```
(FILECREATED "16-Jan-80 19:21:22" <AFFIRM>XEVAL.;6 70382

     changes to:  DO\NARY\OP

     previous date: " 1-Sep-79 19:37:30" <AFFIRM>XEVAL.;5)


(PRETTYCOMPRINT XEVALCOMS)

(RPAQQ XEVALCOMS ((FNS * XEVALFNS)
        (BLOCKS (XEVALBLOCK SIMP\ASSIGN SIMP\R\ACCESS ALPHA AND\TO\ARG\LIST APPLY\STATE APPLY\F\OF\X
                            ARG\LIST\TO\AND ARG\LIST\TO\OP BIND\ARGS CONSTANT\TERM CONSTANT\FACTORS
                            CONSTANT\MULT\FACTORS DEFINE\FUNC DEFINED? DISTRIBUTE DOMAIN DO\NARY\OP DO\POWER
                            EQUAL\QUANTIFIED UNION#XEV EVENP FUNC\BODY FORMAL\PARMS IN\SET? SIMP\ALL
                            SIMP\A\SET SIMP\A\SUB SIMP\DIFF SIMP\DIV SIMP\EQ SIMP\EQUIVALENT SIMP\USER\FUNC
                            SIMP\GE SIMP\GT SIMP\IMP SIMP\LE SIMP\LT SIMP\MOD SIMP\NE SIMP\NEG SIMP\NEG\LIST
                            SIMP\NEG\MULT SIMP\NOT SIMP\NOT\LIST SIMP\POWER SIMP\SOME SIMP\SWAP N\ADD N\AND
                            NARY\X NEG\MAX NEG\MULT NEG\TERM N\MAX N\MIN N\MULT N\OR ORDERED\INSERT ORDERING
                            PRINT\PROVED REMOVE#XEV PRINT\REWRITE SET\DEFINITION SORT#XEV SUPER\PUT\IN
                            UNMATCHED\YS X\ADD\N X\ALPHA X\AND\N XEVAL XEVAL2 X\IF X\IMP X\MAX\N X\MIN\N
                            X\MULT\N X\ORDERED\MEMBER? X\OR\N XEVAL\TURN\OFF XEVAL\TURN\ON COMMONFACTORS
                            FACTORIZE FOUNDIN GETFACTORS GETMAXCONSTS GETNEGPWR GETNUMDEN ISINT MKINV MKPWRL
                            MKQUOT NUMERICAL REMAFACT REMAFACTFROMLIST REMFROMALL REMORMULT REMOVEFACTORS
                            (BLKAPPLYFNS SIMP\ASSIGN SIMP\ALL SIMP\A\SET SIMP\R\ACCESS SIMP\A\SUB SIMP\DIFF
                                         SIMP\DIV SIMP\EQ SIMP\EQUIVALENT SIMP\GE SIMP\GT SIMP\LE SIMP\LT
                                         SIMP\MOD SIMP\NE SIMP\NEG SIMP\NOT SIMP\POWER SIMP\SOME SIMP\SWAP
                                         N\ADD N\AND N\MAX N\MIN N\MULT N\OR X\ADD\N X\AND\N X\IF X\IMP
                                         X\MAX\N X\MIN\N X\MULT\N X\OR\N MKINV MKQUOT)
                            (GLOBALVARS CONTEXT RULETR STATE UFUNS UNDEFINED XEVALTRACESET)
                            (ENTRIES XEVAL XEVAL\TURN\OFF XEVAL\TURN\ON SET\DEFINITION SIMP\NEG NEG\TERM
                                     GETNUMDEN ISINT)
