Series: Monographs in Computer Science
and Computer Applications
Nº 2/74

APPLICATIONS OF QUANTIFIER ELIMINATION TO MECHANICAL THEOREM PROVING - IMPLEMENTATION

by

Emmanuel Piseces Lopes Passos
and
Geraldo F. G. da Silveira

Computer Science Department - Rio Datacenter

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 209 — ZC-20
Rio de Janeiro — Brasil

APPLICATIONS OF QUANTIFIER ELIMINATION TO MECHANICAL THEOREM PROVING - IMPLEMENTATION

Emmanuel Piseces Lopes Passos
Assistent Professor
Computer Science Department
(Informática) PUC/RJ

and

Geraldo F.G. da Silveira
Associate Professor
Electrical Enginnering Department-PUC/RJ

Series Editor: Prof. S. M. dos Santos Julho/1974

ABSTRACT

The purpose of this paper is to implement by using LISP 1.5 functions defined in Galda & PASSOS [1] having as equipment IBM/370-165.

The formula transformations are special functions which can be utilized by any of the algorithms mentioned in [1].

The algorithm in detail is implemented for the theory of Densely Ordered Sets Without First or Last Element.

Appendix D contains a complet list of the program.

In this paper the authors intends, only, to present the implementation of the algorithms described in GALDA & PASSOS. Consequently the previous acquaintance with [1] is required to the understanding of the present paper.

Key words

Automatic Theorem Proving, Quantifier Elimination, LISP.

INTRODUCTION

We are concerned here with algorithms for proving theorems in certain first-order theories, and to program them having as main aim the field of automatic theorem proving. Functions are written in LISP 1.5 (ap - pendix A) since LISP has a simple syntax and is universally used in the field of Artificial Intelligence.

We describe the algorithm mentioned in [1] with some modification. We make explicit each transformation function (written in LISP) by means of a simplified notation. We also give a run example that contains several functions. We list (appendix B) a program which transform the given formulas into the prenex normal formula [(PNF) appendix B] . The external output is also explained (appendix C). In appendix D we give the listing of the main program TEOREMA.

Simplified Notation of the Graphical representation of LISP.

We assume the following graphical symbols and their associated meanings:

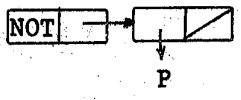
Internal description of the formula logical operators:

∀	has	as	internal	representation	A
∃	"	"	"	"	E
=>	"	"	"	"	IMP
<=>	"	"	"	"	IFF
V	"	"	"	"	OR
&	(∧)	"	"	"	AND
=	"	"	"	"	EQ
<	"	"	"	"	LT
¬	"	"	"	"	NOT

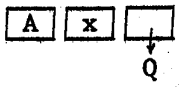
Formulas are internaly represented as a list where the first element is a quantifier, the second element is a variable and the third is a quantifier-free formula. For example:

a) ¬P 

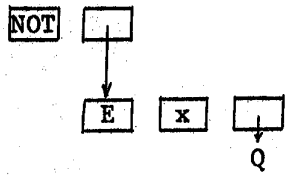
(notation not simplified)



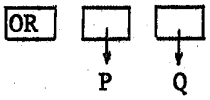
b) $(\forall x)(Q)$



c) $\neg(\exists x)Q$

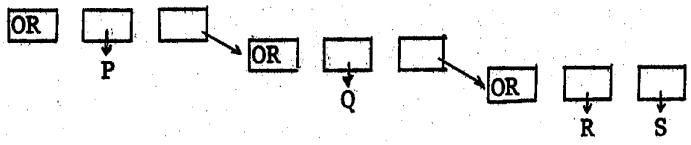


d) P v q



(any binary operators)

e) P v Q v R v S = P v (Q v (R v S)) associative



OUTPUT FORMAT

We construct a lisp-program transforming internal output of LISP 1.5 in order to use it for the output of the functions mentioned in [1]. For example, if we want to prove the theorem.

$$(E\ V_1)(V_2)(V_1 < V_2) \quad \text{(page. 12 of [2])}$$

the internal lisp-print will be

$$((E\ V_1\ (A\ V_2 (LT\ V_1\ V_2))))$$

but our-print (by using our own lisp printer) will be

$$(E\ V_1)\ (A\ V_2)\ (V_1 < V_2)$$

Our two lisp-functions used to obtain this printing are described in the appendix C.

Prenex Normal Formula (PNF)

The algorithm for the densely ordered sets assumes that the formulas are in PNF, therefore we need some manipulations to transform formulas into PNF. To do this we use a LISP program described in appendix A. This program has the following structure [The rules numbered I up to VI are found in page 85 [2]].

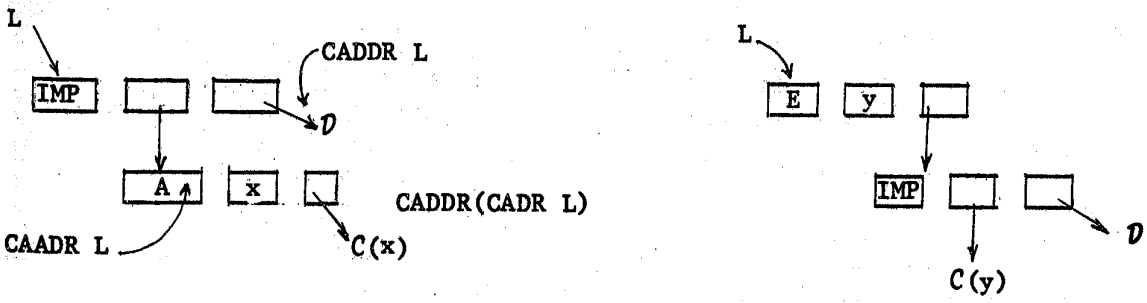
Rule I

$$((\forall x)\ C(x) \Rightarrow \mathcal{D}) \equiv (\exists y)(C\ (y) \Rightarrow \mathcal{D})$$

corresponding in our program to:

$$(IMP\ (A\ x\ C(x))\ \mathcal{D}) \equiv (E\ y\ (IMP\ C(y)\ \mathcal{D}))$$

The structure will be

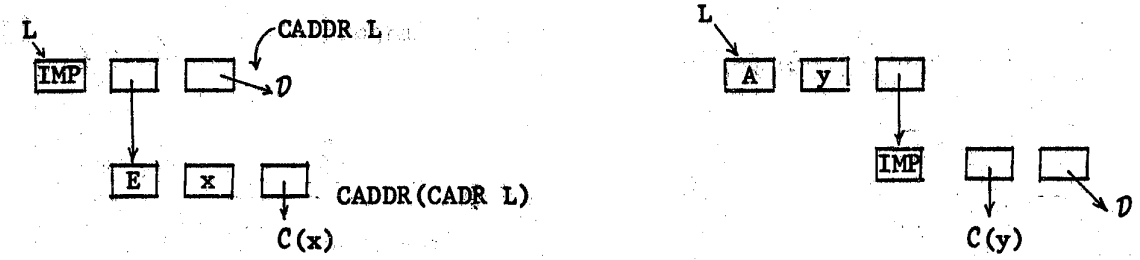


Rule II

$((\exists x)\ C(x) \Rightarrow \mathcal{D}) \equiv (\forall y)(C(y) \Rightarrow \mathcal{D})$

in our program:

(IMP (E x C(x)) $\mathcal{D}$) ≡ (A y (IMP C(y) $\mathcal{D}$))

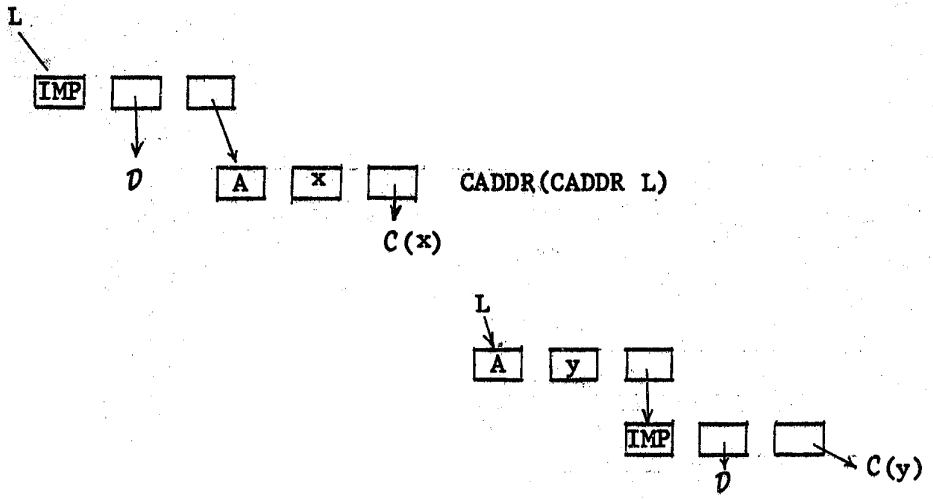


Rule III

$\mathcal{D} \Rightarrow (\forall x)\ C(x) \equiv (\forall y)(\mathcal{D} \Rightarrow C(y))$

in our program

(IMP $\mathcal{D}$ (A x C(x))) ≡ (A y (IMP $\mathcal{D}$ C(y)))

