f) (1 2 3)
67 BEW(x1 gen (x2 gen f))⊃BEW(x2 gen (x1 gen f))
   (1 2 3)
68 ∀x1 x2 f.(BEW(x1 gen (x2 gen f))⊃BEW(x2 gen (x1
   gen f))) (1 2 3)
```

Appendix 3

The axioms for formulas in the unsorted logic

```
AXIOM FIND:
  ∀sq.(FIND(Ø,LAMBDA,sq)≡SEQUENCE(sq)),
  ∀n s sq.(FIND(n,s,sq)≡INTEGER(n)∧STRING(s)∧
    SEQUENCE(sq)∧∃n s1 s2.(INTEGER(n)∧STRING(s1)∧
    STRING(s2)∧(Ø<s∧s<slen(sq))∧(s1=(n sgl sq))∧
    (s=(s1 c s2))∧FIND(n-1,s2,sq)));;

AXIOM FINDTOP:
  ∀sq.(FINDTOP(Ø,SLAMBDA,sq)≡SEQUENCE(sq)),∀n s
    sq.(FINDTOP(n,s,sq)≡INTEGER(n)∧STRING(s)∧
    SEQUENCE(sq)∧∃s1 s2.(STRING(s1)∧STRING(s2)∧
    (s1≠LAMBDA)∧(s=(s1 c s2))∧(s=scar(sq))∧
    FINDTOP(n-1,s2,scar(sq))));;

AXIOM TERM:
  ∀sq.(TERMSEQ(sq)≡SEQUENCE(sq)∧((slen(sq)=1∧
    INDVAR(1 sgl sq))∨(slen(sq)>1∧TERMSEQ(scdr
    (sq))∧(INDVAR(scar(sq))∨∃n s.INTEGER(n)∧
    STRING(s)∧(s=car(scar(sq))∧OPCONST(s)∧n=arity
    (s)∧FIND(n,cdr(scar(sq)),scdr(sq)))))),
  ∀t.(TERM(t)≡STRING(t)∧∃sq.(TERMSEQ(sq)∧t=car(
    sq)));;

AXIOM WFF:
  ∀f.(ELF(f)≡STRING(f)∧(f=FALSESYM∨PREDPAR0(f)∨
    ∃n sq.(INTEGER(n)∧SEQUENCE(sq)∧PREDPAR(car
    (f))∧n=arity(car(f))∧TERMSEQ(sq)∧FINDTOP(n,
    cdr(f),sq)))),
  ∀sq.(FRR(sq)≡SEQUENCE(sq)∧(sq≠SLAMBDA)∧(ELF(
    scar(sq))∨(FRR(scdr(sq))∧∃s1 s2.(STRING(s1)∧
    STRING(s2)∧(((scar(sq)=neg(s1)∧FIND(1,x1,
    scdr(sq)))∨(scar(sq)=(s1 dis s2)∧FIND(2,(s1 c
    s2),scdr(sq)))∨(scar(sq)=(s1 con s2)∧FIND(2,
    (s1 c s2),scdr(sq)))∨(scar(sq)=(s1 impl s2)∧
    FIND(2,(s1 c s2),scdr(sq)))∨(scar(sq)=(s1 gen
    s2)∧INDVAR(s1)∧FIND(1,s2,scdr(sq)))∨(scar(sq)
    =(s1 ex s2)∧INDVAR(s1)∧FIND(1,s2,scdr(sq))))
    )))));
  ∀f.(FORM(f)≡STRING(f)∧∃sq.(FRR(sq)∧f=scar(sq)))
```

References

Prawitz, D,
1965 Natural deduction, A proof theoretical study
Alaquist and Wikseil, Stockholm (1965).

Godel, K.,
1930 Die Voilatandingk.it der Axione dea logischen Funktionenkslkule Monatshefte fur Matheoatik und Physik 37, (1930) 349-360.

Godel, K.,
1931 Uber formal unentacheidbare Satse der Principle mathematics und vervandter Systems I, Monatahefte fur Matheoatik und Physik 38,(1931) 173-198.

Veyhrauch, R.W., and Thomas, A.J.,
1974 POL: A Proof Checker for First-order Logic,Stanford Artificial Intelligence Laboratory, Memo AIM-235 (1974).

Aiello, M., and Weyhrauch, R.W.,
1974 Checking Proofs in the metaaetheaatics of first order logic, Stanford Artificial Intelligence Laboratory, Memo AIM-222 (1974).