                            (NOLINKFNS . T)))
        (DECLARE: DOEVAL@COMPILE (PROP MACRO NULL\SET? SIMP\PAIR SIMP\PREF CONSTANT? MKLISTPREF))
        (DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA SIMP\PREF))

(RPAQQ XEVALFNS (ALPHA AND\TO\ARG\LIST APPLY\F\OF\X APPLY\STATE ARG\LIST\TO\AND ARG\LIST\TO\OP BIND\ARGS
                       COMMONFACTORS CONSTANT? CONSTANT\FACTORS CONSTANT\MULT\FACTORS CONSTANT\TERM DEFINED?
                       DEFINE\FUNC DISTRIBUTE DOMAIN DO\NARY\OP DO\POWER EQUAL\QUANTIFIED EVENP FACTORIZE
                       FORMAL\PARMS FOUNDIN FUNC\BODY GETFACTORS GETMAXCONSTS GETNEGPWR GETNUMDEN IN\SET?
                       ISINT MKINV MKLISTPREF MKPWRL MKQUOT NARY\X NEG\MAX NEG\MULT NEG\TERM NULL\SET?
                       NUMERICAL N\ADD N\AND N\MAX N\MIN N\MULT N\OR ORDERED\INSERT ORDERING PRINT\PROVED
                       PRINT\REWRITE REMAFACT REMAFACTFROMLIST REMFROMALL REMORMULT REMOVE#XEV REMOVEFACTORS
                       SET\DEFINITION SIMP\ALL SIMP\ASSIGN SIMP\A\SET SIMP\A\SUB SIMP\DIFF SIMP\DIV SIMP\EQ
                       SIMP\EQUIVALENT SIMP\GE SIMP\GT SIMP\IMP SIMP\LE SIMP\LT SIMP\MOD SIMP\NE SIMP\NEG
                       SIMP\NEG\LIST SIMP\NEG\MULT SIMP\NOT SIMP\NOT\LIST SIMP\PAIR SIMP\POWER SIMP\PREF
                       SIMP\R\ACCESS SIMP\SOME SIMP\SWAP SIMP\USER\FUNC SORT#XEV SUPER\PUT\IN UNION#XEV
                       UNMATCHED\YS XEVAL XEVAL2 XEVAL\TURN\OFF XEVAL\TURN\ON X\ADD\N X\ALPHA X\AND\N X\IF
                       X\IMP X\MAX\N X\MIN\N X\MULT\N X\ORDERED\MEMBER? X\OR\N))

(DEFINEQ

                                                                                    1

(ALPHA
  (LAMBDA (F X Y)                                   (* If F is defined at X, then sets F
                                                    (X) to Y. Returns the new F)

    (if F=NIL
        then NIL
      elseif (EQUAL F:1:MEM1 X)
        then <(SIMP\PAIR X Y) ! F::1>
      elseif (ORDERING X F:1:MEM1)
        then F
      else <F:1 !(ALPHA F::1 X Y)
           >))

                                                                                    2

(AND\TO\ARG\LIST
  (LAMBDA (PREFIX)                                  (* Change the expression to a list of
```

```
    (if (NLISTP PREFIX)
        then <PREFIX>
      elseif PREFIX:OPR=ANDOP
        then PREFIX:ARGS
      else <PREFIX>))
```