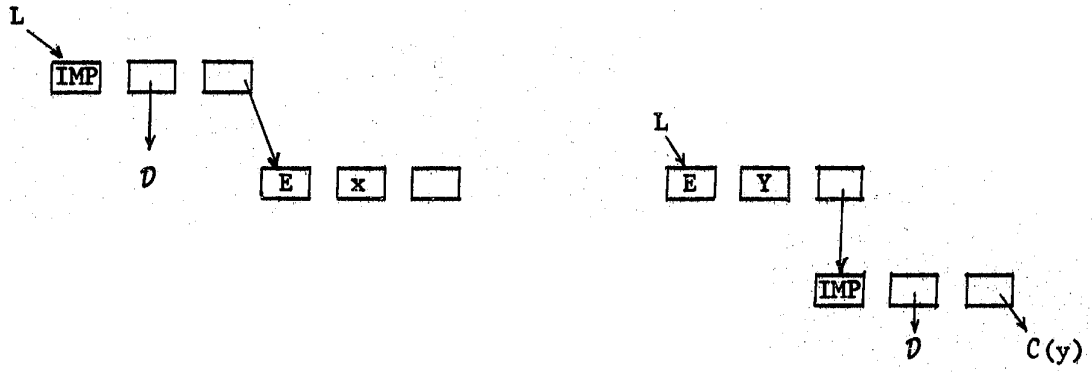


Rule IV

$\mathcal{D} \Rightarrow (\exists x)\, C(x) \equiv (\exists y)(\mathcal{D} \Rightarrow C(y))$

in our program

(IMP $\mathcal{D}$ (E x C(x))) ≡ (E y (IMP $\mathcal{D}$ C(y)))

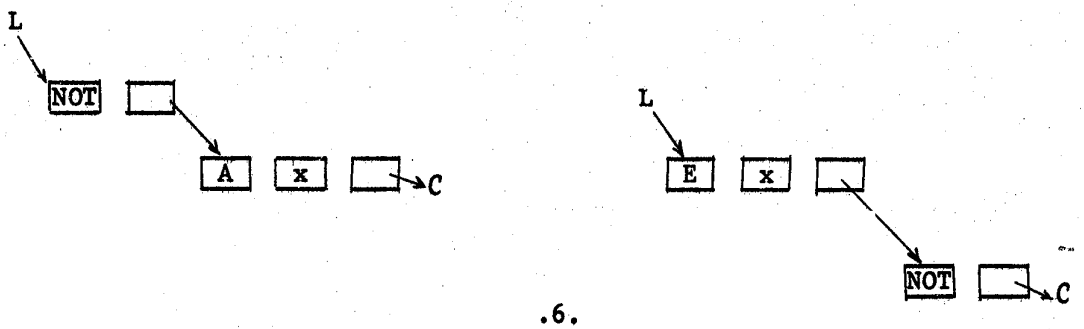


Rule V

$\sim(\forall x)C \equiv (\exists x) \sim C$

in our program

(NOT (A x C)) ≡ (E x (NOT C))

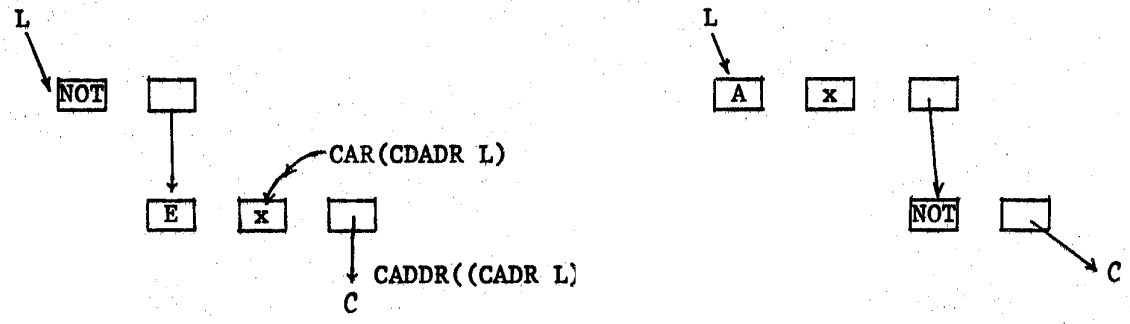


Rule VI

$\sim(\exists x)C \equiv (\forall x) \sim C$

in our program

(NOT (E x C)) ≡ (A x (NOT C))



A example will show how the PNF - program works. It is the example found in [2].

INPUT

((A x(IMP(EQU x x)(A y (IMP(LT x y)(NOT(A z(LT y z)))))))

