A THEOREM PROVER FOR META THEORY

BY

Frank M. Brown

TR-85 Oct. 1978

Department of Computer Sciences, The University of Texas at Austin

A Theorem Prover for Meta theory

Frank Brown
Dept. of Computer Science
The University of Texas at Austin
Austin, Texas 78712

ABSTRACT

We describe an automatic theorem prover for meta theory which is capable of proving the completeness of quantificational logic from intuitively true assumptions.

I. INTRODUCTION

This is a report of some of our research carried out mainly during the spring of 1978. It describes an implementation of a deductive system for meta theory which is capable of proving the completeness of quantificational logic. This deductive system is similar to earlier theorem provers developed by the author [1,2,3]. This meta-theory is based on a modal logic and a theory of meaning which have recently been developed by the author [4,5,6]. The meta theory and its deductive system are described in section 2. Finally, in section 3 we discuss an example theorem which the system has proven, namely the completeness of quantificational logic.

2. DESCRIPTION OF THE THEOREM PROVER

Our theorem prover consists of an interpreter for mathematical expressions and many items of mathematical knowledge. This interpreter is a fairly complex mechanism, but it may be viewed as applying items of mathematical knowledge of the form: $\phi \leftrightarrow \psi$ or $\phi = \psi$ to the theorem being proven, in the following manner. The interpreter evaluates the theorem recursively in a call-by-need manner. That is, if $(f a_1 \ldots a_n)$ is a sub-expression being evaluated, then the interpreter tries to apply its items of knowledge to that sub-expression before evaluating the arguments $a_1 \ldots a_n$. For each sub-expression that the interpreter evaluates, in turn it tries to match the ϕ expression of an item to that sub-expression. If, however, during the application process an argument a_i does not match the corresponding argument of the ϕ expression, then a_i is evaluated, and the system then tries to match the result of that evaluation. If ever the interpreter finds a sub-expression $\phi\theta$ which is an instance of ϕ of some item, then it replaces that expression by the corresponding instance $\psi\theta$ of ψ. At this point all memory of the sub-expression $\phi\theta$ is immediately lost and the interpreter now evaluates $\psi\theta$. If no items can be applied to a sub-expression then the sub-expression is not evaluated again but is simply returned.

Sometimes it will be the case that our interpreter will need to use items which are valid only in certain domains Π. In such a case we could represent the item as a conditional item of the form:

$$\Pi x \rightarrow (\phi x \leftrightarrow \psi x)$$
or
$$\Pi x \rightarrow (\phi x = \psi x)$$

The interpreter handles conditional items in the same way in which it handles non-conditional items until it has found a $\phi\theta$ which matches the sub-expression being evaluated. At this point on a conditional item, the interpreter tries to match each element in the conjunction Πx with some expression which it believes to be true. If such matches are found with substitution $\theta\sigma$ then $\psi\theta\sigma$ is returned. Otherwise the interpreter tries to apply another item as previously described.

However, in this theorem prover we will not bother to represent such items as conditional items, but simply use items of the form $\phi x \leftrightarrow \psi x$ and $\phi x = \psi x$. We will trust the theorem prover itself not to confuse the various domains and in particular not to instantiate any variable x to some term representing an entity in the wrong domain.

The rationale behind this trust is that if the domains are reasonably disjoint then since the theorem prover can only instantiate variables by a process of matching expressions, each variable can only be instantiated to something of the right domain.

There is certainly some sort of moral here and we think it is this: Don't worry about the details, just get on with the proof. We are not here arguing that this is a deep theoretical issue (although perhaps it is) but this approach to domain dependent items is advantageous in that it allows the theorem prover to obtain the proofs more quickly, and more importantly, the proofs obtained are shorter.

It may be argued that proofs obtained by such a theorem prover are worthless, since it would be very difficult to prove that a theorem prover using this method is sound, but as we have argued in [1], the real issue is not whether the theorem prover is sound but merely whether any particular proof is correct or not.

2.1 Logical Knowledge

Our theorem prover has knowledge about twelve logical symbols which are listed below with their English translations:

$\wedge$	and
$\vee$	or
$\sim$	not
■	true
□	false
$\rightarrow$	implies
$\leftrightarrow$	iff
$\exists$	there exists
$\forall$	for all
$=$	equal
$\rightarrow$	implies (This symbol is called a sequent arrow)

and   and (This symbol is used to form an implicit conjunction of sequents)

The sequent arrow may be defined as follows:
$p_1,\ldots,p_n \rightarrow q_1,\ldots,q_m =_{df} (p_1 \wedge \ldots \wedge p_n) \rightarrow (q_1 \vee \ldots \vee q_m)$
where p_i and q_j are sentences. Thus a sequent may be thought of as being a database of statements $p_1,\ldots, p_n$ called assertions which occur before the sequent arrow, and statements $q_1,\ldots,q_m$ called goals which occur after the sequent arrow. The implicit conjunction of different sequents may be thought of as being a group of different databases.

