SET THEORY AND ITS SYMMETRIC LOGIC

Dr. Frank M. Brown

TR-83-13 August 1983

SET THEORY AND ITS SYMMETRIC LOGIC

Dr. Frank M. Brown
Dept. Computer Sciences
The University of Texas at Austin
Austin, Texas 78712
CS.BROWN@UTEXAS-20

1 INTRODUCTION

The SYMbolic EVALuator deduction system and the SYMMETRIC LOGIC rules are applied to proving theorems in classical and Ontological set theories. After briefly describing SYMEVAL and the SYMMETRIC LOCIC, we give one example proof from each set theory.

2 SYMEVAL

The SYMbolic EVALuator is a general interpreter of the Frege's quantificational logic[Frege] (ie. first order logic with schemators, but not higher order logic). Higher order logic is handled by axiomatizing it within first order logic. The actual symbols and inference rules of the logic are arbitrary as far as SYMEVAL is concerned. SYMEVAL evaluates a function value (f arg1...argN) in one of two ways. If the function f is not an FSUBRSYM then it applies the definitions, axioms, and rules about f to the result of evaluating each argument. However if f is an FSUBRSYM then only the first argument is evaluated before the f laws are applied. SYMEVAL's symbolic evaluation of expressions is thus somewhat analogous to LISP's[McCarthy] method of evaluating expression with the exception that SYMEVAL returns a "normalform" expression equal to the input expression whereas LISP returns the meaning of that "normalform" expression. Thus whereas (CONS(QUOTE A)(QUOTE B)) evaluates to its meaning: (A.B) in LISP, it SYMbolically EVALuates to the equal expression (QUOTE(A.B)). The evaluation to an equal expression allows SYMEVAL to handle quantified variables in a graceful manner. For example, whereas the evaluation of (CONS X Y) with unbound variables X and Y gives an error in LISP, its SYMbolic EVALuation results in the equal expression (CONS X Y). SYMEVAL is described in more detail in Brown[50].

3 THE SYMMETRIC LOGIC DEDUCTION SYSTEM

The SYMEVAL theorem prover now runs with a new logical system called SYMMETRIC LOGIC which treats universal and existential quantifiers in an analogous manner. For example as suggested by [Wang2] several years ago, SYMMETRIC LOGIC rewrites both of the following sentences to **(FOO A)**, after evaluating their subexpressions, when trying to prove them:

(ALL X (IMPLIES (EQUAL X A) (FOO X)))

(EX X (AND (EQUAL X A) (FOO X)))

Thus the essence of the SYMMETRIC LOGIC technique is to push quantifiers to the lowest scope possible in hopes of finding a way to eliminate them. Thus unlike the sequent calculus[Szabo,Brown1,3,4,6,12,25,Bibel2] and other logic systems[Bledsoe1,2] based on the Prawitz-Robinson Unification algorithm [Prawitz,Robinson], which essentially loses the scope of the existential quantifier during the skolemization process, SYMMETRIC LOGIC handles equalities very well indeed.

The power of a logic which handles equalities like this is very convincing, in an application domain dealing essentially with equations such as real algebra, logic programming, and language analysis. SYMMETRIC LOGIC is the synthesis of several earlier logic systems including the initial symmetric logic used by the real algebra rule package[Brown24], and the bind logic used by the logic programming and natural language rule packages[Brown26]. SYMMETRIC LOGIC is described in more detail in BROWN[50].

ABSTRACTION LOGIC

The SYMMETRIC LOGIC has a primitive symbol: LAMBDA for functional abstraction,(ie. for forming the werthverlauf of a function value) and a primitive symbol: AP for function application: AP.