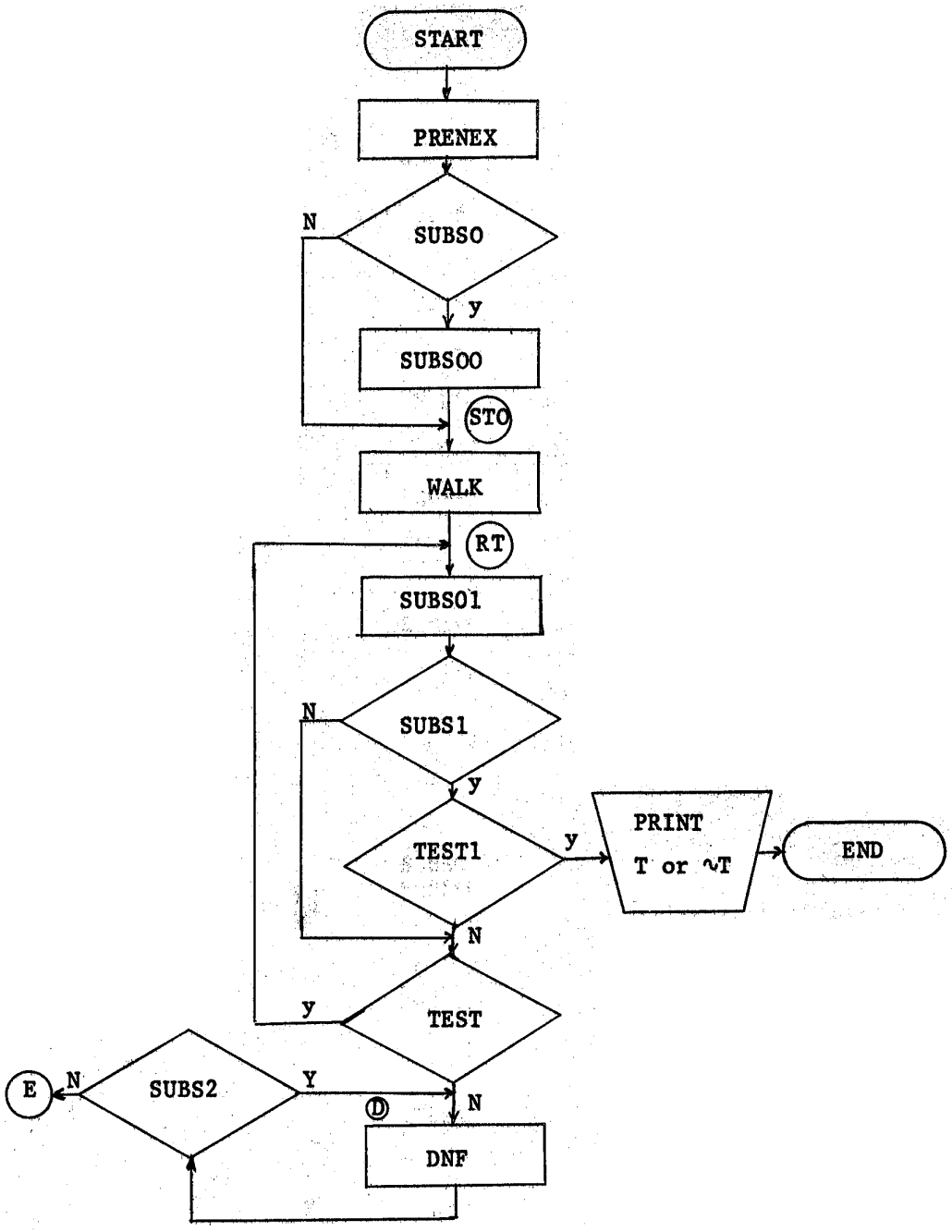
TRACE

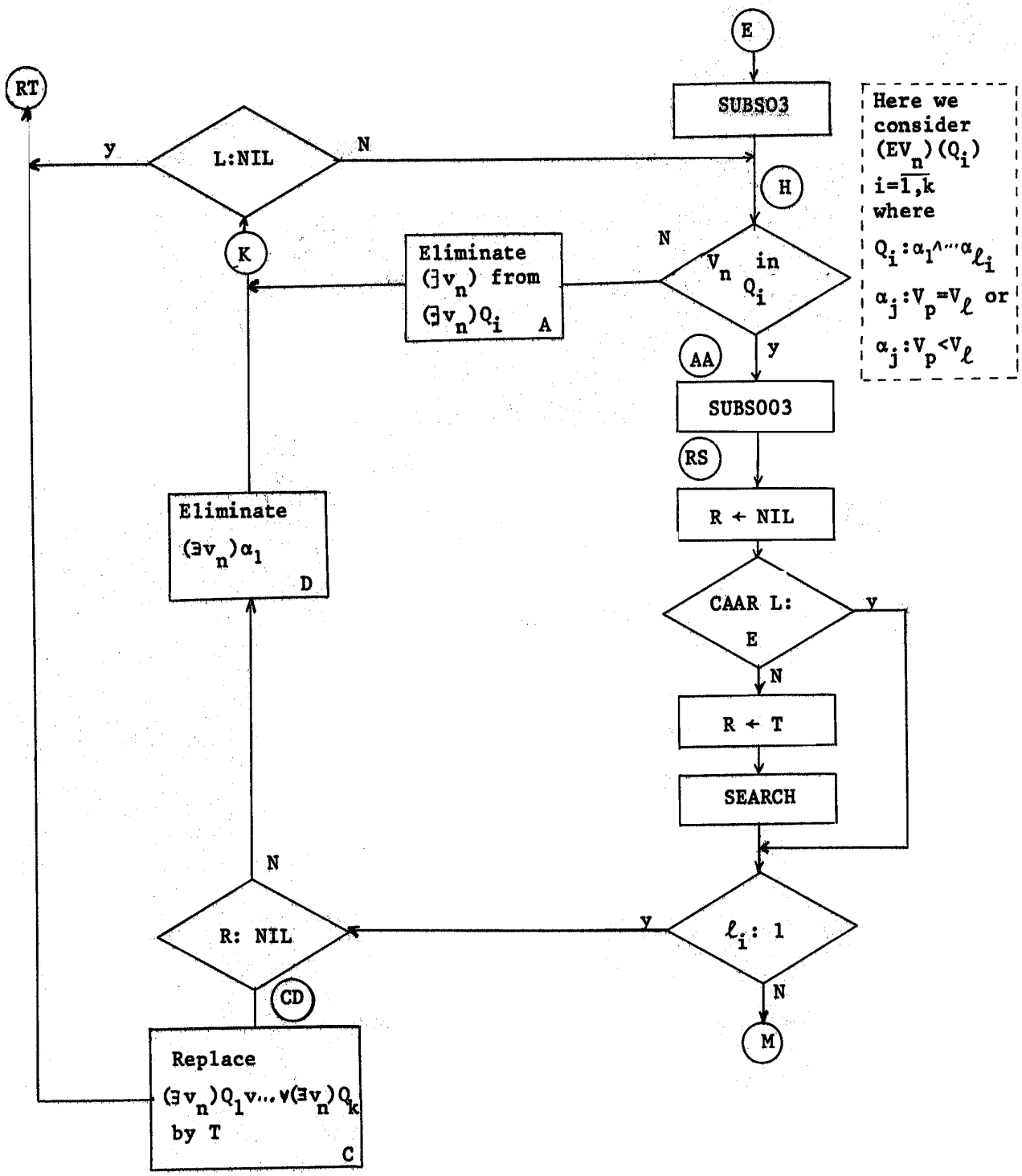
```
PRENEXO((A X (IMP (EQU X X) (A Y (IMP (LT X Y) (NOT (A Z (LT Y Z))))))))
PRENEXO((IMP (EQU X X) (A Y (IMP (LT X Y) (NOT (A Z (LT Y Z)))))))
PRENEXO((EQU X X))
PRENEXO (EQU X X)
PRENEXO((IMP (LT X Y) (NOT (A Z (LT Y Z)))))
PRENEXO((LT X Y))
PRENEXO(LT X Y)
PRENEXO((NOT (A Z (LT Y Z))))
PRENEXO((LT Y Z))
*PRENEX((NOT (A Z (LT Y Z))))
*PRENEX (E Z (NOT (LT Y Z)))
PRENEXO((E Z (NOT (LT Y Z))))
PRENEXO((NOT (LT Y Z)))
PRENEXO((LT Y Z))
*PRENEX (NOT (LT Y Z))
PRENEXO (NOT (LT Y Z))
PRENEXO (E Z (NOT (LT Y Z)))
*PRENEX((IMP (LT X Y) (E Z (NOT (LT Y Z)))))
PREN((Z (NOT (LT Y Z))))
PREN (NOT (LT Y B))
*PRENEX (E B (IMP (LT X Y) (NOT (LT Y B))))
PRENEXO (E B (IMP (LT X Y) (NOT (LT Y B))))
*PRENEX((IMP (EQU X X) (A Y (E B (IMP (LT X Y) (NOT (LT Y B)))))))
PREN((Y (E B (IMP (LT X Y) (NOT (LT Y B))))))
PREN (E B (IMP (LT X C) (NOT (LT C E))))
*PRENEX (A C (IMP (EQU X X) (E B (IMP (LT X C) (NOT (LT C B))))))
PRENEXO((A C (IMP (EQU X X) (E B (IMP (LT X C) (NOT (LT C B)))))))
PRENEXO((IMP (EQU X X) (E B (IMP (LT X C) (NOT (LT C B))))))
PRENEXO((EQU X X))
```

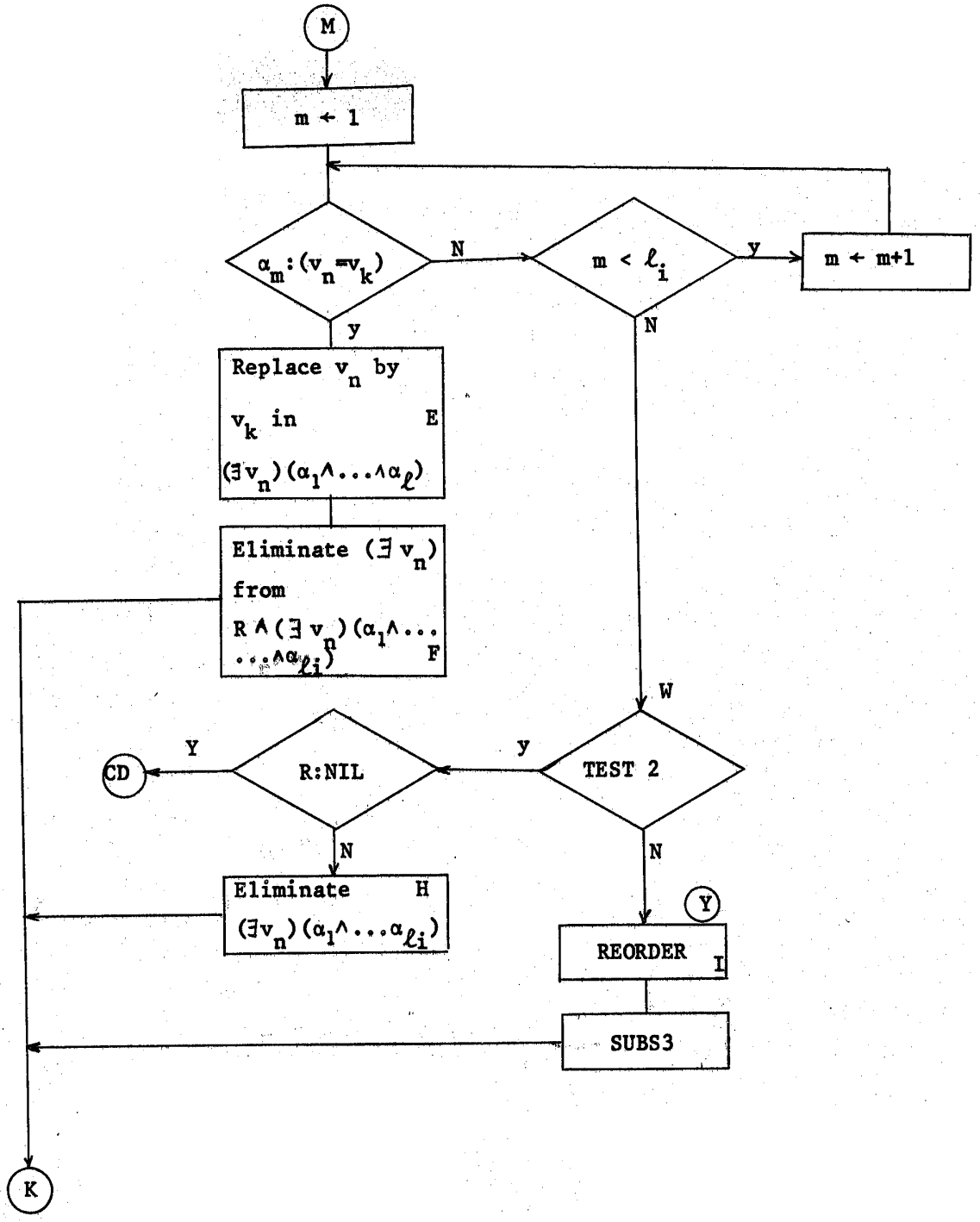
```
  PRENEXO((IMP (LT X C) (NOT (LT C B))))
  PRENEXO((LT X C))
  PRENEXO((NOT (LT C B)))
  PRENEXO((LT C B))
PRENEXO (LT C B)
*PRENEX((NOT (LT C B)))
PRENEXO (NOT (LT C B))
*PRENEX (IMP (LT X C) (NOT (LT C B)))
PRENEXO (IMP (LT X C) (NOT (LT C B)))
*PRENEX((IMP (EQU X X) (E B (IMP (LT X C) (NOT (LT C B))))))
PREN((B (IMP (LT X C) (NOT (LT C B)))))
PREN (IMP (LT X C) (NOT (LT C D)))
*PRENEX (E D (IMP (EQU X X) (IMP (LT X C) (NOT (LT C D)))))
PRENEXO((E D (IMP (EQU X X) (IMP (LT X C) (NOT (LT C D))))))
PRENEXO((IMP (EQU X X) (IMP (LT X C) (NOT (LT C D)))))
PRENEXO((EQU X X))
 PRENEXO((IMP (LT X C) (NOT (LT C D))))
 PRENEXO((LT X C))
  PRENEXO((NOT (LT C D)))
  PRENEXO((LT C D))
  *PRENEX (NOT (LT C D))
  PRENEXO (NOT (LT C D))
*PRENEX (IMP (LT X C) (NOT (LT C D)))
*PRENEX (IMP (EQU X X) (IMP (LT X C) (NOT (LT C D))))
PRENEXO (IMP (EQU X X) (IMP (LT X C) (NOT (LT C D))))
PRENEXO (E D (IMP (EQU X X) (IMP (LT X C) (NOT (LT C D)))))
PRENEXO (A C (E D (IMP (EQU X X) (IMP (LT X C) (NOT (LT C D))))))
PRENEXO (A X (A C (E D (IMP (EQU X X) (IMP (LT X C) (NOT (LT C D)))))))

FINAL OUTPUT

(A x(A y (E z(IMP (EQU (x x)(IMP (LT x y)(NOT(LT y z)))))))

TIME 186 MS
```