The items of logical knowledge, which are all schemata because they involve ellipses (i.e. dots representing arbitrary expressions), are listed below:

2.1.1 Assertion schemata:

$\blacksquare\rightarrow$: $(..\ \blacksquare\ ..\rightarrow ..) \leftrightarrow (..\ ..\rightarrow ..)$
$\square\rightarrow$: $(..\ \square\ ..\rightarrow ..) \leftrightarrow \blacksquare$
$\sim\rightarrow$: $(..\ \sim p\ ..\rightarrow ..) \leftrightarrow (..\ ..\rightarrow p\ ..)$
$\wedge\rightarrow$: $(..\ p\wedge q\ ..\rightarrow ..) \leftrightarrow (..\ p,q\ ..\rightarrow ..)$
$\vee\rightarrow$: $(..\ p\vee q\ ..\rightarrow ..) \leftrightarrow (..\ p\ ..\rightarrow ..)$ and $(..\ q\ ..\rightarrow ..)$
$\supset\rightarrow$: $(..\ p\rightarrow q\ ..\rightarrow ..) \leftrightarrow (..\ ..\rightarrow p\ ..)$ and $(..\ q\ ..\rightarrow ..)$
$\leftrightarrow\rightarrow$: $(..\ p\leftrightarrow q\ ..\ ..) \leftrightarrow (..\ p,q\ ..\rightarrow ..)$ and $(..\ ..\ \rightarrow p,q\ ..)$
$\exists\rightarrow$: $(..\exists x\ \phi x\ ..\rightarrow ..) \leftrightarrow (..\ \phi(f*_1\ \ldots\ *_n)\ ..\rightarrow ..)$
where f is a new skolem function and $*_1\ \ldots\ *_n$ are all the unification variables which occur in ϕx.
$\forall\rightarrow$: $(..\forall x\ \phi x\ ..\rightarrow ..) \leftrightarrow (..\forall(x*)\phi x,\ \phi*\ ..\rightarrow ..)$
where * is a new unification variable.
$=\rightarrow$: $(\Pi a\ \ldots \left\{\begin{matrix}a=t\\ t=a\end{matrix}\right\} ..\ \Gamma a \rightarrow \phi a\ \ldots\ \psi a) \leftrightarrow (\Pi t\ ..\ \Gamma t \rightarrow \phi t\ ..\ \psi t)$
where a is of the form $(f*_1\ldots *_n)$ and f is a skolem function not occurring in t. This is our version of the law of Leibniz.

2.1.2 Goal schemata:

$\rightarrow\blacksquare$: $(..\rightarrow ..\ \blacksquare\ ..) \leftrightarrow \blacksquare$
$\rightarrow\square$: $(..\rightarrow ..\ \square\ ..) \leftrightarrow (..\rightarrow ..\ ..)$
$\rightarrow\sim$: $(..\rightarrow ..\ \sim p\ ..) \leftrightarrow (..\ p\rightarrow ..\ ..)$
$\rightarrow\wedge$: $(..\rightarrow ..\ p\wedge q\ ..) \leftrightarrow (..\rightarrow ..\ p\ ..)$ and $(..\rightarrow ..\ q\ ..)$
$\rightarrow\vee$: $(..\rightarrow ..\ p\vee q\ ..) \leftrightarrow (..\rightarrow ..\ p,q\ ..)$
$\rightarrow\supset$: $(..\rightarrow ..\ p\rightarrow q\ ..) \leftrightarrow (..\ p\rightarrow ..\ q\ ..)$
$\rightarrow\leftrightarrow$: $(..\rightarrow ..\ p\leftrightarrow q\ ..) \leftrightarrow (..\ p\rightarrow\ ..\ q..)$ and $(..\ q\rightarrow\ ..\ p\ ..)$
$\rightarrow\forall$: $(..\rightarrow ..\forall x\ \phi x\ ..) \leftrightarrow (..\rightarrow ..\ \phi(f*_1\ldots *_n)\ ..)$
where f is a new skolem function and $*_1\ \ldots\ *_n$ are all the unification variables which occur in ϕx.
$\rightarrow\exists$: $(..\rightarrow ..\exists x\ \phi x\ ..) \leftrightarrow (..\rightarrow ..\exists\ (x*)\phi x,\ \phi*\ ..)$
where * is a new unification variable.

2.1.3 Replica Creation Schemata

$\forall()\rightarrow$: $(..\ \forall(x\ldots)\phi x\ ..\rightarrow ..) \leftrightarrow (..\ \forall(x\ldots *)\phi x, \phi*\ ..\rightarrow ..)$
where * is a new unification variable and no more than one unification variable occurs in $(x\ldots)$.
$\rightarrow\exists()$: $(..\rightarrow ..\ \exists(x\ldots)\phi x..) \leftrightarrow (..\rightarrow ..\ \exists(x\ldots *)\phi x, \phi*\ ..)$
where * is a new unification variable and no more than one unification variable occurs in $(x\ldots)$.

