# The Consistency of SYMEVAL

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

TR 138 February 1980

# The Consistency of SYMEVAL

## Abstract

In the past few years we have developed a powerful automatic theorem prover: SYMEVAL which has been used to prove numerous theorems in number theory, set theory, algebra, meta methamatics, and various intensional logics. The problem remains that if the basic logical laws used by the theorem prover are inconsistent then the proofs are worthless. For this reason, in this paper we show how to prove the consistency of the classical logic used by SYMEVAL.

## 1. Introduction

In the past few years we have developed a powerful automatic theorem prover SYMEVAL which has been used to prove numerous theorems in number theory [1,2], set theory [3,4,5], algebra [6,7] meta mathematics [8,9], and various intensional logics [10]. However, it has been pointed out to me that if the basic logical inference rules and axiom schematta used by SYMEVAL were inconsistent then all these proofs would be essentially worthless. For this reason, in this paper we sketch how to prove in second order logic the consistency of the logical inference rules and axiom schematta used by SYMEVAL. It should be noted that the actual laws used by SYMEVAL differ considerably from those used in other systems [11,12,13] and thus this consistency proof is quite new. For example SYMEVAL'S unification axiom schemma unlike Robinson's [12], sometimes combines nested skolem functions together for efficiency. In Section 2, we first define SYMEVAL'S single universal inference rule. We then define all the axiom schema for classical logic used by SYMEVAL. In Section 3, we provide the consistency of this inference rule and these schematta relative to second order logic. Finally in section 3 we make a few conclusions.

## 2. Symeval

We first define in section 2.1 the universal inference rule used by SYMEVAL. Then in Section 2.2 we describe the logical schemata used by SYMEVAL.

### 2.1 The Universal Inference Rule

Our theorem prover SYMEVAL 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 $\phi$ 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 $\phi$ 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 $\phi$ of some item, then it replaces that expression by the corresponding instance $\psi\theta$ or $\psi$. 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 $\Pi$. 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 $\Pi x$ 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.

## 2.2 The Logical Schemata

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:

$$p_1, \dots ,p_n \rightarrow q_1, \dots , q_m =_{df} (p_1 \wedge \dots \wedge p_n) \rightarrow (q_1 \vee \dots \vee q_m)$$

where $p_i$ and $q_j$ are sentences. Thus a sequent may be thought of as being a database of statements $p_1, \dots ,p_n$ called assertions which occur before the sequent arrow, and statements $q_1, \dots , q_m$ 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:

2.2.1 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 $(.. .. \rightarrow p,q ..)$

$\exists \rightarrow$ : $(.. \exists x\ \phi x .. \rightarrow ..) \leftrightarrow (.. \phi(\not{f} *_1 \dots *_n) .. \rightarrow ..)$

where $\not{f}$ is a new skolem function and $*_1 \dots *_n$ are all the unification variables which occur in $\phi x$.

$\forall \rightarrow$ : $(.. \forall x\ \phi x .. \rightarrow ..) \leftrightarrow (.. \forall(x^*)\phi x, \phi^* .. \rightarrow ..)$

where $*$ is a new unification variable

$= \rightarrow$ : $(\Pi a \dots \left\{\begin{matrix} a=t \\ t=a \end{matrix}\right\} .. \Gamma a \rightarrow \phi a \dots \psi a) \leftrightarrow (\Pi t .. \Gamma t \rightarrow \phi t .. \psi t)$

where $a$ is of the form $(\not{f} *_1 \dots *_n)$ and $\not{f}$ is a skolem function not occurring in $t$. This is our version of the law of Leibniz.

2.2.2 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 $\phi x$.

$\rightarrow\exists$ : $(.. \rightarrow .. \exists x\, \phi x ..) \leftrightarrow (.. \rightarrow .. \exists(x^*)\phi x,\ \phi^* ..)$
where $*$ is a new unification variable

2.2.3 Replica Creation Schemata

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

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

The item $\forall()\rightarrow$ and $\rightarrow\exists()$ are used to create additional replicas: $\phi^*$, a universally quantified assertion $\forall(x...)\phi x$, and an existentially quantified goal $\exists(x...)\phi x$. 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 $\forall(x...)\phi x$ and $\exists(x...)\phi x$ 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.

2.2.4 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 $\Theta$ 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 ∀ → and → ∃ 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 .

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 ∀ in (.. ∀xϕx .. → ..) 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 .. → .. p ..) ↔ ■

and: (.. and ■ and ..) ↔ (.. and ..)