The listing of the PNF - program is in appendix B.

Modification of the original flowchart for Densely Ordered Sets [1,10]




E
SUBSO3
Here we consider $(EV_n)(Q_i)$ $i=\overline{1,k}$ where
$Q_i : \alpha_1 \wedge \cdots \alpha_{\ell_i}$
$\alpha_j : V_p = V_\ell$ or
$\alpha_j : V_p < V_\ell$
RT
y
L:NIL
N
H
K
Eliminate $(\exists v_n)$ from $(\exists v_n)Q_i$ A
N
V_n in Q_i
AA
y
SUBSO03
RS
R ← NIL
Eliminate $(\exists v_n)\alpha_1$ D
CAAR L: E
y
N
R ← T
SEARCH
N
R: NIL
y
ℓ_i: 1
CD
N
M
Replace $(\exists v_n)Q_1 v \ldots v (\exists v_n)Q_k$ by T C


M
m ← 1
$\alpha_m : (v_n = v_k)$
N
$m < \ell_i$
y
m ← m+1
y
N
Replace v_n by v_k in E
$(\exists v_n)(\alpha_1 \wedge \ldots \wedge \alpha_\ell)$
Eliminate $(\exists v_n)$ from
$R \wedge (\exists v_n)(\alpha_1 \wedge \ldots \wedge \alpha_{\ell_i})$ F
W
Y
CD
R:NIL
y
TEST 2
N
N
Eliminate H
$(\exists v_n)(\alpha_1 \wedge \ldots \alpha_{\ell_i})$
Y
REORDER I
SUBS3
K

REMARKS

- The data-input (formula) must be written in a internal form (found page 5).

- The PRINTER-program prints a formula anytime some modifications happens and the name of the function is printed.

- The flow-chart contains variables inside cycles, which represents labels in the main program called TEOREMA.

- TEOREMA is the main program like flow-chart.

- Listing of the lisp-functions (formula transformations) constructed by us are found in the appendix A.

- Following a little show of the tests.

Examples:

a) page 87, ex. 1 [5]

TEOREMA

((A X (IMP (EQU X X)(A Y (IMP(LT X Y)(NOT(A Z(LT Z Y)))))))
(internal formula)
(A X)((X = X) => (A Y)((X<Y) => ¬(A X)(Z<Y)))
(natural formula - constructed by us)

STEP 1

(A X)(A Y)(E Z)((X=X) ⇒ ((X<Y) ⇒ ¬(Z<Y))) - (PRENEX)

STEP 2

¬(E X)¬¬(E Y)¬(E Z)((X = X) ⇒ ((X < Y) ⇒¬(Z < Y))) - (SUBSO)

STEP 3

¬(E X)(E Y)¬(E Z)((X = X) ⇒((X < Y) ⇒¬(Z < Y))) - (SUBSOO)

STEP 4

¬(E X)(E Y)¬(E Z)(T ⇒ ((X < Y) ⇒ ¬(Z < Y))) - (SUBSO1)

STEP 5

¬(E X)(E Y)¬(E Z)((X < Y) ⇒ ¬(Z < Y)) - (SUBS1)

STEP 6

¬(E X)(E Y)¬(E Z)(¬(X < Y)| ¬(Z < Y)) - (DNF)

STEP 7

¬(E X)(E Y)¬(E Z)((X = Y) | (Y < X) | (Z = Y) | (Y < Z)) - (SUBS2)

STEP 8

¬(E X)(E Y)¬((E Z)(X = Y) | (E Z)(Y < X) | (E Z)(Z = Y) | (E Z)(Y < Z)) - (SUBS03)

STEP 9

¬(E X)(E Y)¬((X = Y) | (E Z)(Y < X) | (E Z)(Z = Y) | (E Z)(Y < Z)) - (A)

STEP 10

¬(E X)(E Y)¬((X = Y) | (Y < X) | (E Z)(Z = Y) | (E Z)(Y < Z)) - (A)

STEP 11

¬(E X)(E Y)¬ T - (C)

STEP 12

¬(E X)¬ T - (SUBS1)

STEP 13

T - (SUBS1)

TIME 778MS

b-)

TEOREMA

((E X(A Y(NOT (E Z(AND(OR(LT Y Y)(OR(EQU Z Y) (LT X Y)))(OR(LT X X)(NOT(EQU Z X)))))
))))

(E X)(A Y)¬(E Z)(((Y < Y) | (Z = Y) | (X < Y)) ε((X < X) | ¬(Z = X)))

(E X)(A Y)(A Z)¬(((Y < Y) | (Z = Y) | (X < Y)) ε ((X < X) | ¬(Z = X))) (PRENEX)

(E X)¬(E Y)¬¬(E Z)¬¬(((Y < Y) | (Z = Y) | (X < Y)) ε ((X < X) | ¬(Z = X))) (SUBSO)

(E X)¬(E Y)(E Z)(((Y < Y) | (Z = Y) | (X < Y)) ε ((X < X) | ¬(Z = X))) (SUBSOO)

(E X)¬(E Y)(E Z)((¬ T | (Z = Y) | (X < Y)) ε (¬T | ¬(Z = X))) (SUBSO1)

(E X)¬(E Y)(E Z)(((Z = Y) | (X < Y))ε ¬(Z = X)) (SUBS1)

(E X)¬(E Y)(E Z)(((Z = Y) ε¬(Z = X)) | ((X < Y)ε ¬(Z = X))) (DNF)

(E X)¬(E Y)(E Z)(((Z = Y) ε ((Z < X) | (X < Z))) | ((X < Y) ε ((Z < X) | (X < Z)))) (SUBS2)

(E X)¬(E Y)(E Z)(((Z = Y) ε (Z < X)) | ((Z = Y) ε (X < Z)) | ((X < Y) ε (Z < X)) | ((X < Y) ε (
X < Z))) (DNF)

(E X)¬(E Y)((E Z)((Z = Y) ε (Z < X)) | (E Z)((Z = Y) ε (X < Z)) | (E Z)((X < Y) ε (Z < X)) | (E
Z)((X < Y) ε (X < Z))) (SUBSO3)

(E X)¬(E Y)((E Z)((Y = Y) ε (Y < X)) | (E Z)((Z = Y) ε (X < Z)) | (E Z)((X < Y) ε (Z < X)) | (E
Z)((X < Y) ε (X < Z))) (E)

(E X)¬(E Y)(((Y = Y) ε (Y < X)) | (E Z)((Z = Y) ε (X < Z)) | (E Z)((X < Y) ε (Z < X)) | (E Z)((
X < Y) ε (X < Z))) (F)

(E X)¬(E Y)(((Y = Y) ε (Y < X)) | (E Z)((Y = Y) ε (X < Y)) | (E Z)((X < Y) ε (Z < X)) | (E Z)((X < Y) ε (X < Z))) (E)

(E X)¬(E Y)(((Y = Y) ε (Y < X)) | ((Y = Y) ε (X < Y)) | (E Z)((X < Y) ε (Z < X)) | (E Z)((X < Y) ε (X < Z))) (F)

(E X)¬(E Y)(((Y = Y) ε (Y < X)) | ((Y = Y) ε (X < Y)) | ((X < Y) ε (E Z)(Z < X)) | (E Z)((X < Y) ε (X < Z))) (SUBS003)