arguments)

3

```
(APPLY\F\OF\X
  (LAMBDA (F X)
    (if (NULL\SET? F)
        then UNDEFINED
      elseif (EQUAL F:1:MEM1 X)
        then F:1:MEM2
      elseif (ORDERING X F:1:MEM1)
        then UNDEFINED
      else (APPLY\F\OF\X F::1 X))
```

(* Returns F (X))

4

```
(APPLY\STATE
  (LAMBDA (S X)
    (if (DEFINED? S X)
        then (APPLY\F\OF\X S X)
      else X))
```

(* apply state S to element X)

5

```
(ARG\LIST\TO\AND
  (LAMBDA (AL)

    (if AL=NIL
        then TRUE
      elseif AL::1=NIL
        then AL:1
      else (MKLISTPREF ANDOP AL))
```

(* Convert the arg list AL to a prefix expr with the AND operator)

6

```
(ARG\LIST\TO\OP
  (LAMBDA (AL OPA TOSSVALUE)

    (if AL=NIL
        then TOSSVALUE
      elseif AL::1=NIL
        then AL:1
      else (MKLISTPREF OPA AL))
```

(* Convert the arg list AL to a prefix expr with any n-ary op OPA)

7

```
(BIND\ARGS
  (LAMBDA (STATE2 FARGS ACTARGS)

    (if FARGS=NIL
        then STATE2
      else (X\ALPHA (BIND\ARGS STATE2 FARGS::1 ACTARGS::1)
                    FARGS:1 ACTARGS:1)))
```

(* Returns a new state with actual args bound to formal args)

8

```
(COMMONFACTORS
  (LAMBDA (LX LY)

    (if LX=NIL or LY=NIL
        then NIL
```

(* Makes a list of factors common to factor list LX and LY)

```
    else (PROG (GONE FACTORS)
               (GONE←(REMAFACTFROMLIST LX LY:1))
               (if GONE
                   then (RETURN <LY:1 !(COMMONFACTORS GONE LY::1)
                                >))
               (if (NLISTP LY:1) or ~(EQUAL LY:1:OPR PIROP)
                   then (RETURN (COMMONFACTORS LX LY::1)))
               (FACTORS←NIL)
               (GONE←T)
               [while GONE do (PROGN GONE←(REMAFACTFROMLIST LX LY:1:ARG1)
                                     (if GONE
                                         then (PROGN LX←GONE
                                                     FACTORS← <LY:1:ARG1 ! FACTORS>]
               (RETURN < ! FACTORS !(COMMONFACTORS LX LY::1)
                       >])
```

9

(CONSTANT?

```
  [LAMBDA (A)
    (if (NUMBERP A) or A=NEGINF or A=POSINF or A=TRUE or A=FALSE])
```

(* Returns T if A is a constant)

10

(CONSTANT\FACTORS

```
  [LAMBDA (X)

    (if (NUMBERP X)
        then <X 1>
      elseif (NLISTP X)
        then <1 X>
      elseif X:OPR=NEGOP
        then <(- 1)
              X:ARG1>
      elseif X:OPR=MULTOP
        then (CONSTANT\MULT\FACTORS 1 NIL X:ARGS)
      elseif X:OPR=INVOP and (NUMBERP X:ARG1)
        then <X 1>
      else <1 X>])
```

(* Returns a list WITH the NUMERICAL factor of X first, remaining factors second)

11

(CONSTANT\MULT\FACTORS

```
  [LAMBDA (K HLX TLX)

    (if TLX=NIL
        then (if HLX=NIL
                 then <K 1>
               elseif HLX::1=NIL
                 then <K HLX:1>
               else <K (MKLISTPREF MULTOP (REVERSE HLX))
                    >)
      else (PROG (X)
                 (X←TLX:1)
                 (RETURN (if (NUMERICAL X)
                             then (CONSTANT\MULT\FACTORS (DO\NARY\OP MULTOP K X)
                                                         HLX TLX::1)
                           elseif (NEG\TERM X)
                             then (CONSTANT\MULT\FACTORS (SIMP\NEG K)
                                                         <(SIMP\NEG X) ! HLX> TLX::1)
                           else (CONSTANT\MULT\FACTORS K <X ! HLX> TLX::1])
```

(* Multiply the const from each factor and place in front of expr)

12

(CONSTANT\TERM

```
  [LAMBDA (LX)

    (if LX=NIL
```

(* Finds a numeric term in sorted list LX if one exists, else NIL)

```
    then NIL
  elseif (NUMERICAL LX:1)
    then LX:1
  else (CONSTANT\TERM LX::1))
```

13

```
(DEFINED?
  (LAMBDA (F X)                                    (* Return T if function F is defined? at X)
    (if F=NIL
        then NIL
      elseif (EQUAL X F:1:MEM1)
        then T
      elseif (ORDERING X F:1:MEM1)
        then NIL
      else (DEFINED? F::1 X))
```

14

```
(DEFINE\FUNC
  (LAMBDA (F PARS BODY)                            (* DEFINE\FUNC defines the function F by
                                                   putting it into UFUNS)
    UFUNS←(X\ALPHA UFUNS F (SIMP\PAIR PARS BODY))
```

15

```
(DISTRIBUTE
  (LAMBDA (OPCODE X TERMS)

        (* For each term in TERMS, apply the OPCODE to X and the term. Only works if OPCODE is defined n-ary.)

    (if TERMS
        then <(DO\NARY\OP OPCODE X TERMS:1) !(DISTRIBUTE OPCODE X TERMS::1)
             >))
```