(LAMBDA v(p v))
(AP g a)

Five Axioms and theorems similar to the set theory abstraction axioms used in [Brown4,6] are assumed for lambda conversion:

```
T1.(BADLAMBDAP X)    ==>       (AP S X) = NIL
T2.(NOT(BADLAMBDAP X))  ==>   (AP(LAMBDA X(p X))X) = (p X)
A3.                           (AP(LAMBDA X(p X))X) = (IF(BADLAMBDAP X)NIL(p X))
T4.(AND(NOT(BADLAMBDAP X))
      (NOT(BADLAMBDAP S))
      (NOT(LAMBDAP S))) ==> (AP S X) = (EQUAL S X)
A5.(AND(NOT(BADLAMBDAP S))
      (NOT(LAMBDAP S)))  ==> (AP S X) = (IF(BADLAMBDAP X)NIL(EQUAL S X))
```

One axiom for determining the equality of functions is assumed:

```
A6.(EQUAL(LAMBDA v(p v))(LAMBDA v(q v))) =
                (ALL X(EQUAL(AP(LAMBDA v(p v))X) (AP(LAMBDA v(p v))X) ))
```

One theorem (similar to T1 above) is assumed for use by the type system:

```
T7.(AP F X) ==> (NOT(BADLAMBDAP X))
```

Set theoretic abstaction: SET and ELEmenthood are defined as follows:

```
(SET v(p v)) = (LAMBDA v(p v))
(ELE x s) = (APPLY s x)
```

Thus every function is a set and every set is a function. Functional equality is of course EQUAL, whereas set equality is:

```
(SETEQUAL x y) = (ALL V(IFF(ELE V x)(ELE V y)))
```

The set theory axioms are neutral with respect to a classical or Lesniewskian set theory. For example, at least the following axiom should be added for Lesniewskian set theory: (EQUAL (SET v(EQUAL v X)) X)

4 TRACING PROOFS

As **SYMEVAL** recursively evaluates expressions tracing information is produced whenever a definition, axiom, or rule is applied. This information consists of three parts: 1. An input expression to which the axiom is being applied, called I. 2. The midterm expression produced by the application of the axiom, called M. 3. The output expression obtained by recursively evaluating the M expression, called O. Thus, a trace will generally be of the form:

```
I1:exp
M1:exp
   I2:exp
   M2:exp
      ...
   O2:exp
   I2:exp
   M2:exp
      ...
   O2:exp
O1:exp
```

where the numbers immediately following I,M,or O are the level at which that application takes place. At a given level number i, Oi and Mi are always associated with the preceeding Ii.

5 CLASSICAL SET THEORY

Axioms for LAMBDA abstraction are part of the SYMEVAL-QCL system and are described earlier. Set Theory is developed in terms of LAMBDA abstraction by first defineing set theoretic abstraction and elementhood in terms of LAMBDA abstraction and APPLYcation, and then by asserting which sets are good sets in the sense that they may be members of other sets.

EXAMPLE

SYMEVAL can prove that the Weiner-Kurtowski set theoretic definition of an ordered pair is in fact an ordered pair. This proof is obtaind without the use of any lemmas whatsoever, and in fact in the course of the proof SYMEVAL proves a number of interesting lemmas about the equality of unordered pairs and unit sets. The only other automatic proof of this theorem that we are aware of in the literature is the sequent calculas based proof in [Brown6] which assumed several lemmas about unordered pairs and unitsets. (That sequent calculas theorem prover could prove the lemmas that were assumed if they were explicitly given to it.)

In order to state the ordered pair theorem, quantifiers whose bound variables range over anything but bad sets are declared

```
(SETQ QUANTSYM(APPEND '(QALL QEX SET) QUANTSYM))
(VASSUME 'SET (REMOVE 'BADLAMBDAP UNIVERSE))
(VASSUME 'QALL (REMOVE 'BADLAMBDAP UNIVERSE))
(VASSUME 'QEX  (REMOVE 'BADLAMBDAP UNIVERSE))
```