(E X)¬(E Y)(((Y = Y) ε (Y < X)) | ((Y = Y) ε (X < Y)) | (X < Y) | (E Z)((X < Y) ε (X < Z))) (D)

(E X)¬(E Y)(((Y = Y) ε (Y < X)) | ((Y = Y) ε (X < Y)) | (X < Y) | ((X < Y) ε (E Z)(X < Z))) (SUBS003)

(E X)¬(E Y)(((Y = Y) ε (Y < X)) | ((Y = Y) ε (X < Y)) | (X < Y) | (X < Y)) (D)

(E X)¬(E Y)((T ε (Y < X)) | (T ε (X < Y)) | (X < Y) | (X < Y)) (SUBS01)

(E X)¬(E Y)((Y < X) | (X < Y) | (X < Y) | (X < Y)) (SUBS1)

(E X)¬((E Y)(Y < X) | (E Y)(X < Y) | (E Y)(X < Y) | (E Y)(X < Y)) (SUBS03)

(E X)¬T (C)

¬T (SUBS1)

TIME 2166MS,...

c) page 12 [1]

```
TEOREMA
((E V1 (A V2 (LT V1 V2))))

(E V1)(A V2)(V1 < V2)

(E V1)¬(E V2)¬(V1 < V2)          (SUBS0)

(E V1)¬(E V2)((V1 = V2) | (V2 < V1))          (SUBS2)

(E V1)¬((E V2)(V1 = V2) | (E V2)(V2 < V1))          (SUBS03)

(E V1)¬T          (C)

¬T          (SUBS1)

TIME    212MS, ...
```

d) page 12 [1]

```
TEOREMA
((NOT (A V1 (E V2 (AND (LT V1 V2) (LT V2 V1))))))

¬(A V1)(E V2)((V1 < V2) ε (V2 < V1))

(E V1)(A V2)¬((V1 < V2) ε (V2 < V1))          (PRENEX)

(E V1)¬(E V2)¬¬((V1 < V2) ε (V2 < V1))          (SUBS0)

(E V1)¬(E V2)((V1 < V2) ε (V2 < V1))          (SUBS00)

(E V1)¬(V1 < V1)          (SUBS3)

(E V1)¬¬T          (SUBS01)

T          (SUBS1)

TIME    199MS, ...
```

e) page 5 [1]

TEOREMA
((A V1 (NOT (LT V1 V1))))

(A V1) ¬(V1 < V1)

¬(E V1)¬¬(V1 < V1) (SUBSO)

¬(E V1)(V1 < V1) (SUBSOO)

¬(E V1)¬T (SUBSO1)

T (SUBS1)
TIME 126MS,...

f) page 87, ex.2 [5]

TEOREMA
((IMP (LT X Y) (E Y (IMP (EQU Y Y) (IMP (E X (EQU X X)) (EQU Y Y))))))

((X < Y) => (E Y)((Y = Y) => ((E X)(X = X) => (Y = Y))))

(E Y)(A X)((X < Y) => ((Y = Y) => ((X = X) => (Y = Y)))) (PRENEX)

(E Y)¬(E X)¬((X < Y) => ((Y = Y) => ((X = X) => (Y = Y)))) (SUBSO)

(E Y)¬(E X)¬((X < Y) = (T => (T => T))) (SUBSO1)

(E Y)T (SUBS1)

T (SUBS1)

TIME 386MS,...

T

g) suggested by Pietrzkowski

```
TEOREMA
((E X (E Y (E Z (AND (LT Z Y) (AND (LT Y X) (LT X Z)))))))

(E X)(E Y)(E Z)((Z< Y) ε (Y < X) ε (X < Z))

(E X)(E Y)((Y < X) ε (E Z)(Z < Y) ε (X < Z))          (SUBS003)

(E X)(E Y)((Y < X) ε (E Z)(X < Z) ε (Z < Y))          (I)

(E X)(E Y)((Y < X) ε (X < Y))          (SUBS3)

(E X)(E Y)((X < Y) ε (Y < X))          (I)

(E X)(X < X)          (SUBS3)

(E X)¬T          (SUBS01)

¬T          (SUBS1)
TIME     406MS,
```

h) suggested by Pietrzkowski

```
TEOREMA
((E X (E Y (E Z (AND (LT Z Y) (AND (LT Y X) (LT Z X)))))))

(E X)(E Y)(E Z)((Z < Y) ε (Y < X) ε (Z < X))

(E X)(E Y)((Y < X) ε (E Z)(Z < Y) ε (Z < X))          (SUBS003)
(E X)(E Y)(Y < X)    (H)
(E X)T          (C)
T          (SUBS1)
TIME     276MS,                    .19.
```

References

1. Galda & Passos, E.P.L. - Applications of Quantifier Elimination to Mechanical Theorem Proving - Monographs in Computer Science and Computer Applications nº 6/72 - PUC/Rio de Janeiro.

2. McCarthy, J. et al., LISP 1.5 Programmer's Manual, The M.I.T. Press Cambridge, 1965.

3. Bolce, J.F., LISP/360 University of Waterloo - Waterloo, Ontario, 1968.

4. Weissman, C., LISP 1.5 PRIMER, Dickenson Publishing Company, California 1967.

5. Mendelson - Introduction to Mathematical Logic-Van Nostrand - 1964.

6. Passos, E.P.L. - Introduction to Mechanical Theorem Proving - Monographs in Computer Science and Computer Applications nº 10/71 - PUC/Rio de Janeiro.

7. Santos, S.M. and Millan, M.R. - Automatic Proofs for Theorems on Predicate Calculus - Monographs in Computer Science and Computer Application nº 8/72 - PUC/Rio de Janeiro.