16

```
(DOMAIN
  (LAMBDA (F)

        (* Return the domain (an ordered set) of the function F. Since the function is ordered on first members, they
        can be picked up in sequence to get the ordered set that is the domain.)

    (if F
        then <F:1:MEM1 !(DOMAIN F::1)
             >))
```

17

```
(DO\NARY\OP
  (LAMBDA (OPCODE X Y)                             (* R. Bates "14-Dec-79 14:43")

        (* Applies OPCODE to the already XEVAL ed args X and Y. Use this proc instead of N\ADD, N\AND, N\MAX, N\MIN,
        N\MULT, or N\OR.)

    (if (NLISTP X) or ~(EQUAL X:OPR OPCODE)
        then (if (NLISTP Y) or ~(EQUAL Y:OPR OPCODE)
                 then (PROG (NEGFCN Z (MKFCN (GETP OPCODE 'NARYS)))
                            (if MKFCN=NIL
                                then (PRINTLINES OPCODE "not defined as N-ary op *****"))
                            (NEGFCN←MKFCN:3)
                            (MKFCN←MKFCN:1)
                            (Z←(BLKAPPLY MKFCN <X Y (BLKAPPLY NEGFCN <X>)
                                                    >))
```

```
                    (if (NLISTP Z) or ~(EQUAL Z:OPR OPCODE) or (ORDERING Z:ARG1 Z:ARG2)
                        then (RETURN Z)
                      else (RETURN (SIMP\PREF Z:OPR Z:ARG2 Z:ARG1)
              else (SUPER\PUT\IN <X> Y:ARGS OPCODE))
      elseif (NLISTP Y) or ~(EQUAL Y:OPR OPCODE)
        then (SUPER\PUT\IN <Y> X:ARGS OPCODE)
      elseif (FLENGTH X) gt (FLENGTH Y)
        then (SUPER\PUT\IN Y:ARGS X:ARGS OPCODE)
      else (SUPER\PUT\IN X:ARGS Y:ARGS OPCODE))
```

18

```
(DO\POWER
  (LAMBDA (X Y)                                   (* Multiply X by itself Y-1 times.
                                                     NUMBERP Y must be T and Y must be
                                                     non-negative.)
    (if (NUMBERP X)
        then (if X=0 and Y=0
                  then (SIMP\PREF PWROP X Y)
                elseif Y=0
                  then 1
                else (for J from 1 to Y by 1 bind Z←1 do Z←Z*X finally (RETURN Z)))
      elseif (ILEQ Y 0)
        then (SIMP\PREF PWROP X Y)
      elseif Y=1
        then X
      else (DO\NARY\OP MULTOP X (DO\POWER X Y-1))
```

19

```
(EQUAL\QUANTIFIED
  (LAMBDA (X Y)                                   (* Are X, Y equivalent expressions except for
                                                     the quantified variables?)
    (if (NLISTP X) or (NLISTP Y)
        then (EQUAL X Y)
      elseif (EQUAL X:OPR Y:OPR)
        then (if X:OPR=ALLOP or X:OPR=SOMEOP
                  then (EQUAL\QUANTIFIED X:ARG2 (SUBST X:ARG1 Y:ARG1 Y:ARG2))
                else (EQUAL\QUANTIFIED X:1 Y:1))
```

20

```
(EVENP
  (LAMBDA (X)                                     (* Is X (must be NUMBERP) even)
    (IREMAINDER X 2)=0))
```

21

```
(FACTORIZE
  (LAMBDA (X)
    (PROG (DIVISOR FACTORS)                       (* Return a list of ALL FACTORS of the number
                                                     X)
          (X←(ABS X))
          (if X lt 2
              then (RETURN NIL))
          (FACTORS←NIL)
          (while (REMAINDER X 2)=0 do (PROGN X←X/2
                                             FACTORS← <2 ! FACTORS>))
          (DIVISOR←3)
          (while (ILEQ DIVISOR*DIVISOR X) do (if (REMAINDER X DIVISOR)=0
                                                 then (PROGN X←X/DIVISOR
                                                             FACTORS← <DIVISOR ! FACTORS>)
                                               else DIVISOR←DIVISOR+2))
          (RETURN (if X=1
                      then FACTORS
                    else <X ! FACTORS>))
```