The logical items are not all used at the same time. In particular the ∀()→, →∃(), 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:

■ →, □ →, ∼→, ∧ →, ∃→, = →

(2) Non splitting goal items:

→ ■ , → □ , →∼, →∨, →⊃ , →∀

(3) Non logical items

(4) The atom and "and" items

(5) Splitting goal items: →∧ , → ↔

(6) Splitting assertion items: ∨ →, ⊃ →, ↔ →

(7) → ∃

(8) ∀ →

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 ■ then the processes terminates because the theorem has been proven. But, if the application of the unification item does not result in ■, then the interpreter applies the ∀()→ and →∃() items to certain formulas, and then repeats the whole process starting at step (1)

## 3. Consistency of Symeval

We first prove the consistency of the Universal inference rule in section 3.1 and then in section 3.2 give consistency proofs of a representative set of the logical schematta. All these proofs are carried out in second order logic and hence the consistency of the system is relative to the consistency of second order logic.

### 3.1 Consistency of the Universal Inference Rule

The universal inference rule has two simple versions:

from $p \leftrightarrow q$ and $\psi p$ infer $\psi q$

from $p = q$ and $\psi p$ infer $\psi q$

which may be viewed as instances of the universal inference rule's two conditional versions where C = true.

from $(C \rightarrow (p \leftrightarrow q)$ and $(C \rightarrow \psi p)$ infer $C \rightarrow \psi q$

from $(C \rightarrow (p = q))$ and $C \rightarrow \psi p$ infer $C \rightarrow \psi q$

These can be rewritten as meta theorems [8,9] as

$$C = (p \leftrightarrow q) \wedge (C \rightarrow \psi p) \rightarrow (C \rightarrow \psi q)$$

$$C \rightarrow (p = q) \wedge (C \rightarrow \psi p) \rightarrow (C \rightarrow \psi q)$$

which would be true if:

$$(p \leftrightarrow q) \wedge \psi p \rightarrow \psi p$$

$$(p = q) \wedge \psi p \rightarrow \psi q$$

However since

$$p \leftrightarrow q \wedge \psi p \rightarrow \psi q \quad : \text{ logic}$$

$$\psi p \leftrightarrow \psi p$$

and since

$p = q \wedge \psi p \rightarrow \psi q$

$(\forall \psi\ \psi p \leftrightarrow \psi q \wedge \psi p \rightarrow \psi q$ : def. of = in 2nd order logic

$\psi\theta := \psi\theta$ : substitution

$\psi p \leftrightarrow \psi q \wedge \psi p \rightarrow \psi q$ : logic

$\psi q \rightarrow \psi q$ : logic

it follows that the versions of SYMEVALS universal inference rule are consistent relative to second order logic.

## 3.2 Consistency of the Logical Schematta

We will now prove that the following schematta of our sequent logic are consistent: $\rightarrow\wedge$, $\rightarrow\vee$, $\rightarrow\sim$, $\rightarrow\forall$, $\rightarrow\exists$, $\rightarrow$Unify, and $= \rightarrow$ Proofs of the consistency of the remaining schematta may be obtained analogously.

These proofs will be carried out in the second order logic similar to that which Frege used in Begriftschrift. In particular the skolem functions: $f$ used in our logic will be represented as universally quantified function variables $\forall f$. In these proofs:

(1) $\forall \underline{f}$ represents a sequence of universal quantifiers: $\forall f_1 \ldots \forall f_n$.

(2) $\exists\underline{*}$ represents a sequence of existential quantifiers:

$\exists *_1 \ldots \exists *_n$.

(3) $\underline{S}$ represents a conjunction of sequents: $S_1 \wedge \ldots \wedge S_n$.

(4) $\phi$ when it occurs within a sequent represents a sequence of formulas $\phi_1, \ldots, \phi_n$ otherwise it represents a conjunction of formulas.

(5) $\psi$ when it occurs within a sequent represents a sequence of formulas, otherwise it represents a disjunction of formulas.

Recall that a sequent $(\phi \rightarrow \psi)$ or rather: $(\phi_1, \ldots, \phi_n \rightarrow \psi_1 \ldots, \psi_n)$ is interpreted as:

$(\phi \supset \psi)$ or rather: $(\phi_1 \wedge \ldots \wedge \phi_n \supset \psi_1 \vee \ldots \vee \psi_n)$

We will now prove the consistency of each of the above six rules in succession:

$\to\wedge$ $\quad \forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \to \psi, (p \wedge q)))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \supset \psi \vee (p \wedge q)))$ : logic

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge \phi \supset (\psi \vee p) \wedge (\psi \vee q)))$ : logic

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee p)) \wedge (\phi \supset (\psi \vee q))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \to \psi, p) \wedge (\phi \to \psi, q))$

$\to\vee$ $\quad \forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \to \psi, (p \vee q)))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \supset \psi \vee p \vee q))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \to \psi, p, q))$