The items $\forall()\rightarrow$ and $\rightarrow\exists()$ are used to create additional replicas: $\phi*$, a universally quantified assertion $\forall(x\ldots)\phi x$, and an existentially quantified goal $\exists(x\ldots)\phi x$. The replica $\phi*$ is exactly like the original formula except that the initial quantifier is deleted and the bound variable associated with that quantifier is replaced by a new free unification variable.

A Unification Variable is a free variable which is created by $\forall\rightarrow$, $\rightarrow\exists$, $\forall()\rightarrow$, or $\rightarrow\exists()$ items, and which may later be instantiated to some term by the unification item (see section 2.1.4). Unification variables are written as a star sign: * possibly with numeric subscripts, such as: $*_1$, $*_2$, $*_3$.

In these four items we have seen formulae of the form $\forall(x\ldots)\phi x$ and $\exists(x\ldots)\phi x$ which are not usually thought of as being well formed sentences of logic. Such formulae should be interpreted as respectively $\forall x\phi x$ and $\exists x\phi x$ which are well formed sentences of logic. The ... list, which is called the replica instance list, is used merely to store certain pragmatic information used by the deductive system. This information is basically the list of unification variables (or more precisely the list of instantiations of the unification variables, see section 2.1.4) that were produced from this quantifier by applications of the $\forall\rightarrow$, $\rightarrow\exists$, $\forall()\rightarrow$, and $\rightarrow\exists()$ items.

2.1.4 The Unification schema:

Unify: $[(..\ p_1\ ..\rightarrow ..\ q_1\ ..)$ and .. and $(..\ p_n\ ..\rightarrow ..\ q_n\ ..)] \leftrightarrow [(..\ p_i\ ..\rightarrow ..\ q_i\ ..)$ and .. and $(..\ p_n\ ..\rightarrow ..\ q_n\ ..)]\ \Theta$

where $1 \leq i \leq n$ and Θ is any one of the sets of substitutions of terms for unification variables which satisfy both the forcing restriction and the instantiation restriction. These two restrictions are described below.

The forcing restriction is the requirement that the substitution makes tautologous the greatest number of sequents starting with the first sequent and progressing towards the nth sequent.

In the case that there actually is some substitution which will make all the sequents tautologous, without further unification variables being created by the $\forall\rightarrow$ and $\rightarrow\exists$ items, then Θ will be one such substitution. As a minor point, if Θ makes all the sequents tautologous, then the unification schema is defined to return $\blacksquare$.

The instantiation restriction is the requirement that no unification variable be instantiated to a term which already occurs in the replica instance list of the quantifier of the given sequent which contains the unification variable. The rationale behind this restriction is that if a term t occurs in the replica instance list of a quantifier such as $\forall$ in $(..\forall x \phi x .. \rightarrow ..)$ then the sub-formulae of ϕt must already occur in some sequent which must be proven in order to prove the theorem.

2.1.5 Other-logical schemata

atom: $(..\ p\ ..\rightarrow ..\ p\ ..) \leftrightarrow \blacksquare$

and: $(..\ \text{and}\ \blacksquare\ \text{and}\ ..) \leftrightarrow (..\ \text{and}\ ..)$

The logical items are not all used at the same time. In particular the $\forall() \rightarrow$, $\rightarrow \exists()$, and unify items are used in a special way. Initially, the interpreter evaluates each sequent trying to apply the items in the following order:

(1) Non splitting assertion items:
$\blacksquare\rightarrow$, $\square\rightarrow$, $\sim\rightarrow$, $\wedge\rightarrow$, $\exists\rightarrow$, $=\rightarrow$
(2) Non splitting goal items:
$\rightarrow\blacksquare$, $\rightarrow\square$, $\rightarrow\sim$, $\rightarrow\vee$, $\rightarrow\supset$, $\rightarrow\forall$
(3) Non logical items
(4) The atom and "and" items
(5) Splitting goal items: $\rightarrow\wedge$, $\rightarrow\leftrightarrow$
(6) Splitting assertion items:
$\vee\rightarrow$, $\supset\rightarrow$, $\leftrightarrow\rightarrow$
(7) $\rightarrow\exists$
(8) $\forall\rightarrow$

After the above items have been applied as many times as possible, the interpreter then tries to apply the unify item to the resulting conjunction of sequents.

If the application of the unification item results in $\blacksquare$ then the processes terminates because the theorem has been proven. But, if the application of the unification item does not result in $\blacksquare$, then the interpreter applies the $\forall() \rightarrow$ and $\rightarrow\exists()$ items to certain formulas, and then repeats the whole process starting at step (1).