8. References in Passos & Galda [1].

APENDIX A

```
DEFINE((
(WALK (LAMBDA (TOP L)
(COND ((ATOM (CAR L)) L)
      ((NULL (CDDAR L))
       (COND ((NOT (EQ (CAADR (CAR L))(QUOTE E))) TOP)
             (T (PROG2 (SETQ TOP (STACK TOP L))
                       (WALK TOP (CDDR (CADAR L)))))))
      ((NOT (EQ (CAAR L)(QUOTE E))) TOP)
      (T (PROG2 (SETQ TOP (STACK TOP L))
                (WALK TOP (CDDAR L)))))
      ))
DEFINE((
(STACK (LAMBDA (TOP X)
(PROG (P)
      (SETQ P (LIST NIL))
      (RPLACD P TOP)
      (RPLACA P X)
      (SETQ TOP P)
      (RETURN TOP))))
      ))
DEFINE((
(SEARCH (LAMBDA (E)
(COND ((EQ (CAAR E)(CAAR (CDDAR E)))
                (SEARCH (CDDAR E)))
      (T E))))
      ))
DEFINE((
(SUBS0 (LAMBDA (L)
(COND ((EQ (CAR L)(QUOTE A))
         (LIST (QUOTE NOT)(LIST (QUOTE E)(CADR L)
            (LIST (QUOTE NOT)(SUBS0 (CADDR L))))))
      ((EQ (CAR L)(QUOTE E))
         (LIST (QUOTE E)(CADR L)(SUBS0 (CADDR L))))
      ((EQ (CAR L)(QUOTE NOT))
         (LIST (QUOTE NOT)(SUBS0 (CADR L))))
      (T L))))
      ))
DEFINE((
(SUBS00 (LAMBDA (L)
(COND ((ATOM L) L)
      ((NULL (CDDR L))
  (COND ((ATOM (CADR L)) L)
        (T (COND ((NULL (CDR (CDADR L)))(SUBS00 (CAR (CDADR L))))
                 (T (LIST (QUOTE NOT)(SUBS00 (CADR L))))))))
      (T (LIST (CAR L)(SUBS00 (CADR L))(SUBS00 (CADDR L)))))
      ))
DEFINE((
(SUBS01 (LAMBDA (L)
(COND ((ATOM L) L)
      ((NULL (CDDR L))(LIST (QUOTE NOT)(SUBS01 (CADR L))))
      ((AND (ATOM (CADR L))(ATOM (CADDR L)))
       (COND ((EQ (CADR L)(CADDR L))
```

```
                (COND ((EQ (CAR L)(QUOTE EQU)) T)
                      ((EQ (CAR L)(QUOTE IT))(LIST (QUOTE NOT) T))
                      (T L)))
               (T L)))
        (T (LIST (CAR L)(SUBS01 (CADR L))(SUBS01 (CADDR L)))))))
        ))
DEFINE((
(SUBS1 (LAMBDA (L)
   (PROG (E)
      (SETQ E (LIST NIL))
   A  (COND ((EQUAL E L)(RETURN L)))
      (SETQ E L)
      (SETQ L (*SUBS1 L))
      (GO A))))
      ))
DEFINE((
(*SUBS1 (LAMBDA (L)
(COND ((ATOM L) L)
      ((NULL (CDDR L))
  (COND ((ATOM (CADR L)) L)
        (T (COND ((NULL (CDR (CDADR L)))(*SUBS1 (CAR (CDADR L))))
                 (T (LIST (QUOTE NOT)(*SUBS1 (CADR L))))))))
      ((EQ (CADR L) T)
       (COND ((EQ (CAR L)(QUOTE OR)) T)
             ((OR (EQ (CAR L)(QUOTE AND))(EQ (CAR L)(QUOTE IMP))
                  (EQ (CAR L)(QUOTE IFF)))(*SUBS1 (CADDR L)))))
      ((EQ (CADDR L) T)
       (COND ((OR (EQ (CAR L)(QUOTE OR))(EQ (CAR L)(QUOTE IMP))
                  (EQ (CAR L)(QUOTE E))) T)
             ((OR (EQ (CAR L)(QUOTE AND))(EQ (CA )(QUOTE IFF)))
                  (*SUBS1 (CADR L)))))
      ((EQUAL (CADR L)(LIST (QUOTE NOT) T))
       (COND ((EQ (CAR L)(QUOTE IMP)) T)
             ((EQ (CAR L)(QUOTE OR))(*SUBS1 (CADDR L)))
             ((EQ (CAR L)(QUOTE AND))(LIST (QUOTE NOT) T))
             ((EQ (CAR L)(QUOTE IFF))
                  (LIST (QUOTE NOT)(*SUBS1 (CADDR L))))))
      ((EQUAL (CADDR L)(LIST (QUOTE NOT) T))
       (COND ((EQ (CAR L)(QUOTE OR))(*SUBS1 (CADR L)))
             ((OR (EQ (CAR L)(QUOTE AND))(EQ (CAR L)(QUOTE E)))
                  (LIST (QUOTE NOT) T))
             ((OR (EQ (CAR L)(QUOTE IMP))(EQ (CAR L)(QUOTE IFF)))
                  (LIST (QUOTE NOT)(*SUBS1 (CADR L))))))
      (T (LIST (CAR L)(*SUBS1 (CADR L))(*SUBS1 (CADDR L)))))))
      ))
DEFINE((
(DNF (LAMBDA (L)
      (PROG (E)
      (SETQ E (LIST NIL))
      (SETQ L (IMPEL L))
   A  (COND ((EQUAL E L)(GO B)))
      (SETQ E L)
      (SETQ L (SUBS00 (DE*MORGAN L)))
      (GO A)
```

```
   B  (SETQ L (*DNF L))
   C  (COND ((EQUAL E L)(GO D)))
      (SETQ E L)
      (SETQ L (*DNF L))
      (GO C)
   D  (COND ((OR (EQ (CAR L)(QUOTE E))(EQ (CAR L)(QUOTE A)))
               (RETURN (LIST (CAR L)(CADR L)(*DNFU (DNF*OR (CADDR L))))))
              (T (RETURN (*DNFU (DNF*OR L))))))
      ))
DEFINE((
(IMPEL (LAMBDA (L)
(COND ((ATOM L) L)
      ((NULL (CDDR L))(LIST (QUOTE NOT)(IMPEL (CADR L))))
      ((EQ (CAR L)(QUOTE IMP))(LIST (QUOTE OR)
(LIST (QUOTE NOT)(IMPEL (CADR L)))(IMPEL (CADDR L))))
      ((EQ (CAR L)(QUOTE IFF))(LIST (QUOTE AND)
(LIST (QUOTE OR)(LIST (QUOTE NOT)(IMPEL (CADR L)))(IMPEL (CADDR L)))
(LIST (QUOTE OR)(LIST (QUOTE NOT)(IMPEL (CADDR L)))(IMPEL (CADR L)))))
      (T (LIST (CAR L)(IMPEL (CADR L))(IMPEL (CADDR L)))))))
      ))
DEFINE((
(DE*MORGAN (LAMBDA (L)
(COND ((ATOM L) L)
      ((NULL (CDDR L))
       (COND ((ATOM (CADR L)) L)
             (T (COND ((EQ (CAADR L)(QUOTE OR))
(LIST (QUOTE AND)(LIST (QUOTE NOT)(DE*MORGAN (CAR (CDADR L))))
                   (LIST (QUOTE NOT)(DE*MORGAN (CADR (CDADR L))))))
                      ((EQ (CAADR L)(QUOTE AND))
(LIST (QUOTE OR)(LIST (QUOTE NOT)(DE*MORGAN (CAR (CDADR L))))
                  (LIST (QUOTE NOT)(DE*MORGAN (CADR (CDADR L))))))
                      (T (LIST (QUOTE NOT)(DE*MORGAN (CADR L)))))))))
      (T (LIST (CAR L)(DE*MORGAN (CADR L))(DE*MORGAN (CADDR L)))))))
      ))
DEFINE((
(*DNF (LAMBDA (L)
(COND ((ATOM L) L)
      ((NULL (CDDR L))(LIST (QUOTE NOT)(*DNF (CADR L))))
      ((EQ (CAR L)(QUOTE AND))
       (COND ((AND (EQ (CAADR L)(QUOTE OR))
                   (EQ (CAR (CADDR L))(QUOTE OR)))
(LIST (QUOTE OR)
(*DNF (LIST (QUOTE AND)(CAR (CDADR L))(CADR (CADDR L))))
(LIST (QUOTE OR)
(*DNF (LIST (QUOTE AND)(CAR (CDADR L))(CADDR (CADDR L))))
(LIST (QUOTE OR)
(*DNF (LIST (QUOTE AND)(CADR (CDADR L))(CADR (CADDR L))))
(*DNF (LIST (QUOTE AND)(CADR (CDADR L))(CADDR (CADDR L)))))))
             ((EQ (CAADR L)(QUOTE OR))(LIST (QUOTE OR)
(*DNF (LIST (QUOTE AND)(CAR (CDADR L))(CADDR L)))
(*DNF (LIST (QUOTE AND)(CADR (CDADR L))(CADDR L)))))
             ((EQ (CAR (CADDR L))(QUOTE OR))(LIST (QUOTE OR)
(*DNF (LIST (QUOTE AND)(CADR L)(CADR (CADDR L))))
```

```
(*DNF (LIST (QUOTE AND)(CADR L)(CADDR (CADDR L))))))
             (T (LIST (QUOTE AND)(*DNF (CADR L))(*DNF (CADDR L))))))
      (T (LIST (CAR L)(*DNF (CADR L))(*DNF (CADDR L)))))))
      ))
DEFINE((
(*DNF0 (LAMBDA (L)
(COND ((ATOM L) L)
      ((EQ (CAR L)(QUOTE OR))
        (LIST (QUOTE OR)(DNF*AND (CADR L))(*DNF0 (CADDR L))))
      (T (COND ((EQ (CAR L)(QUOTE AND))(DNF*AND L))
               (T L))))))
      ))
DEFINE((
(DNF*OR (LAMBDA (L)
(COND ((EQ (CAR L)(QUOTE OR))
       (COND ((EQ (CAADR L)(QUOTE OR))
              (COND ((EQ (CAAR (CDADR L))(QUOTE OR))
                     (DNF*OR (LIST (QUOTE OR)(DNF*OR (CADR L))
                             (DNF*OR (CADDR L)))))
                    (T (DNF*OR (REPLACE L)))))
             (T (LIST (QUOTE OR)(CADR L)(DNF*OR (CADDR L))))))
      (T L))))
      ))
DEFINE((
(DNF*AND (LAMBDA (L)
(COND ((EQ (CAR L)(QUOTE AND))
       (COND ((EQ (CAADR L)(QUOTE AND))
              (COND ((EQ (CAAR (CDADR L))(QUOTE AND))
                     (DNF*AND (LIST (QUOTE AND)(DNF*AND (CADR L))
                             (DNF*AND (CADDR L)))))
                    (T (DNF*AND (REPLACE L)))))
             (T (LIST (QUOTE AND)(CADR L)(DNF*AND (CADDR L))))))
      (T L))))
      ))
DEFINE((
(REPLACE (LAMBDA (L)
   (PROG (E X)
      (SETQ E L)
      (SETQ X (CDDAR (SEARCH (CDR L))))
      (SETQ L (CADR L))
      (RPLACA (CDR E)(CAR X))
      (RPLACA X E)
      (RETURN L))))
      ))
DEFINE((
(SUBS2 (LAMBDA (L)
   (PROG ( )
      (SETQ L (*SUBS2 L))
      (COND ((OR (EQ (CAR L)(QUOTE E))
                 (EQ (CAR L)(QUOTE A)))
    (RETURN (LIST (CAR L)(CADR L)(*DNF0 (DNF*OR (CADDR L))))))
            (T (RETURN (*DNF0 (DNF*OR L)))))))
      ))
DEFINE((
```

```
(*SUBS2 (LAMBDA (L)
(COND ((ATOM L) L)
      ((NULL (CDDR L))
       (COND ((ATOM (CADR L)) L)
             ((AND (ATOM (CAR (CDADR L)))(ATOM (CADR (CDADR L))))
              (COND ((EQ (CAADR L)(QUOTE EQU))
(LIST (QUOTE OR)(LIST (QUOTE LT)(CAR (CDADR L))(CADR (CDADR L)))
                    (LIST (QUOTE LT)(CADR (CDADR L))(CAR (CDADR L)))))
                    ((EQ (CAADR L)(QUOTE LT))
(LIST (QUOTE OR)(LIST (QUOTE EQU)(CAR (CDADR L))(CADR (CDADR L)))
                    (LIST (QUOTE LT)(CADR (CDADR L))(CAR (CDADR L)))))
                    (T L)))
             (T (LIST (QUOTE NOT)(*SUBS2 (CADR L))))))
      (T (LIST (CAR L)(*SUBS2 (CADR L))(*SUBS2 (CADDR L)))))
      ))
DEFINE((
(SUBSU3 (LAMBDA (X L)
(COND ((OR (ATOM L)(NULL (CDDR L)))(LIST (QUOTE E) X L))
      ((EQ (CAR L)(QUOTE OR))
       (LIST (QUOTE OR)(LIST (QUOTE E) X (CADR L))(SUBSU3 X (CADDR L))))
      (T (LIST (QUOTE E) X L)))))
      ))
DEFINE((
(SUBSU03 (LAMBDA (U L Q)
(COND ((ATOM (CAR Q))(COND ((EQ (CADR L)(CAR Q)) L)
                           (T (CAR Q))))
      ((EQ (CAAR Q)(QUOTE AND))
       (COND ((MEMBER (CADR L)(CADAR Q))(SUBSU03 U L (CDDAR Q)))
             (T (PROG2 (PROG2 (SETQ U (CADAR Q))(RPLACA Q (CAR (CDDAR Q)
                       )))(LIST (QUOTE AND) U (SUBSU03 U L Q))))))
      (T (COND ((MEMBER (CADR L)(CAR Q)) L)
               (T (COND ((EQ (CAR (CADDR L))(QUOTE AND))
                         (PROG2 (PROG2 (SETQ U (SEARCH (CDDR L)))
                                       (RPLACA U (CADAR U)))
                                (LIST (QUOTE AND)(CAR Q) L)))
                        (T (CAR Q)))))))))
      ))
DEFINE((
(TEST2 (LAMBDA (X L)
   (OR (NOT (TESTF X L))(TESTS X L))))
(TESTF (LAMBDA (X L)
(COND ((EQ (CAAR L)(QUOTE AND))
       (COND ((EQ (CADR (CADAR L)) X)
              (OR NIL (TESTF X (CDDAR L))))
             (T T)))
      (T (COND ((EQ (CADAR L) X) NIL)
               (T T))))))
(TESTS (LAMBDA (X L)
(COND ((EQ (CAAR L)(QUOTE AND))
       (COND ((EQ (CADR (CADAR L)) X) NIL)
             (T (AND T (TESTS X (CDDAR L))))))
      (T (COND ((EQ (CADAR L) X) NIL)
               (T T))))))
```

```
      ))
DEFINE((
(REORDER (LAMBDA (E L Q)
(COND ((ATOM (CAR Q))(CAR Q))
      ((EQ (CAAR Q)(QUOTE AND))
       (COND ((EQ (CAR (CADAR Q))(QUOTE LT))
              (COND ((EQ (CADR (CADAR Q))(CAR L))
                         (REORDER E L (CDDAR Q)))
                    (T (PROG2 (PROG2 (SETQ E (CADAR Q))
                                     (RPLACA Q (CAR (CDDAR Q))))
                              (LIST (QUOTE AND) E (REORDER E L Q))))))
             (T (REORDER E L (CDDAR Q)))))
      (T (COND ((EQ (CAAR Q)(QUOTE LT))
                (COND ((EQ (CADAR Q)(CAR L))(CADR L))
                      (T (COND ((EQ (CAADR L)(QUOTE AND))
             (PROG2 (PROG2 (SETQ E (SEARCH (CDR L)))
                           (RPLACA E (CADAR E)))
                    (LIST (QUOTE AND)(CAR Q)(CADR L))))
                             (T (CAR Q))))))
               (T (CADR L)))))))
      ))
DEFINE((
(SUBS3 (LAMBDA (Q)
   (PROG (X E S)
      (SETQ X (CADR Q))
      (SETQ E (CADDR Q))
   A  (COND ((EQ (CAR E)(QUOTE LT))(GO B))
            ((EQ (CAR (CDADR E)) X)(GO B))
            (T (SETQ E (CADDR E))))
      (GO A)
   B  (SETQ Q (*SUBS3 X (CADDR Q) E E))
      (COND ((NULL (CAR (CADDR Q))) (RETURN (CADR Q))))
      (SETQ S (SEARCH (LIST Q)))
      (RPLACA S (CADAR S))
      (RETURN Q))))
      ))
DEFINE((
(*SUBS3 (LAMBDA (X L E H)
(COND ((EQ (CAR L)(QUOTE LT)) (LIST NIL))
      (T (COND ((EQ (CAR (CDADR L)) X) (LIST NIL))
               (T (COND ((EQ (CAR E)(QUOTE LT))(LIST (QUOTE AND)
(LIST (QUOTE LT)(CAR (CDADR L))(CADDR E))(*SUBS3 X (CADDR L) H H)))
                        (T (LIST (QUOTE AND)(LIST (QUOTE LT)
(CAR (CDADR L)) (CADR (CDADR E)))(*SUBS3 X L (CADDR E) H)))))))))))
      ))
```