$\to\sim$ $\quad \forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \to \psi, \sim p))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee \sim p)))$ : logic

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge ((\phi \wedge p) \supset \psi))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi, p \to \psi))$

$\to\forall$ $\quad \forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \to \psi, \forall x P x))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee (\forall x P x))))$ : renaming x by a gensymed variable a

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee (\forall a P a))))$ : since a does not occur in $\psi$

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge (\phi \supset \forall a(\psi \vee Pa)))$ : since a does not occur in $\phi$

$\forall\underline{f}\ \exists\underline{*}(\underline{S} \wedge \forall a(\phi \supset (\psi \vee Pa)))$ : since a does not occur in $\underline{S}$

$\forall\underline{f}\ \exists\underline{*}\ \forall a(\underline{S} \wedge (\phi \supset (\psi \vee Pa))$ : def of sequent

$\forall\underline{f}\ \exists\underline{*}\ \forall a(\underline{S} \wedge (\phi \to \psi, Pa))$ : def of $\exists\underline{*}$

$\forall\underline{f}\ \exists *_1 \ldots \exists *_n\ \forall a(\underline{S} \wedge (\phi \to \psi, Pa))$ : second order logic

$\forall\underline{f}\ \exists *_1 \ldots \exists *_{n-1}\ \forall a\ \exists *_n(\underline{S} \wedge (\phi \to \psi, P(a *_1)))$ $\vdots$

$\vdots$

$\forall\underline{f}\ \exists *_1\ \forall a\ \exists *_2 \ldots \exists *_n(\underline{S} \wedge (\phi \to \psi\ P(a *_1 \ldots *_{n-1})))$ : second order logic

$\forall\underline{f}\ \forall a\ \exists *_1 \ldots \exists *_n(\underline{S} \wedge (\phi \to \psi, P(a *_1 \ldots *_n)))$ : def of $\exists\underline{*}$

$\forall\underline{f}\ \forall a\ \exists\underline{*}(\underline{S} \wedge (\phi \to \psi, P(a *_1 \ldots *_n)))$

The law of second order logic that is used $n$ times in this proof is:

$$(\exists * \; \forall a(\Pi\; a\; *)) \leftrightarrow \forall a\; \exists * (\Pi(a\; *)*)$$

Note that $\forall \underline{f}\; \forall a$ is a sequence of universally quantified variables, and hence is essentially of the same form as $\forall \underline{f}$.

$\rightarrow \exists \quad \forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge (\phi \rightarrow \psi, (\exists x P x)))$ : def of sequent

$\forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee (\exists x P x))))$ : logic

$\forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee (\exists x P x) \vee (\exists x P x))))$ : renaming the second x to a gensyned variable x

$\forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee (\exists x P x) \vee (\exists \alpha P \alpha))))$ : since α does not occur in Px

$\forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge (\phi \supset (\psi \vee \exists \alpha((\exists x P x) \vee P\alpha))))$ : since α does not occur in ψ

$\forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge (\phi \supset \exists \alpha(\psi \vee (\exists x P x) \vee P\alpha)))$ : since α does not occur in ϕ

$\forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge \exists \alpha(\phi \supset \psi \vee \exists x\, P x \vee P\alpha))$ : since α does not occur in $\underline{S}$

$\forall \underline{f}\; \exists \underline{*}\; \exists \alpha(\underline{S} \wedge (\phi \supset (\psi \vee \exists x\, P x \vee P\alpha)))$ : def of sequent

$\forall \underline{f}\; \exists \underline{*}\; \exists \alpha(\underline{S} \wedge (\phi \rightarrow \psi, \exists x P x, P\alpha))$

Note that $\exists \underline{*}\; \exists \alpha$ is a sequence of existentially quantified variables and hence essentially of the form of $\exists \underline{*}$.

<u>Unify</u> We let $\exists \underline{*}'$ be the sequence of existential quantifiers whose variables are instantiated by the substitution $\theta = [\underline{*}' \leftarrow \underline{f}]$ of the unification rule. We let $\exists \underline{*}$ be the quantifiers for any other unification variables.

We let $\underline{T}$ be the sequents which are made tautologous by the substitution θ and let $\underline{S}$ be any other sequents.