However, because the proof described in this paper is obtained without using the $\forall() \rightarrow$ and $\rightarrow\exists()$ items we will not bother to describe the exact way they are used. The way they are used, along with more details about the Unify item, has however, been described in previous publications [2, 3].

One major difference between this theorem prover and our previous sequent logic theorem provers [2, 3] is that the $\rightarrow\exists$ and $\forall\rightarrow$ have been inserted into the initial evaluation procedure as steps 7 and 8, and thus one instance of every quantifier is initially created before the Unify rule is ever applied. The reason for this is due to the vast numbers of trivial quantifiers produced by the modal sequent logic described in section 2.2. It was found that unless instances of these quantifiers were produced before unification takes place, many important bindings would not be found quickly enough and irrelevant bindings would be produced by the forcing restriction, due to the fact that the relevant formulae would be available for matching.

2.2 Theory of Modality

Our theory of modality is based on a very strong modal logic which is described in [4, 5]. It consists of a single primitive unary symbol: $\vdash$ which is interpreted as logical truth. This modal logic is stronger than S5 and can be described by the following minimal set of inference rules and axioms:

RO: from p infer $\vdash p$
A1: $\vdash p \rightarrow p$
A2: $\vdash(p \rightarrow q) \rightarrow (\vdash p \rightarrow \vdash q)$
A3: $\vdash p \vee \vdash \sim \vdash p$
A4: $(\forall q\ \text{World*}q \rightarrow \vdash^{*} q\ p) \rightarrow \vdash p$

The inference rule RO and the axioms A1, A2 and A3 are essentially the S5 modal logic. The last axiom A4 expresses Leibniz's intuition that something is logically true only if it is true in all worlds. The World* and $\vdash$* symbols have the same meaning as respectively the World and the binary $\vdash$ symbols. World and binary $\vdash$ are both defined in section 2.2.2.

Our automatic theorem prover does not use the above axioms but is based on the sequent calculus derived from these axioms which is described in [5]. We describe this modal sequent calculus in section 2.2.1 and then list some definitions of modal concepts used by the theorem prover in section 2.2.2. Finally, in section 2.2.3 we discuss the possibility problem of modal logic.

2.2.1 Rules of the Sequent Calculus

We list below thirteen theorems of our modal logic of the form $p \rightarrow q$ or $r \rightarrow (p \leftrightarrow q)$ which may be used as rewrite rules replacing p by q in any context in which r is a hypothesis. The symbols World* and $\vdash$* have the same meaning respectively as World and $\vdash$, but are never to be replaced by their definitions in a proof procedure using these rules. Furthermore any initial theorem given to such a proof procedure must not itself contain the World* or $\vdash$* symbols, although it could of course contain the World and $\vdash$ symbols.

$\vdash$: $(\vdash p) \leftrightarrow \forall w(\text{World*}\ w) \rightarrow \vdash^{*}w\ p$
$\vdash\wedge$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(p \wedge q)) \leftrightarrow (\vdash^{*}w\ p \wedge \vdash^{*}w\ q))$
$\vdash\vee$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(p \vee q)) \leftrightarrow (\vdash^{*}w\ p \vee \vdash^{*}w\ q))$
$\vdash\rightarrow$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(p \rightarrow q)) \leftrightarrow (\vdash^{*}w\ p \rightarrow \vdash^{*}w\ q))$
$\vdash\leftrightarrow$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(p \leftrightarrow q)) \leftrightarrow (\vdash^{*}w\ p \leftrightarrow \vdash^{*}w\ q))$
$\vdash\sim$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(\sim p)) \leftrightarrow \sim \vdash^{*}w\ p)$
$\vdash\blacksquare$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w\ \blacksquare) \leftrightarrow \blacksquare)$
$\vdash\square$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w\ \square) \leftrightarrow \square)$
$\vdash\forall$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(\forall x\ \phi x)) \leftrightarrow (\forall x\ \vdash^{*}w\ \phi x))$
$\vdash\exists$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(\exists x\ \phi x)) \leftrightarrow (\exists x\ \vdash^{*}w\ \phi w))$
$\vdash a$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(\forall p\ \phi p)) \leftrightarrow (\forall p\ \vdash^{*}w\ \phi p))$
$\vdash e$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(\exists p\ \phi p)) \leftrightarrow (\exists p\ \vdash^{*}w\ \phi p))$
$\vdash\vdash$: $(\text{World*}\ w) \rightarrow ((\vdash^{*}w(\vdash p)) \leftrightarrow \vdash p)$

The $\vdash\forall$ and $\vdash\exists$ theorems pertain to quantifiers of object language variables whereas the $\vdash$a, $\vdash$e theorems pertain to quantifiers for propositional variables. The $\vdash\forall$ and $\vdash\exists$ theorems are equivalent to the fact that something is an object iff it is logically true that it is an object. All these theorems hold regardless of whether propositions are objects or not.