APENDIX B

```
DEFINE((
(PRENEX (LAMBDA (L)
(PROG (P Q)
   (SETQ Q (CONS (QUOTE (X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13
                         X14 X15 X16 X17 X18 X19 X20)) NIL))
   (SETQ P (CONS P NIL))
   (SETQ L (PRENEXU L))
   (SETQ L (SUBLIS (CAR P) L))
   (RETURN L))))
      ))
DEFINE((
(PRENEXU (LAMBDA (L)
(COND ((ATOM L) L)
      ((EQ (CAR L)(QUOTE IMP))
         (COND ((EQ (CAADR L)(QUOTE A))
(PRENEXU (*PRENEX (LIST (QUOTE IMP)(LIST (QUOTE A)(CAR (CDADR L))
      (PRENEXU (CADDR (CADR L))))(PRENEXU (CADDR L))))))
                ((EQ (CAADR L)(QUOTE E))
(PRENEXU (*PRENEX (LIST (QUOTE IMP)(LIST (QUOTE E)(CAR (CDADR L))
      (PRENEXU (CADDR (CADR L))))(PRENEXU (CADDR L))))))
                (T (COND ((EQ (CAR (CADDR L))(QUOTE A))
(PRENEXU (*PRENEX (LIST (QUOTE IMP)(PRENEXU (CADR L))(LIST (QUOTE A)
      (CADR (CADDR L))(PRENEXU (CADDR (CADDR L))))))))
                         ((EQ (CAR (CADDR L))(QUOTE E))
(PRENEXU (*PRENEX (LIST (QUOTE IMP)(PRENEXU (CADR L))(LIST (QUOTE E)
      (CADR (CADDR L))(PRENEXU (CADDR (CADDR L))))))))
                         (T (*PRENEX (LIST (QUOTE IMP)
(PRENEXU (CADR L))(PRENEXU (CADDR L)))))))))
      ((EQ (CAR L)(QUOTE NOT))
         (COND ((EQ (CAADR L)(QUOTE A))
(PRENEXU (*PRENEX (LIST (QUOTE NOT)(LIST (QUOTE A)(CAR (CDADR L))
       (PRENEXU (CADDR (CADR L))))))))
                ((EQ (CAADR L)(QUOTE E))
(PRENEXU (*PRENEX (LIST (QUOTE NOT)(LIST (QUOTE E)(CAR (CDADR L))
      (PRENEXU (CADDR (CADR L))))))))
                (T (*PRENEX (LIST (QUOTE NOT)(PRENEXU (CADR L)))))))
      (T (LIST (CAR L)(PRENEXU (CADR L))(PRENEXU (CADDR L)))))))


      ))
DEFINE((
(*PRENEX (LAMBDA (L)
(COND ((ATOM L) L)
      ((EQ (CAR L)(QUOTE IMP))
         (COND ((EQ (CAADR L)(QUOTE A))
(LIST (QUOTE E)(CAAR Q)(LIST (QUOTE IMP)
      (PREN (CDADR L))(CADDR L))))
                ((EQ (CAADR L)(QUOTE E))
(LIST (QUOTE A)(CAAR Q)(LIST (QUOTE IMP)
      (PREN (CDADR L))(CADDR L))))
                (T (COND ((EQ (CAR (CADDR L))(QUOTE A))
(LIST (QUOTE A)(CAAR Q)(LIST (QUOTE IMP)
      (CADR L)(PREN (CDR (CADDR L))))))
```

```
                              ((EQ (CAR (CADDR L))(QUOTE E))
(LIST (QUOTE E)(CAAR Q)(LIST (QUOTE IMP)
      (CADR L)(PREN (CDR (CADDR L))))))
                              (T L)))))
      ((EQ (CAR L)(QUOTE NOT))
         (COND ((EQ (CAADR L)(QUOTE A))
(LIST (QUOTE E)(CAR (CDADR L))(LIST (QUOTE NOT)
      (CADDR (CADR L)))))
                ((EQ (CAADR L)(QUOTE E))
(LIST (QUOTE A)(CAR (CDADR L))(LIST (QUOTE NOT)
      (CADDR (CADR L)))))
                (T L)))
      (T L))))
      ))
DEFINE((
(PREN (LAMBDA (E)
(PROG (X S)
      (SETQ X (CAR E))
      (SETQ S (CAAR Q))
      (RPLACA Q (CDAR Q))
      (SETQ E (SUBST S X (CADR E)))
      (COND ((MEMBERG X (CAR P))
                    (RPLACA P (SUBST S X (CAR P))))
            (T (RPLACA P (PAIRLIS (LIST S)(LIST X)(CAR P)))))
      (RETURN E))))
      ))
```