$\forall \underline{f}\; \exists \underline{*}\; \exists \underline{*}'\, \underline{S} \wedge \underline{T}$ : is implied by:

$\forall \underline{f}\; \exists \underline{*}(\underline{S} \wedge \underline{T})\; [\underline{*}' \leftarrow \underline{f}]$ : substitution

$\forall \underline{f}\; \exists \underline{*}(\underline{S}[\underline{*}' \leftarrow \underline{f}]) \wedge (\underline{T}[\underline{*}' \leftarrow \underline{f}])$ : $\underline{T}$ is made tautologous by θ

$\forall \underline{f}\; \exists \underline{*}(\underline{S}[\underline{*}' \leftarrow \underline{f}]) \wedge \blacksquare$

$\forall \underline{f}\; \exists \underline{*}\; \underline{S}[\underline{*}' \leftarrow \underline{f}]$

One final point is that we define the substitution $t^*_1$ for $*_o$ into a skolem function $f^*_o$ as $f^*_1$. This is clearly consistent since $\forall f\ \forall t(\phi(f(t^*_1))(t^*_1))$ is implied by $\forall f\ \forall t(\phi(f^*_o)(t^*_o))$.

Note that every occurence of $f$ in $\forall f \forall t(\phi(f(t^*_1))(t^*_o))$ is required to contain a term equal to $t^*_o$ as its argument.

$\forall \underline{f}\ \forall a \quad \exists \underline{*}\ (S \wedge (\varphi,\ (a^*_1 \ldots *_k) = t\underline{*} \rightarrow \psi))$ : def. of sequent

$\forall \underline{f}\ \forall a\ \exists \underline{*}\ (S \wedge ((\varphi \wedge (a\ *_1 \ldots *_n) = t\underline{*}) \supset \psi))$ : logic

$\forall \underline{f}\ \forall a\ \exists \underline{*}\ (S \wedge ((a^*_1 .. *_n) = t \supset (\varphi \supset \psi)))$ : 2nd order logic

$\forall \underline{f}\ \forall a\ \exists \underline{*}\ (S \wedge ((\forall \Gamma\ \Gamma(a^*_1 .. *_n) \leftrightarrow \Gamma t) \supset (\varphi \supset \psi)))$ : instance

$$\Gamma(a^*_1 \ldots *_n) = (\varphi \supset \Psi)\ [a^*_1 \ldots *_n]$$

$\forall \underline{f}\ \forall a\ \exists \underline{*}\ (S \wedge ((\varphi \supset \Psi \leftrightarrow (\varphi \supset \psi)[t]) \supset (\varphi \supset \psi))$: logic

$\forall \underline{f}\ \forall a\ \exists \underline{*}\ S \wedge (\varphi \supset \Psi)\ [t]$ : def. of sequent

$\forall f\ \forall a\ \exists \underline{*}\ S \wedge (\varphi \rightarrow \psi)\ [t]$

## 3. Conclusions

We have shown how to prove the consistency of logical laws used by our automatic theorem prover SYMEVAL relative to second order logic. We feel that this provides some evidence that the proofs obtained by this theorem prover are correct.

## References

1. Brown, F. M., "A Deductive System for Elementary Arithmetic," 2nd AISB Conference Proceedings, Edinburgh, July 1976.

2. Brown, F. M., "Doing Arithmetic Without Diagrams," Artificial Intelligence Vol. 8, Spring 1977.

3. Brown, F. M., "A Theorem Prover for Elementary Set Theory," IJCA15 Conference Proceedings, MIT, August 1977. Also the abstract is in the Workshop on Automatic Deduction Collected Abstracts, MIT, August 1977.

4. Brown, F. M., "Towards the Automation of Set Theory and its Logic," Artificial Intelligence Vol. 10, 1978.

5. Brown, F. M., "Towards the Automation of Mathematical Reasoning, Thesis for Ph. D., University of Edinbrugh 1977.

6. Brown, F. M. and Tarnlund, Sten-Ake, "Inductive Reasoning in Mathematics," IJCAI 5 Conference Proceedings, MIT, August 1977.

7. Brown, F. M. and Tranlund, Sten-Ake, "Inductive Reasoning on Recursive Equations" Artificial Intelligence, Vol. 12, 1979.

8. Brown, F. M., "An Automatic proof of the Completeness of Quantificational Logic," DAI Research Report 52, 1978.

9. Brown, F. M., "A Theorem Prover for Metatheory," De-artment of Computer Sciences, U.T. Austin, Technical Report #85 to appear in 4th Workshop on Automatic Theorem Proving Conf. Proc., Austin, Texas, 1979.

10. Brown, F. M., "A Sequent Calculus for Modal Quantificational Logic," 3rd AISB/GI Conference Proceedings, Hamburg, July 1978.

11. Prawitz, D., "An improved proof procedure" Theoria 26, 1960.

12. Bledsoe W. W. "Splitting and reductions hewristics in automatic theorem proving" Artifical Intelligence Vol. 2 1971.

13. Pastre D., Demonstration Automatique de Theorems en Theorie des Ensembles. These de 3eme cycle, Paris 1976.

14. Robinson J. A. A Machine-Oriented Logic Based on the Resolution Principle" J. ACM Vol. 12 Nov. 1965.