and then the following definitions are made:

```
(DCLLQ(EQUAL (ELE X Y) (AP Y X))
      (EQUAL (SET X(SCH1 X)) (LAMBDA X(SCH1 X)))
      (EQUAL (QALL X(SCH1 X)) (ALL X(IF (BADLAMBDAP X) T (SCH1 X))))
      (EQUAL (QEX X(SCH2 X)) (EX X(AND (NOT(BADLAMBDAP X))(SCH2 X))))
      (EQUAL (EQUALSETS A B) (QALL X(IFF(ELE X A)(ELE X B))) )
      (EQUAL (UNITSET A) (SET X(EQUAL X A)) )
      (EQUAL (PAIRSET A B) (SET X (LOR(EQUAL X A)(EQUAL X B))) )
```

```
(EQUAL (ORDPAIRSET A B) (PAIRSET(UNITSET A)(PAIRSET A B)) )
```

Two axioms of set theory are assumed, namely that unit sets and unordered pairsets are not BADLAMBDAPs:

```
AX5:  (EQUAL(BADLAMBDAP(LAMBDA X(EQUAL X A))) NIL)
AX6:  (EQUAL(BADLAMBDAP(LAMBDA X(IF(EQUAL X A)T(EQUAL X B)))) NIL)
```

The ordered pair theorem: that two ordered pairs are equalsets iff their components are equal is now proven.

```
The expression to be recursively simplified is:
(QALL X (QALL Y (QALL U (QALL V (IFF (EQUALSETS (ORDPAIRSET X Y)
                                                (ORDPAIRSET U V))
                                     (AND (EQUAL X U)
                                          (EQUAL Y V)))))))
  I1:(ORDPAIRSET X Y)
     by use of: ORDPAIRSET
  M1:(PAIRSET (UNITSET X)
              (PAIRSET X Y))
    I2:(UNITSET X)
       by use of: UNITSET
    M2:(SET *1 (EQUAL *1 X))
    O2:(LAMBDA *1 (EQUAL *1 X))
    I2:(PAIRSET X Y)
       by use of: PAIRSET
    M2:(SET *2 (LOR (EQUAL *2 X)
                    (EQUAL *2 Y)))
    O2:(LAMBDA *2 (IF (EQUAL *2 X)
                      T
                      (EQUAL *2 Y)))
    I2:(PAIRSET (LAMBDA *1 (EQUAL *1 X))
                (LAMBDA *2 (IF (EQUAL *2 X)
                               T
                               (EQUAL *2 Y))))
       by use of: PAIRSET
    M2:(SET *3 (LOR (EQUAL *3 (LAMBDA *1 (EQUAL *1 X)))
                    (EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)
                                             T
                                             (EQUAL *2 Y))))))
    O2:(LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1 (EQUAL *1 X)))
                      T
                      (EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)
                                               T
                                               (EQUAL *2 Y))))))
  O1:(LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1 (EQUAL *1 X)))
                    T
                    (EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)
                                             T
                                             (EQUAL *2 Y))))))
  I1:(ORDPAIRSET U V)
     by use of: ORDPAIRSET
  M1:(PAIRSET (UNITSET U)
              (PAIRSET U V))
    I2:(UNITSET U)
       by use of: UNITSET
    M2:(SET *4 (EQUAL *4 U))
    O2:(LAMBDA *4 (EQUAL *4 U))
    I2:(PAIRSET U V)
       by use of: PAIRSET
    M2:(SET *5 (LOR (EQUAL *5 U)
                    (EQUAL *5 V)))
    O2:(LAMBDA *5 (IF (EQUAL *5 U)
```

```
                     T
                     (EQUAL *5 V)))
  I2:(PAIRSET (LAMBDA *4 (EQUAL *4 U))
              (LAMBDA *5 (IF (EQUAL *5 U)
                             T
                             (EQUAL *5 V))))
     by use of: PAIRSET
  M2:(SET *6 (LOR (EQUAL *6 (LAMBDA *4 (EQUAL *4 U)))
                  (EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
                                           T
                                           (EQUAL *5 V))))))
  O2:(LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4 (EQUAL *4 U)))
                    T
                    (EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
                                             T
                                             (EQUAL *5 V))))))
 O1:(LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4 (EQUAL *4 U)))
                   T
                   (EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
                                            T
                                            (EQUAL *5 V))))))
 I1:(EQUALSETS (LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1 (EQUAL *1 X)))
                              T
                              (EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)
                                                       T
                                                       (EQUAL *2 Y))))))
               (LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4 (EQUAL *4 U)))
                              T
                              (EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
                                                       T
                                                       (EQUAL *5 V)))))))