22

```
(FORMAL\PARMS
  (LAMBDA (F)                                   (* Returns the formal param list of func F,
                                                   with all param descriptors)
    (if (DEFINED? UFUNS F)
        then (APPLY\F\OF\X UFUNS F):MEM1))
```

23

```
(FOUNDIN
  (LAMBDA (X EXP)                               (* Makes list of occurrences in EXP of atom X
                                                   used as an OPERATOR)
    (if (NLISTP EXP)
        then NIL
      else (PROG (FOUND)
                 (if EXP:OPR=X
                     then FOUND← <EXP>
                   else FOUND←NIL)
                 (EXP←EXP:ARGS)
                 (while EXP do (PROGN FOUND← < !(FOUNDIN X EXP:1) ! FOUND> EXP←EXP::1))
                 (RETURN FOUND)))
```

24

```
(FUNC\BODY
  (LAMBDA (F)                                   (* Returns the body of the function F)
    (if (DEFINED? UFUNS F)
        then (APPLY\F\OF\X UFUNS F):MEM2))
```

25

```
(GETFACTORS
  (LAMBDA (X)                                   (* Gets a list of factors
                                                   (except sums) in X)
    (if (NLISTP X)
        then (if (NUMBERP X)
                 then (FACTORIZE X)
               else <X>)
      elseif X:OPR=MULTOP
        then (PROG (FACTORS)
                   (FACTORS←NIL)
                   (X←X:ARGS)
                   (while X do (PROGN FACTORS← < !(GETFACTORS X:1) ! FACTORS> X←X::1))
                   (RETURN FACTORS))
      elseif X:OPR=PWROP
        then (PROG (FACTORS PWRFACTORS)
                   (FACTORS←(GETFACTORS X:ARG1))
                   (PWRFACTORS←NIL)
                   (while FACTORS do (PROGN PWRFACTORS← <(SIMP\PREF PWROP FACTORS:1 X:ARG2) ! PWRFACTORS>
                                            FACTORS←FACTORS::1))
                   (RETURN PWRFACTORS))
      elseif X:OPR=ADDOP
        then (PROG (FACTORS)
                   (X←X:ARGS)
                   (FACTORS←(GETFACTORS X:1))
                   (X←X::1)
                   (while X do (PROGN FACTORS←(COMMONFACTORS FACTORS (GETFACTORS X:1))
                                      X←X::1))
                   (RETURN FACTORS))
      else <X>))
```

26

```
(GETMAXCONSTS
  (LAMBDA (LX)
    (PROG (FACTORS NFACTORS)                    (* Get NUMBERP constants in ALL ARGS of a MAX
                                                   or MIN ARG list LX)
```

```
        (FACTORS←(GETFACTORS (MKLISTPREF ADDOP LX)))
        (NFACTORS←NIL)
        (while FACTORS do (PROGN (if (NUMBERP FACTORS:1)
                                     then NFACTORS← <(ABS FACTORS:1) ! NFACTORS>)
                                 FACTORS←FACTORS::1))
        (RETURN NFACTORS))
```

27

```
(GETNEGPWR
  (LAMBDA (X)
    (PROG (FACTORS NPFACTORS CHFACTORS)                    (* Separates negl\term powers
                                                              (WITH sign of power CHANGED) from other
                                                              FACTORS)
          (FACTORS←(GETFACTORS X))
          (NPFACTORS←NIL)
          (CHFACTORS←NIL)
          (while FACTORS do (PROGN (if (LISTP FACTORS:1) and FACTORS:1:OPR=PWROP and (NEG\TERM FACTORS:1:ARG2)
                                       then (PROGN NPFACTORS← <FACTORS:1 ! NPFACTORS> CHFACTORS←
                                                   <(SIMP\PREF PWROP FACTORS:1:ARG1 (SIMP\NEG FACTORS:1:ARG2))
                                                     ! CHFACTORS>))
                                   FACTORS←FACTORS::1))
          (if NPFACTORS=NIL
              then (RETURN <1 X>)
            else (RETURN <(ARG\LIST\TO\OP (SORT*XEV CHFACTORS)
                                          MULTOP 1)
                         (REMOVEFACTORS X NPFACTORS)
                         >))
          (RETURN NIL))
```