APENDIX C

```
DEFINE((
(PRIN3 (LAMBDA (X TOPS)
   (PROG (J)
      (COND ((ATOM X)(GO A)))
      (SETQ J X)
      (COND ((EQ (CAR J)(QUOTE NOT))(GO B))
            ((OR (EQ (CAR J)(QUOTE A))
                 (EQ (CAR J)(QUOTE E)))(GO C))
            ((OR (EQ (CAR J)(QUOTE IMP))
                 (EQ (CAR J)(QUOTE IFF)))(GO E))
            (T (GO D)))
   A  (PRIN1 X)
      (RETURN X)
   B  (PRIN1 NOT)
      (SETQ J (CADR J))
      (PRIN3 J TOPS)
      (RETURN X)
   C  (PRIN1 LPAR)
      (PRIN1 (CAR J))
      (PRIN1 BLANK)
      (PRIN1 (CADR J))
      (PRIN1 RPAR)
      (SETQ J (CAR (CDDR J)))
      (PRIN3 J TOPS)
      (RETURN X)
   D  (SETQ TOPS (STACK TOPS (CAR J)))
      (COND ((EQ (CAR J)(CADR TOPS))(GO F)))
   E  (PRIN1 LPAR)
   F  (PRIN3 (CADR J) TOPS)
      (PRIN1 BLANK)
      (PRIN1 (CADR (ASSOC (CAR J) LS)))
      (PRIN1 BLANK)
      (PRIN3 (CAR (CDDR J)) TOPS)
      (COND ((OR (EQ (CAR J)(QUOTE IMP))
                 (EQ (CAR J)(QUOTE IFF)))(GO G)))
      (SETQ TOPS (CDR TOPS))
      (COND ((EQ (CAR J)(CAR TOPS))(GO H)))
   G  (PRIN1 RPAR)
   H  (RETURN X))))
      ))
DEFINE((
(PRINTER (LAMBDA (X)
   (PROG (TOPS NOT LPAR RPAR BLANK LS)
      (SETQ NOT (QUOTE $$'¬'))
      (SETQ LPAR (QUOTE $$'('))
      (SETQ RPAR (QUOTE $$')'))
      (SETQ BLANK (QUOTE $$' '))
      (SETQ LS (QUOTE ((EQU $$'=') (LT $$'<') (OR $$'|') (AND $$'&')
(IMP $$'=>') (IFF $$'<=>'))))
      (SETQ TOPS (LIST NIL))
      (TERPRI)
      (TERPRI)
      (PRIN3 X TOPS)
```

```
(PRIN1 (QUOTE $$'          '))
(RETURN X))))
))
```

APENDIX D

```
DEFINE((
(TEOREMA (LAMBDA (L)
(PROG (U E J TOP R B)
      (TERPRI)
      (PRINTER L)
      (TERPRI)
      (SETQ U L)
      (SETQ L (PRENEX L))
      (COND ((EQUAL U L)(GO A)))
      (PRINTER L)
      (PRINT (QUOTE (PRENEX)))
A     (SETQ U L)
      (SETQ L (SUBS0 L))
      (COND ((EQUAL U L)(GO STO)))
      (PRINTER L)
      (PRINT (QUOTE $$'(SUBS0)'))
      (SETQ U L)
      (SETQ L (SUBS00 L))
      (COND ((EQUAL U L)(GO STO)))
      (PRINTER L)
      (PRINT (QUOTE $$'(SUBS00)'))
STO   (SETQ L (LIST L))
      (SETQ J L)
      (SETQ TOP (WALK TOP L))
RT    (SETQ L (CAR TOP))
      (SETQ TOP (CDR TOP))
      (SETQ U (CAR L))
      (RPLACA L (SUBS01 (CAR L)))
      (COND ((EQUAL U (CAR L))(GO B)))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(SUBS01)'))
      (SETQ U (CAR L))
B     (RPLACA L (SUBS1 (CAR L)))
      (COND ((EQUAL U (CAR L))(GO BD)))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(SUBS1)'))
      (COND ((EQ (CAR J) T)(RETURN T))
            ((EQUAL (CAR J)(LIST (QUOTE NOT) T))
                  (RETURN (QUOTE $$'NIL'))))
BD    (COND ((OR (EQ (CAR L) T)
                 (EQUAL (CAR L)(LIST (QUOTE NOT) T)))(GO RT)))
D     (SETQ U (CAR L))
      (RPLACA L (DNF (CAR L)))
      (COND ((EQUAL U (CAR L))(GO BB)))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(DNF)'))
BB    (COND ((NULL (CDDAR L))(SETQ L (CDAR L))))
      (SETQ U (CAR L))
      (RPLACA L (SUBS2 (CAR L)))
      (COND ((EQUAL U (CAR L))(GO E)))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(SUBS2)'))
      (GO D)
```

```
E     (COND ((NOT (EQ (CAAR (CDDAR L))(QUOTE OR)))(GO H)))
      (RPLACA L (SUBS03 (CADAR L)(CAR (CDDAR L))))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(SUBS03)'))
H     (COND ((NOT (EQ (CAAR L)(QUOTE OR)))(GO I)))
      (SETQ L (CDAR L))
I     (COND ((MEMBERG (CADAR L)(CAR (CDDAR L)))(GO AA)))
      (RPLACA L (CAR (CDDAR L)))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(A)'))
K     (SETQ L (CDR L))
      (COND ((NULL L)(GO RT))
            (T (GO H)))
AA    (SETQ U (CAR L))
      (RPLACA L (SUBS003 U (CAR L)(CDDAR L)))
      (COND ((EQ U (CAR L))(GO RS)))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(SUBS003)'))
RS    (SETQ R NIL)
      (SETQ U L)
      (SETQ B L)
      (COND ((EQ (CAAR L)(QUOTE E))(GO L))
            (T (SETQ R T)))
      (SETQ B (SEARCH L))
      (SETQ U (CDDAR B))
L     (COND ((EQ (CAAR (CDDAR U))(QUOTE AND))(GO M)))
N     (COND ((NULL R)(GO CD)))
      (RPLACA B (CADAR B))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(D)'))
      (GO K)
CD    (SETQ U (CAR TOP))
      (COND ((EQ (CAAR U)(QUOTE NOT))(SETQ U (CDDR (CADAR U))))
            (T (SETQ U (CDDAR U))))
      (COND ((EQ (CAAR U)(QUOTE NOT))(SETQ U (CDAR U))))
      (RPLACA U T)
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(C)'))
      (GO RT)
M     (SETQ B (CDDAR U))
AL    (COND ((NOT (EQ (CAAR B)(QUOTE AND)))(GO EQN)))
      (SETQ B (CDAR B))
EQN   (COND ((EQ (CAAR B)(QUOTE EQU))
             (COND ((EQ (CADAR B)(CAR (CDDAR B)))(GO CONT))
                   (T (GO SUR)))))
CONT  (SETQ B (CDR B))
      (COND ((NULL B)(GO W))
            (T (GO AL)))
SUR   (COND ((EQ (CADAR B)(CADAR U))(SETQ B (CAR (CDDAR B))))
            (T (SETQ B (CADAR B))))
      (RPLACA (CDDAR U)(SUBST B (CADAR U)(CAR (CDDAR U))))
      (PRINTER (CAR J))
      (PRINT (QUOTE $$'(E)'))
      (RPLACA U (CAR (CDDAR U)))
```

```
        (PRINTER (CAR J))
        (PRINT (QUOTE $$'(F)'))
        (GO K)
W       (COND ((NULL (TEST2 (CADAR U)(CDDAR U)))(GO Y)))
        (SETQ B (SEARCH L))
        (COND ((NULL R)(GO CD)))
        (RPLACA B (CADAR B))
        (PRINTER (CAR J))
        (PRINT (QUOTE $$'(H)'))
        (GO K)
Y       (SETQ B (LIST (CADAR U)(CAR (CDDAR U))))
        (SETQ R (CAR (CDDAR U)))
        (RPLACA (CDDAR U)(REORDER E B (CDR B)))
        (COND ((EQUAL R (CAR (CDDAR U)))(GO Z)))
        (PRINTER (CAR J))
        (PRINT (QUOTE $$'(I)'))
Z       (RPLACA U (SUBS3 (CAR U)))
        (PRINTER (CAR J))
        (PRINT (QUOTE $$'(SUBS3)'))
        (GO K))))
        ))
```