In order to try to prove a theorem ψ with a proof procedure using these rules, sometimes it must actually try to prove $\vdash \psi$ instead. There is a deep and beautiful reason for this which is basically that this initial $\vdash$ inserted before ψ is a symbol of the metalanguage of this logic as are all the $\vdash$ * and World* symbols. Essentially $\vdash \psi$ is the statement in the methatheory that ψ is logically true, and it is this rather than ψ itself which we are trying to prove.

Our theorem prover also contains the following special laws dealing with equality.

$= \vdash \leftrightarrow$: $\vdash (p \leftrightarrow q) \leftrightarrow p = q$
$= \vdash \sim$: $\vdash \sim p \quad \leftrightarrow p = \square$
$= \vdash$: $\vdash p \quad \leftrightarrow p = \blacksquare$

These laws are used only on assertions in a sequent, and only in the case that either p or q is a skolem function.

These three laws are not derivable from the minimal axiomatization described earlier, but are derivable if the following axiom is added:

A5: $\vdash (p \leftrightarrow q) \rightarrow p = q$

2.2.2 Modal Concepts

D$\vdash$: $\vdash p\ q \leftrightarrow \vdash (p \rightarrow q)$	"p entails q"
D$\equiv$: $p \equiv q \leftrightarrow \vdash (p \leftrightarrow q)$	"p is synonomous to q"
D$\Diamond$: $\Diamond p \leftrightarrow \sim \vdash \sim p$	"p is possible"
Ddet : (Det p q) $\leftrightarrow \vdash p\ q \vee \vdash p \sim q$	"p determines q"
Dcom :(Complete p) $\leftrightarrow \forall q$(Det p q)	"p is complete"
Dwor :(World p) $\leftrightarrow \Diamond p \wedge$ (Complete p)	"p is a world"
Dval :(Valid p) $\leftrightarrow \forall q$(World q) $\rightarrow \vdash q\ p$	"p is valid"
Dsat :(Sat p) $\leftrightarrow \exists q$(World q) $\wedge \vdash q\ p$	"p is satisfiable"
Duniq :(Uniq <u>p</u> ϕp) $\leftrightarrow \forall p \forall q\ \phi p \wedge \phi q \rightarrow p \equiv q$	"p is unique in ϕ"
Done :(One <u>p</u> ϕp) $\leftrightarrow \exists q\ \forall r(\phi r \leftrightarrow q \equiv r)$	"p is one in ϕ"
Dcat :(Cat p) $\leftrightarrow$ (One q(World q $\wedge \vdash$ q p))	"p is categorical"
Dmaxp :(Maxpos p) $\leftrightarrow$ (One q($\Diamond$q $\wedge \vdash$ q p))	"p is maximally possible"
Dmax :(Max p) $\leftrightarrow$ (One q($\vdash$ q p))	"p is maximal"

2.2.3 The Possibility Problem

The possibility problem of modal logic is that from the logical axioms of modal logic we cannot prove certain elementary facts about the possibility of conjunctions of distinct possibly negated atomic expressions consisting of nonlogical symbols. For example, if we have a theory formulated in our modal logic which contains the nonlogical atomic expression (ON A B) then since $\sim$(ON A B) is not logically true, it follows that (ON A B) must be possible. Yet $\Diamond$(ON A B) is not a theorem of our modal logic.

Thus, for any theory expressed in modal logic, a certain number of axioms dealing with possibility should also be added.
For example, in the case of the propositional logic, or in the case of the quantificational logic over a finite domain since it reduces to propositional logic, one sufficient but inefficient axiomatization would be to assert the possibility of all consistent disjunctions of conjunctions of literals as additional nonlogical axioms:

$$\Diamond(\bigvee(\bigwedge \text{Literals})$$

A computationally better axiomatization which is obtained by noting that the possibility of a disjunction of sentences is implied by the possibility of any one of those sentences:

$$\Diamond x \rightarrow \Diamond(x \vee y)$$

is to assert only the possibility of all consistent conjunctions of literals:

$$\Diamond\ (\bigwedge \text{literals})$$

Using the meaning function M defined in section 2.4 this may be done in a finite manner by adding the single axiom:

Prop Pos Ax: (Conj S) $\wedge$ (Consist S) $\rightarrow \Diamond$(M S)

where Conj and Consist are recursive functions defined as follows:

(Conj S)= df (Lit S) $\vee \exists$T $\exists$R (S=[T '$\wedge$ R] $\wedge$ (Lit T) $\wedge$ (Conj R))
(Lit S) = df ($\exists$T S=['$\sim$T] $\wedge$ (Atomicsent T)) $\vee$ (Atomicsent S)
(Consist [ ]) = df $\blacksquare$
(Consist [S.L]) = df (Consist2 S L) $\wedge$ (Consist L)
(Consist2 S [ ])= df $\blacksquare$
(Consist2 S[T.L]) = df $\sim$(Opp S T) $\wedge$ (Consist2 S L)
(Opp S T) = df S=['$\sim$T] $\vee$ T=['$\sim$S]

The methods for representing object language expressions in our logic and for obtaining their meanings are defined in sections 2.3. and 2.4.