28

```
(GETNUMDEN
  (LAMBDA (X)
    (PROG (FACTORS DENFACTORS TERM CHFACTORS)              (* Returns list (numerator x,denominator X)
                                                              after making common denominator if necessary)
          (if (NLISTP X)
              then (RETURN <X 1>))
          (if X:OPR=ADDOP
              then FACTORS←(GETFACTORS (MKLISTPREF MULTOP X:ARGS))
            else FACTORS←(GETFACTORS X))
          (DENFACTORS←NIL)
          (CHFACTORS←NIL)
          (while FACTORS do (PROGN TERM←FACTORS:1
                                   (if (LISTP TERM) and TERM:OPR=INVOP and ~(MEMBER TERM DENFACTORS)
                                       then (PROGN DENFACTORS← <TERM ! DENFACTORS> CHFACTORS←
                                                   <TERM:ARG1 ! CHFACTORS>))
                                   FACTORS←FACTORS::1))
          (if DENFACTORS=NIL
              then (RETURN <X 1>))
          (CHFACTORS←(ARG\LIST\TO\OP (SORT*XEV CHFACTORS)
                                     MULTOP 1))
          (RETURN <(REMORMULT X DENFACTORS)
                    *       CHFACTORS>))
```

29

```
(IN\SET?
  (LAMBDA (X Y)                                            (* Returns logical value of
                                                              "X is an element of Y.")
    (X\ORDERED\MEMBER? X Y))
```

30

```
(ISINT
  (LAMBDA (X)                                              (* Returns T if expression X is known to be
                                                              integer, else NIL)
```

(* ALL variables ARE considered integer. Only inverse (which QUOTIENT simplifies to) and EXPT to a negative power return REAL (non-integer))

```
(if (FOUNDIN INVOP X)
    then NIL
  else (PROG (WHEREFOUND NOTFOUND)
             (WHEREFOUND←(FOUNDIN PHROP X))
             (NOTFOUND←T)
             (while NOTFOUND and WHEREFOUND do (PROGN NOTFOUND←(if (SIMP\LE 0 WHEREFOUND:1:ARG2)=TRUE
                                                                   then T
                                                                 else NIL)
                                                      WHEREFOUND←WHEREFOUND::1))
             (RETURN NOTFOUND))
```

31

```
(MKINV
  (LAMBDA (X)                                   (* Simplify the inverse of an arithmetic
                                                   expression X)
    (if X=0
      then (SIMP\PREF INVOP X)
     elseif X=POSINF or X=NEGINF
      then (SIMP\PREF INVOP X)
     elseif X=1 or X=(- 1)
      then X
     elseif (NLISTP X)
      then (SIMP\PREF INVOP X)
     else (PROG (XNUMDEN XNP)
                (XNUMDEN←(GETNUMDEN X))
                (if ~(EQUAL XNUMDEN:EL2 1)
                    then (RETURN (DO\NARY\OP MULTOP XNUMDEN:EL2 (MKINV XNUMDEN:EL1)
                (XNP←(GETNEGPWR X))
                (if ~(EQUAL XNP:EL1 1)
                    then (RETURN (DO\NARY\OP MULTOP XNP:EL1 (MKINV XNP:EL2)))
                  else (RETURN (SIMP\PREF INVOP X)))
                (RETURN NIL))
```

32

```
(MKLISTPREF
  (LAMBDA (XOP XARGLIST)                        (* Define MKLISTPREF (op,arglist) = op %.
                                                   ARGLIST)
    <XOP ! XARGLIST>))
```

33

```
(MKPWRL
  (LAMBDA (LX Y)                                (* Takes ALL items in list LX to power Y and
                                                   orders the results)
    (if LX=NIL
        then NIL
      else (ORDERED\INSERT (SIMP\POWER LX:1 Y)
                           (MKPWRL LX::1 Y))
```