    by use of: EQUALSETS
 M1:(QALL *7 (IFF (ELE *7 (LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1
                                                   (EQUAL *1 X)))
                                         T
                                         (EQUAL *3 (LAMBDA
                                                   *2
                                                   (IF (EQUAL *2 X)
                                                       T
                                                       (EQUAL *2 Y)))))))
                  (ELE *7 (LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4
                                                   (EQUAL *4 U)))
                                         T
                                         (EQUAL *6 (LAMBDA
                                                   *5
                                                   (IF (EQUAL *5 U)
                                                       T
                                                       (EQUAL *5 V)))))))))
  I2:(IFF (IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
              T
              (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
                                       T
                                       (EQUAL *2 Y)))))
          (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
              T
              (EQUAL *7 (LAMBDA *5 (IF (EQUAL *5 U)
                                       T
                                       (EQUAL *5 V))))))
     by use of: IFF
  M2:(IF (IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
             T
             (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
```

```
                                      T
                                      (EQUAL *2 Y)))))
     (IF (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
             T
             (EQUAL *7 (LAMBDA *5 (IF (EQUAL *5 U)
                                         T
                                         (EQUAL *5 V)))))
         T NIL)
     (IF (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
             T
             (EQUAL *7 (LAMBDA *5 (IF (EQUAL *5 U)
                                         T
                                         (EQUAL *5 V)))))
         NIL T))
I3:(EQUAL (LAMBDA *1 (EQUAL *1 X))
          (LAMBDA *4 (EQUAL *4 U)))
   by use of: (LISPLINK SYMEQUAL)
M3:(ALL *8 (EQUAL (AP (LAMBDA *1 (EQUAL *1 X))
                      *8)
                  (AP (LAMBDA *4 (EQUAL *4 U))
                      *8)))
O3:(EQUAL U X)
I3:(EQUAL (LAMBDA *1 (EQUAL *1 X))
          (LAMBDA *5 (IF (EQUAL *5 U)
                         T
                         (EQUAL *5 V))))
   by use of: (LISPLINK SYMEQUAL)
M3:(ALL *9 (EQUAL (AP (LAMBDA *1 (EQUAL *1 X))
                      *9)
                  (AP (LAMBDA *5 (IF (EQUAL *5 U)
                                     T
                                     (EQUAL *5 V)))
                      *9)))
O3:NIL
I3:(EQUAL (LAMBDA *2 (IF (EQUAL *2 X)
                         T
                         (EQUAL *2 Y)))
          (LAMBDA *4 (EQUAL *4 U)))
   by use of: (LISPLINK SYMEQUAL)
M3:(ALL *10 (EQUAL (AP (LAMBDA *2 (IF (EQUAL *2 X)
                                      T
                                      (EQUAL *2 Y)))
                       *10)
                   (AP (LAMBDA *4 (EQUAL *4 U))
                       *10)))
O3:(IF (EQUAL U X)
       (EQUAL Y X)
       NIL)
I3:(EQUAL (LAMBDA *2 (IF (EQUAL *2 X)
                         T
                         (EQUAL *2 Y)))
          (LAMBDA *5 (IF (EQUAL *5 U)
                         T
                         (EQUAL *5 V))))
   by use of: (LISPLINK SYMEQUAL)
M3:(ALL *11 (EQUAL (AP (LAMBDA *2 (IF (EQUAL *2 X)
                                      T
                                      (EQUAL *2 Y)))
                       *11)
                   (AP (LAMBDA *5 (IF (EQUAL *5 U)
                                      T
                                      (EQUAL *5 V)))
```

```
                        *11)))
  O3:(IF (EQUAL X U)
         (IF (EQUAL Y U)
             (EQUAL V U)
             (EQUAL Y V))
         (IF (EQUAL X V)
             (IF (EQUAL Y V)
                 NIL
                 (EQUAL Y U))
             NIL))

 O2:(IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
        (EQUAL U X)
        (IF (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
                                     T
                                     (EQUAL *2 Y))))
            (IF (EQUAL U X)
                (IF (EQUAL Y X)
                    T
                    (EQUAL Y V))
                (IF (EQUAL X V)