It is important to note that it is unlikely that this finite recursive axiomatization of the possibility problem for propositional logic, could be extended to even first order quantificational logic. The reason is that if it could then it would provide a positive solution to Hilbert's Einscheingunts problem. However, if we are willing to forgo the requirement of recursiveness then by letting S in PropPosAx be an infinite conjunction of variable free literals then it is possible to produce an axiomatization for a first order quantificational logic object language. See section 2.7.

2.3 Theory of Syntax

Our theory of syntax contains two basic primitive symbols, a binary symbol: Cons and a zeroary symbol: Nil. It also contains a number of other zeroary symbols
'$\wedge$, '$\vee$, '$\rightarrow$, '$\leftrightarrow$, '$\sim$, '$\blacksquare$, '$\square$, '$\forall$, '$\exists$, 'x, 'p
each of which may be thought of as being a name of the symbol obtained from it by deleting the accent sign. The accented symbols may, however, be thought of as being defined in terms of Cons and Nil and thus need not be viewed as additional primitive symbols.

Our theory of syntax currently consists of only one definition scheme:

Dlist: $[x_1 \ldots x_n]$ = df (Cons $x_1 \ldots$ (Cons x_n Nil)...)

Our theory of syntax is described in more detail in [6].

2.4 The Theory of Meaning

The syntax of our theory of meaning currently consists of one binary primitive symbol: m, such that (m S A) is interpreted as the meaning of an expression S in the association list A. It also consists of one unary defined symbol: M, which stands for the meaning of an expression in a null association list.

We assume that the definition and recursive axioms for meaning are all logically true.

Our theory of meaning is based on the recursive meaning function m described in [6]. This function consists of the following recursive axioms:

M :	$(M\ S) = df\ (m\ S\ Nil)$
$m\ '\wedge$:	$(m[S\ '\wedge\ T]A) \leftrightarrow (m\ S\ A) \wedge (m\ T\ A)$
$m\ '\vee$:	$(m[S\ '\vee\ T]A) \leftrightarrow (m\ S\ A) \vee (m\ T\ A)$
$m\ '\rightarrow$:	$(m[S\ '\rightarrow T]A) \leftrightarrow (m\ S\ A) \rightarrow (m\ T\ A)$
$m\ '\leftrightarrow$:	$(m[S\ '\leftrightarrow\ T]A) \leftrightarrow (m\ S\ A) \leftrightarrow (m\ T\ A)$
$m\ '\sim$:	$(m['\sim S]A) \leftrightarrow \sim(m\ S\ A)$
$m\ '\blacksquare$:	$(m\ '\blacksquare\ A) \leftrightarrow \blacksquare$
$m\ '\square$:	$(m\ '\square\ A) \leftrightarrow \square$
$m\ '\forall$:	$(m\ '\forall\ V\ S]A) \leftrightarrow \forall X(m\ S[[V.X].A])$
$m\ '\exists$:	$(m['\exists\ V\ S]A) \leftrightarrow \exists X(m\ S[[V.X].A])$
$m\ '\ a$:	$(m['\forall\ U\ S]A) \leftrightarrow \forall p(m\ S[[U.p].A])$
$m\ '\ e$:	$(m['\exists\ U\ S]A) \leftrightarrow \exists p(m\ S[[U.p].A])$
$m\ V$:	$(m\ V\ A) \leftrightarrow (Val\ V\ A)$
$m\ U$:	$(m\ U\ A) \leftrightarrow (Val\ U\ A)$

m' : $(m['\phi S_1 \ldots S_n]A) \leftrightarrow (\phi(mS_1A)\ldots(mS_nA))$

for each non-logical symbol ϕ of the object language.

In these rules the notation $[\alpha_1\ldots\alpha_n]$ is an abbreviation for $(Cons\ \alpha_1\ldots(Cons\ \alpha_n\ Nil))$ and the notation $[\alpha.\beta]$ is an abbreviation for $(Cons\ \alpha\ \beta)$. Thus, for example the $m\ '\sim$ law actually is:

$(m(Cons\ '\sim(Cons\ S\ Nil))A) \leftrightarrow \sim(m\ S\ A)$

S and T range over expressions, usually sentences. A ranges over association lists. V ranges over object variables and U ranges over propositional variables.