34

```
(MKQUOT
  (LAMBDA (X Y)                                 (* Simplify the QUOTIENT
                                                   (REAL division) of X and Y)
    (DO\NARY\OP MULTOP X (MKINV Y))
```

35

```
(NARY\X
  (LAMBDA (LX OPNX)
```

(* Narry's the top level of LX and all items connected to it by the operator OPNX. Does not take as input or produce as output the top level operator.)

```
(for X in LX bind Y←NIL do (if (NLISTP X) or ~(EQUAL X:OPR OPNX)
                               then Y← <(XEVAL2 X) ! Y>
                              else Y← < !(NARY\X X:ARGS OPNX) ! Y>)
   finally (RETURN Y))
```

36

```
(NEG\MAX
  (LAMBDA (LX)
```

(* Determines if a MAX (or MIN or PLUS) is considered NEG\TERM. Count the NEG\TERMs among the arguments to determine if they outnumber non-NEG\TERMs. If tie, use ORDERING.)

```
    (for X in LX bind COUNT←0 do (if (NEG\TERM X)
                                     then COUNT←(COUNT-1)
                                    else COUNT←(COUNT+1))
       finally (RETURN (if (MINUSP COUNT)
                           then T
                          elseif COUNT gt 0
                           then NIL
                          elseif (ORDERING LX (SIMP\NEG\LIST LX))
                           then NIL
                          else T))
```

37

```
(NEG\MULT
  (LAMBDA (LX)                                    (* Returns T if any factor is NEG\TERM)
    (if LX=NIL
        then NIL
      elseif (NEG\TERM LX:1)
        then T
      else (NEG\MULT LX::1))
```

38

```
(NEG\TERM
  (LAMBDA (X)
```

(* Returns T if NEGINF, -NUMERIC, " - (Y) ", " ... * (NEG\TERM) * ...) ", " (... * (NEG\TERM) * ...) / B ", " NEG\TERM ↑ NUMERIC ", more args NEG\TERM of +, MAX, MIN.)

```
    (if X=NEGINF
        then T
      elseif (NUMBERP X)
        then (MINUSP X)
      elseif (NLISTP X)
        then NIL
      elseif X:OPR=NEGOP
        then T
      elseif X:OPR=ADDOP or X:OPR=MAXOP or X:OPR=MINOP
        then (NEG\MAX X:ARGS)
      elseif X:OPR=MULTOP
        then (NEG\MULT X:ARGS)
      elseif X:OPR=INVOP
        then (NEG\TERM X:ARG1)
      elseif X:OPR=DIVOP
        then (NEG\TERM X:ARG1)
      elseif X:OPR=PWROP and (NUMBERP X:ARG2)
        then (NEG\TERM X:ARG1)
      else NIL))
```

39

```
(NULL\SET?
  (LAMBDA (X)
    X=NIL))
```

40

```
(NUMERICAL
  (LAMBDA (X)                                   (* Returns T if X is a number or an
                                                   expression of only numbers)
    (if (NLISTP X)
        then (NUMBERP X)
      elseif (GETPROP X:OPR 'EVFUN)=NIL
        then NIL
      else (PROG (NUMONLY)
                 (NUMONLY←T)
                 (X←X:ARGS)
                 (while NUMONLY and X do (PROGN NUMONLY←(NUMERICAL X:1)
                                                X←X::1))
                 (RETURN NUMONLY))
```