                    (IF (EQUAL Y V)
                        NIL
                        (EQUAL Y U))
                    NIL))
            (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
                NIL
                (IF (EQUAL *7 (LAMBDA *5 (IF (EQUAL *5 U)
                                             T
                                             (EQUAL *5 V))))
                    NIL T))))
 I2:(QALL *7 (IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
                 (EQUAL U X)
                 (IF (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
                                              T
                                              (EQUAL *2 Y))))
                     (IF (EQUAL U X)
                         (IF (EQUAL Y X)
                             T
                             (EQUAL Y V))
                         (IF (EQUAL X V)
                             (IF (EQUAL Y V)
                                 NIL
                                 (EQUAL Y U))
                             NIL))
                     (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
                         NIL
                         (IF (EQUAL *7 (LAMBDA *5 (IF (EQUAL *5 U)
                                                      T
                                                      (EQUAL *5 V))))
                             NIL T)))))
   by use of: QALL
 M2:(ALL *12
        (IF (BADLAMBDAP *12)
            T
            (IF (EQUAL *12 (LAMBDA *1 (EQUAL *1 X)))
                (EQUAL U X)
                (IF (EQUAL *12 (LAMBDA *2 (IF (EQUAL *2 X)
                                              T
                                              (EQUAL *2 Y))))
                    (IF (EQUAL U X)
                        (IF (EQUAL Y X)
```

```
                                T
                                (EQUAL Y V))
                            (IF (EQUAL X V)
                                (IF (EQUAL Y V)
                                    NIL
                                    (EQUAL Y U))
                                NIL))
                        (IF (EQUAL *12 (LAMBDA *4 (EQUAL *4 U)))
                            NIL
                            (IF (EQUAL *12 (LAMBDA *5 (IF (EQUAL *5 U)
                                                          T
                                                          (EQUAL *5 V))))
                                NIL T))))))
    I3:(BADLAMBDAP (LAMBDA *1 (EQUAL *1 X)))
       by use of: AX5
    M3:NIL
    O3:NIL
    I3:(BADLAMBDAP (LAMBDA *2 (IF (EQUAL *2 X)
                                  T
                                  (EQUAL *2 Y))))
       by use of: AX6
    M3:NIL
    O3:NIL
    I3:(BADLAMBDAP (LAMBDA *4 (EQUAL *4 X)))
       by use of: AX5
    M3:NIL
    O3:NIL
    I3:(BADLAMBDAP (LAMBDA *5 (IF (EQUAL *5 X)
                                  T
                                  (EQUAL *5 V))))
       by use of: AX6
    M3:NIL
    O3:NIL
    I3:(BADLAMBDAP (LAMBDA *4 (EQUAL *4 X)))
       by use of: AX5
    M3:NIL
    O3:NIL
    I3:(BADLAMBDAP (LAMBDA *5 (IF (EQUAL *5 X)
                                  T
                                  (EQUAL *5 V))))
       by use of: AX6
    M3:NIL
    O3:NIL
   O2:(IF (EQUAL U X)
          (IF (EQUAL Y X)
              (EQUAL V X)
              (EQUAL Y V))
          NIL)
  O1:(IF (EQUAL U X)
         (IF (EQUAL Y X)
             (EQUAL V X)
             (EQUAL Y V))
         NIL)
  I1:(IFF (IF (EQUAL U X)
              (IF (EQUAL Y X)
                  (EQUAL V X)
                  (EQUAL Y V))
              NIL)
          (IF (EQUAL X U)
              (EQUAL Y V)
              NIL))
     by use of: IFF
```

```
 M1:(IF (IF (EQUAL U X)
            (IF (EQUAL Y X)
                (EQUAL V X)
                (EQUAL Y V))
            NIL)
        (IF (IF (EQUAL X U)
                (EQUAL Y V)
                NIL)
            T NIL)
        (IF (IF (EQUAL X U)
                (EQUAL Y V)
                NIL)
            NIL T))
  O1:T
  I1:(QALL V T)
     by use of: QALL
  M1:(ALL *21 (IF (BADLAMBDAP *21)
                  T T))
  O1:T
  I1:(QALL U T)
     by use of: QALL
  M1:(ALL *22 (IF (BADLAMBDAP *22)
                  T T))
  O1:T
  I1:(QALL Y T)
     by use of: QALL
  M1:(ALL *23 (IF (BADLAMBDAP *23)
                  T T))
  O1:T
  I1:(QALL X T)
     by use of: QALL
  M1:(ALL *24 (IF (BADLAMBDAP *24)
                  T T))
  O1:T

The result of recursive simplification is:
T
which is true. QED.

EX1=0/EX2=0/EX3=0/EX4=0/EX5=0/EX6=0/EX7=0/EX8=0/EX9=0/EX10=0/
ALL1=0/ALL2=0/ALL3=22/ALL4=0/ALL5=0/ALL6=10/ALL7=0/ALL8=0/ALL9=0/ALL10=0/
```