The Val function is defined as follows:

V1: $(Val\ X[[X.Z].A])=Z$

V2: $(\sim X=Y) \rightarrow (Val\ X[[Y.Z].A])=(Val\ X\ A)$

2.5 Model Theory

Our Model Theory consists of the following definitions:

D\$Com $(\$Complete\ p\ \underline{q}\ \phi q) \leftrightarrow (\forall q\ \phi q \rightarrow (Det\ p\ q))$
"p is complete for ϕ"

D\$Wor$(\$World\ p\ \underline{q}\ \phi q) \leftrightarrow \Diamond p \wedge (\$Complete\ p\ \underline{q}\phi q)$
"p is a world for ϕ"

D\$Val$(\$Valid\ p\ \underline{q}\ \phi q) \leftrightarrow \forall q\ (\$World\ q\ \underline{r}\ \phi r) \rightarrow\ \vdash q\ p$
"p is valid for ϕ"

D\$Sat$(\$Sat\ p\ \underline{q}\ \phi q) \leftrightarrow \exists q\ (\$World\ q\ \underline{r}\ \phi r) \wedge \vdash q\ p$
"p is satisfiable for ϕ"

D\$Cat$(\ Cat\ p\ \underline{q}\ \phi q) \leftrightarrow (One\ \underline{q}\ (\$World\ q\ \underline{r}\ \phi r) \wedge \vdash q\ p$
"p is categorical for ϕ"

D\$M $(Model\ p) \leftrightarrow (\$World\ p\ \underline{q}\ (\exists S\ (M\ S) \equiv q))$
"p is a model"

D\$MSat$(Modelsat\ p) \leftrightarrow \exists q\ (Model\ q) \wedge \vdash q\ p$
"p is satisfied by some model"

D\$MVal $(Modelval\ p) \leftrightarrow \forall q\ (Model\ q) \rightarrow\ \vdash q\ p$
"p is true in all models"

D\$MCat $(Modelcat\ p) \leftrightarrow (One\ \underline{q}\ Model\ q \wedge \vdash\ q\ p)$
" p is true in exactly one model"

Note that a model is essentially nothing more than a \$World with respect to the subdomain of concepts which are expressible in the object language.

2.6 Proof Theory

Our proof consists of a single unary primitive symbol: $\Vdash$ such that $\Vdash S$ is interpreted to mean that the sentence S is a theorem, or rather that S is provable.

Our proof theory also contains several defined symbols which are listed below with their definitions and interpretations.

D$\Vdash$: $\Vdash T\ S \leftrightarrow \Vdash[T \rightarrow S]$
"S is a theorem of T"

D$\Diamond$: $\Diamond\ S\ \leftrightarrow \sim\Vdash['\sim S]$
"S is consistent"

D$\Diamond_o$: $\Diamond_o\ S \leftrightarrow \sim\Vdash['\sim S]$
"S is consistent"

Ddec : $(Dec\ T\ S) \leftrightarrow \Vdash T\ S\ \vee \Vdash T['\sim S]$
"T decides S"

DThCom : $(ThComplete\ T) \leftrightarrow\ \forall S(Dec\ T\ S)$
"T is a complete theory"

DThWor : $(ThWorld\ T) \leftrightarrow\ \Diamond_o T \wedge (ThComplete\ T)$
"T is a world theory"

DThWor$_o$: $(ThWorld_o\ T) \leftrightarrow\ \Diamond_o T \wedge (ThComplete\ T)$
"T is a world theory"

DthWExt : $(ThWorldext\ S) \leftrightarrow\ \exists T\ (ThWorld\ T) \wedge \Vdash T\ S$
"T has a world theory extension"

The purpose of the $\Diamond_o$ and ThWorld$_o$ definitions is pragmatic and will be explained in section 2.7.

Furthermore, the variables S and T in the last definition (DThWExt) are not of the same sort. The sort distinctions have been omitted in accordance with this theorem prover's basic rational for handling sorts as discussed in section 2. In this last definition the variable S ranges over finite sentences whereas the variable T ranges over both finite and infinite sentences. Infinite conjunctions of finite sentences, however, are probably sufficient.

2.7 Miscellaneous Lemmas

The following lemmas are assumed as extra rewrite rules in various proofs. In particular they are necessary for a proof of the completeness theorem.

Sound: $\Vdash S\ \rightarrow \vdash(M\ S)$

LIND: $\Diamond S\ \rightarrow (ThWorldext\ S)$

It should not be thought that the axiom LIND is either logically false or even that it is false in a meta theory which is strong enough to derive Godel's incompleteness theorem because LIND claims only that there is a complete consistent extention, and not that this extention is a finite sentence.

PosAx: $(ThWorld\ S) \rightarrow \Diamond(M\ S) \wedge (ThWorld_o\ S)$

The PosAx theorem is equivalent to:

$(ThWorld\ S) \rightarrow \Diamond(MS)$

The intuitive justification for this axiom lies in the fact that a Thworld may be thought of as being an infinite conjunction of variable free literals. Viewed in this manner we see that PosAx is nothing more than a solution to the possibility problem for those concepts which are expressible in the object language.