6 ONTOLOGY

Lesniewski's Ontology[Luschei,Henry] is a set theory which grew out of the traditions of Medieval logic. Its "sets" closely correspond to noun phrases, including names, fictitious names(eg. Pegasus) and more general nouns. Its "elementhood" predicate corresponds to to the intransitive verb IS in English, and more closely to the Latin EST.

EXAMPLE SYMEVAL can prove that '(Z X Y) is a permutation of '(X Y Z). In order to do this, we first define recursive ontological definitions of of the notion of a permutation:

```
(DCLLQ
(EQUAL(PERMSET L)(IF (EQUAL  L (NILSET))
                     NIL
                     (INSERTSET (CAR L)(PERMSET (CDR L)))))
(EQUAL(NILSET) (LAMBDA X (EQUAL X NIL)))
(EQUAL(CONSET A B)
      (LAMBDA X(EX Y(EX Z(AND(IS Y A)(AND(IS Z B)(EQUAL X(CONS Y Z)))))))
```

```
(EQUAL(INSERTSET X L)
      (IF(EQUAL L (NILSET))
         (CONS X NIL)
         (NOMINAL.OR(CONSET X L)
                    (LAMBDA Y(EX Z(AND(IS Z L)
                                      (IS Y(CONSET(CAR Z)
                                                  (INSERTSET X(CDR Z))))))))))
(EQUAL(NOMINAL.OR A B)
      (LAMBDA X (LOR(IS X A)(IS X B)))))


(SETQ TRACELIST '(NOMINAL.OR CONSET PERMSET INSERTSET))
```

A proof of the Ontological theorem:

```
      (IS (QUOTE(Z Y X)) (PERMSET(QUOTE(X Y Z))))
```

is given below. The proof was edited by deleting most traces less than level 2.

The expression to be recursively simplified is:

```
(IS (QUOTE (Z Y X))
    (PERMSET (QUOTE (X Y Z))))
  I1:(PERMSET (QUOTE (X Y Z)))
     by use of: PERMSET
  M1:(IF (EQUAL (QUOTE (X Y Z))
                (NILSET))
         NIL
         (INSERTSET (CAR (QUOTE (X Y Z)))
                    (PERMSET (CDR (QUOTE (X Y Z))))))
    I2:(PERMSET (QUOTE (Y Z)))
       by use of: PERMSET
    M2:(IF (EQUAL (QUOTE (Y Z))
                  (NILSET))
           NIL
           (INSERTSET (CAR (QUOTE (Y Z)))
                      (PERMSET (CDR (QUOTE (Y Z))))))
      I3:(PERMSET (QUOTE (Z)))
         by use of: PERMSET
      M3:(IF (EQUAL (QUOTE (Z))
                    (NILSET))
             NIL
             (INSERTSET (CAR (QUOTE (Z)))
                        (PERMSET (CDR (QUOTE (Z))))))
        I4:(PERMSET NIL)
           by use of: PERMSET
        M4:(IF (EQUAL NIL (NILSET))
               NIL
               (INSERTSET (CAR NIL)
                          (PERMSET (CDR NIL))))
        04:NIL
      03:(QUOTE (Z))
      I3:(INSERTSET (QUOTE Y)
                    (QUOTE (Z)))
         by use of: INSERTSET
      M3:(IF (EQUAL (QUOTE (Z))
                    (NILSET))
             (CONS (QUOTE Y)
                   NIL)
             (NOMINAL.OR (CONSET (QUOTE Y)
                                 (QUOTE (Z)))
                         (LAMBDA
                          *8
                          (EX *9 (AND (IS *9 (QUOTE (Z)))
```

```
                                    (IS *8 (CONSET (CAR *9)
                                                   (INSERTSET (QUOTE Y)
                                                              (CDR *9))))))))
       )
  O3:(LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
                            *20)
                     T
                     (EQUAL (QUOTE (Z Y))
                            *20)))
 O2:(LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
                           *20)
                    T
                    (EQUAL (QUOTE (Z Y))
                           *20)))
 I2:(INSERTSET (QUOTE X)
               (LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
                                      *20)
                               T
                               (EQUAL (QUOTE (Z Y))
                                      *20))))
   by use of: INSERTSET
 M2:(IF (EQUAL (LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
                                      *20)
                               T
                               (EQUAL (QUOTE (Z Y))
                                      *20)))
               (NILSET))
        (CONS (QUOTE X)
              NIL)
        (NOMINAL.OR
          (CONSET (QUOTE X)
                  (LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
                                         *20)
                                  T
                                  (EQUAL (QUOTE (Z Y))
                                         *20))))
          (LAMBDA *21
                  (EX *22 (AND (IS *22 (LAMBDA
                                         *20
                                         (IF (EQUAL (QUOTE (Y Z))
                                                    *20)
                                             T
                                             (EQUAL (QUOTE (Z Y))
                                                    *20))))
                               (IS *21 (CONSET (CAR *22)
                                               (INSERTSET (QUOTE X)
                                                          (CDR *22))))))))
 O2:(LAMBDA *61
            (IF (EQUAL *61 (QUOTE (X Y Z)))
                T
                (IF (EQUAL *61 (QUOTE (X Z Y)))
                    T
                    (IF (EQUAL *61 (QUOTE (Y X Z)))
                        T
                        (IF (EQUAL *61 (QUOTE (Y Z X)))
                            T
                            (IF (EQUAL *61 (QUOTE (Z X Y)))
                                T
                                (EQUAL *61 (QUOTE (Z Y X)))))))))
O1:(LAMBDA *61
           (IF (EQUAL *61 (QUOTE (X Y Z)))
               T
```

```
(IF (EQUAL *61 (QUOTE (X Z Y)))
    T
    (IF (EQUAL *61 (QUOTE (Y X Z)))
        T
        (IF (EQUAL *61 (QUOTE (Y Z X)))
            T
            (IF (EQUAL *61 (QUOTE (Z X Y)))
                T
                (EQUAL *61 (QUOTE (Z Y X))))))))
```

```
The result of recursive simplification is:
T
which is true. QED.

EX0=0/EX1=6/EX2=0/EX3=26/EX4=0/EX5=6/EX6=4/EX7=0/EX8=0/EX9=0/EX10=0/
ALL0=0/ALL1=0/ALL2=0/ALL3=1/ALL4=0/ALL5=0/ALL6=1/ALL7=0/ALL8=0/ALL9=0/ALL10=0/
```