The above three axioms which are only implications are used as rewrite rules only on assertions in a sequent. Thus, for example, the sequent:

$\Vdash S \rightarrow \phi$

would become:

$\vdash (M\ S) \rightarrow \phi$

by application of the soundness lemma: Sound, to the assertion $\Vdash S$.
The Soundness lemma would not be applied to the sequent

$\phi \rightarrow \Vdash S$

because $\Vdash S$ is a goal.

It is easy to see that if unrestricted Lindenbaum Lemma: LIND could be applied an infinite number of times, since $\Diamond S$ is rewritten as (ThWorldext S) which in turn may, by using various definitions, be rewritten as:
$(\exists U\ \Diamond U \wedge (\text{ThComplete } U) \wedge \Vdash U\ S)$, thus producing a new formulae $\Diamond U$ to which LIND can again be applied. For this reason we restrict LIND to being applied only to $\Diamond S$ formulas not generated by previous applications of LIND. This restriction is implemented by simply defining ThWorldext S to produce $(\exists U\ \Diamond_o U \wedge (\text{ThComplete } U) \wedge \Vdash U\ S)$ where $\Diamond_o$ is a distinct symbol from $\Diamond$ even though it has exactly the same meaning.

It is also easy to see that PosAx would have the same sort of problem, and for this reason it uses the symbol ThWorld_o.

One final point worth noting about these lemmas is that even though they are all implications and not equivalences, they would be true even if they were equivalences. The reason we do not make Sound and PosAx equivalences is that we don't need to know these facts in order to prove, for example, the completeness theorem, and furthermore for pragmatic reasons we would never want to apply them to goal formulas anyway. In the case of the soundness theorem, the reason we do not assume an equivalence is that the equivalence is itself a version of the completeness theorem, and it obviously makes no sense to assume the completeness theorem while trying to prove it.

3. THE COMPLETENESS THEOREM

The theorem prover for metatheory is capable of proving the completeness of quantificational logic from intuitively true assumptions. Specifically the theorems described in section 2 are assumed as axioms. The completeness theorem which states that the meaning of every consistent theory is possible is written as follows:

$\Diamond S \rightarrow \Diamond (MS)$

This theorem was proven by our theorem prover on a DEC KI10 computer in 38 seconds. The number of sequents produced was 326 octal. See [7].

This proof, by the way, is probably the first proof of a completeness theorem relating provability to the modal concept of logical truth. The notion of truth unlike set theoretic characterizations [8,9] can be shown to satisfy Tarski's famous criteria for truth in a world [10] namely that (in that world) something is a true sentence in a world iff it is the case.

$\vdash w\ (\vdash w(MS) \leftrightarrow (MS))$

(in w) (S is a true sentence iff (MS)).
One final advantage of this completeness proof is that it is a proof purely from the laws of logic (plus a few recursive functions which are probably definable in second order logic) alone, and assumes no axioms of set theory such as the axiom of choice. So in fact it has shown that not only does completeness follow from certain strong set theoretic principles, but that it is logically true which is a much more general result.

Acknowledgements

It has taken me over one half of a decade to develop the meta theory on which the automatic theorem prover is based. I thank Professor B. Meltzer for supporting this research over this long period, and for commenting on various drafts of this theory, the first of which was written five and one half years ago. I also thank the many other people who have interacted on various aspects of this theory, including Dr. R. Kowalski, Dr. K. Clark, Dr. B. Richards, Dr. W. Bibel, Dr. S. Tarnlund, and especially Dr. C. Schwind.

References

1. Brown, F.M. "An investigation into the goals of research in Automatic Theorem Proving as related to Mathematical Reasoning", DAI Research Report 49, 1978.
2. Brown, F.M. "A Theorem Prover for Elementary Set Theory", IJCAI5 Conf. Proc., MIT, 1977.
3. Brown, F.M. "Towards the Automation of Set Theory and its Logic", DAI Research Report 34, 1977.
4. Brown, F.M. "A Theory of Meaning", DAI Working Paper, 17, 1976.
5. Brown, F.M. "A Sequent Calculus for Modal Quantificational Logic", 3rd AISB/GI Conf. Proc., Hamburgh, 1978.
6. Brown, F.M. "The Theory of Meaning", DAI Research Report 35, 1977.
7. Brown, F.M. "An Automatic Proof of the Completeness of Quantificational Logic" DAI Research Report 52, 1978.
8. Henkin, L. "The Completeness of the First Order Functional Calculus", JSL, Vol. 14, 1949.
9. Mendelson, E. Introduction to Mathematical Logic, van Nostrand Reinhold Company, New York, 1964.
10. Tarski, A. "The Concept of Truth in Formalized Languages" (1931) Logic, Semantics, Mata mathematics, trans by J.H. Woodger, Oxford, Clarendon Press, 1956.