7 CONCLUSION

The SYMMETRIC LOGIC has been applied to proving some theorems in set theory in order to test its ability as a general logic for deduction. We do not pretend that the SYMMETRIC LOGIC can yet prove as wide a range of theorems as the sequent calculas based set theory theorem provers[Brown4,6,Bledsoe1,2, Pastre]. However, we have shown that the SYMMETRIC LOGIC can prove theorems which have not been proven with sequent calculas based theorem provers.

I wish to thank my students who have worked on this project, particularly Song Park worked on the set theory examples.

8 REFERENCES

Bibel2 W. and Schreiber J. "Proof Search in a Gentzen like system of first order logic", INTERNATIONAL COMPUTING SYMPOSIUM 1975, North Holland, Amsterdam.

Bledsoe1 W.W. "Splitting and Reduction Heuristics in Automatic Theorem Proving" ARTIFICIAL INTELLIGENCE vol. 2, no.1 Spring 1971.

Bledsoe2 W.W. "Non Resolution Theorem Proving" ARTIFICIAL INTELLIGENCE Vol. 9, 1977.

Brown4,F.M., "A Theorem Prover for Elementary Set Theory", 5th INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, MIT, August 1977. Also the abstract is in the WORKSHOP ON AUTOMATIC DEDUCTION COLLECTED ABSTRACTS, MIT, August 1977.

Brown6,F.M., "Towards the Automation of Set Theory and its Logic", ARTIFICIAL INTELLIGENCE, Vol. 10, 1978.

Brown24,F.M. "A Deductive System for Real Algebra" TR 141, March 1980.

Brown26,F.M. "Computation with Automatic Theorem Provers", to appear in the proceedings of the NSF workshop on Logic Programming, Los Angeles, 1981.

Brown25,F.M "Experimental Logic and the Automatic Analysis of Algorithms" The Army Conference on the Applications of Artificial Intelligence to Battlefield Information Management" To appear in Spring 1984.

Frege Gottlob, "Begfiffschrift, a formula language, modeled upon that of arithmetic, for pure thought" 1879, in FROM FREGE TO GODEL, Harvard Univ. Press 1967.

Henry,D.P., MEDIEVAL LOGIC AND METAPHYSICS, Hutchinson & CO LTD, 1972.

Luschei,E. C., THE LOGICAL SYSTEMS OF LESNIEWSKI, 1962.

McCarthy, J. et. al. LISP 1.5 Programmer's Manual, MIT Press, 1966.

Pastre D. DEMONSTRATION AUTOMATIQUE DE THEOREMS EN THEORIE DES ENSEMBLES These de 3eme cycle, Paris VI, 1976.

Prawitz D. "An Improved Proof Procedure" THEORIA 26, 1960.

Robinson,J.A. "A machine oriented logic based on the resolution principle" J.ACM 12, 1965.

Szabo,M.E. ed. THE COLLECTED PAPERS OF GERHARD GENTZEN, North Holland, Amsterdam, 1969.

Wang2 Hao "On the long-range prospects of automatic theorem-proving", LECTURE NOTES IN MATHEMATICS: SYMPOSIUM ON AUTOMATIC DEMONSTRATION, held at Versailles/France 1968, Springer-Verlag 1970.