41

```
(N\ADD
  (LAMBDA (X Y NEGX)                            (* Simplify the SUM of two arguments)
    (if (LISTP X) and X:OPR=ADDOP or (LISTP Y) and Y:OPR=ADDOP
        then (DO\NARY\OP ADDOP X Y)
      elseif (NUMBERP X) and (NUMBERP Y)
        then X+Y
      elseif Y=0
        then X
      elseif X=0
        then Y
      elseif X=POSINF and Y=NEGINF or X=NEGINF and Y=POSINF
        then (SIMP\PREF ADDOP X Y)
      elseif X=POSINF or X=NEGINF
        then X
      elseif Y=POSINF or Y=NEGINF
        then Y
      elseif (EQUAL X Y)
        then (DO\NARY\OP MULTOP 2 X)
      elseif (EQUAL NEGX Y)
        then 0
      elseif (LISTP Y) and Y:OPR=MAXOP
        then (MKLISTPREF MAXOP (SORT*XEV (DISTRIBUTE ADDOP X Y:ARGS)))
      elseif (LISTP Y) and Y:OPR=MINOP
        then (MKLISTPREF MINOP (SORT*XEV (DISTRIBUTE ADDOP X Y:ARGS)))
      elseif (LISTP X) and X:OPR=MAXOP
        then (MKLISTPREF MAXOP (SORT*XEV (DISTRIBUTE ADDOP Y X:ARGS)))
      elseif (LISTP X) and X:OPR=MINOP
        then (MKLISTPREF MINOP (SORT*XEV (DISTRIBUTE ADDOP Y X:ARGS)))
      elseif (LISTP X) and (X:OPR=MULTOP or X:OPR=INVOP) or (LISTP Y) and (Y:OPR=MULTOP or Y:OPR=INVOP)
        then (PROG (XFACTORS YFACTORS XNUMDEN YNUMDEN RNUM)
                   (XFACTORS←(CONSTANT\FACTORS X))
                   (YFACTORS←(CONSTANT\FACTORS Y))
                   (if (EQUAL XFACTORS:EL2 YFACTORS:EL2) and ~(EQUAL XFACTORS:EL2 1)
                       then (RETURN (DO\NARY\OP MULTOP (DO\NARY\OP ADDOP XFACTORS:EL1 YFACTORS:EL1)
                                                       XFACTORS:EL2)))
                   (XNUMDEN←(GETNUMDEN X))
                   (YNUMDEN←(GETNUMDEN Y))
                   (if XNUMDEN:EL2=1 and YNUMDEN:EL2=1
                       then (RETURN (SIMP\PREF ADDOP X Y)))
                   (RNUM←(DO\NARY\OP ADDOP (DO\NARY\OP MULTOP YNUMDEN:EL2 XNUMDEN:EL1)
                                           (DO\NARY\OP MULTOP XNUMDEN:EL2 YNUMDEN:EL1)))
                   (RETURN (if (NLISTP RNUM) or ~(EQUAL RNUM:OPR ADDOP)
                               then (MKQUOT RNUM (DO\NARY\OP MULTOP XNUMDEN:EL2 YNUMDEN:EL2))
                             else (SIMP\PREF ADDOP X Y)))
      else (SIMP\PREF ADDOP X Y))
```

```
(N\AND
  (LAMBDA (X Y NOTX)
    (if (LISTP X) and X:OPR=ANDOP or (LISTP Y) and Y:OPR=ANDOP
        then (DO\NARY\OP ANDOP X Y)
      elseif X=FALSE or Y=FALSE
        then FALSE
      elseif X=TRUE
        then Y
      elseif Y=TRUE
        then X
      elseif (EQUAL\QUANTIFIED X Y)
        then X
      elseif (EQUAL\QUANTIFIED NOTX Y)
        then FALSE
      elseif (LISTP X) and X:OPR=IMPOP and (EQUAL\QUANTIFIED X:ARG1 Y)
        then (DO\NARY\OP ANDOP Y X:ARG2)
      elseif (LISTP Y) and Y:OPR=IMPOP and (EQUAL\QUANTIFIED X Y:ARG1)
        then (DO\NARY\OP ANDOP X Y:ARG2)
      else (SIMP\PREF ANDOP X Y)))
```

(* Simplify logical and of two arguments)


```
(N\MAX
  (LAMBDA (X Y NOTX)

    (if (LISTP X) and X:OPR=MAXOP or (LISTP Y) and Y:OPR=MAXOP
        then (DO\NARY\OP MAXOP X Y)
      else (PROG ((R (SIMP\LE X Y)))
                 (if R=TRUE
                     then (RETURN Y)
                   elseif R=FALSE
                     then (RETURN X)
                   else (RETURN (SIMP\PREF MAXOP X Y))
```

(* Simplify the maximum of two expressions. NOTX is presently ignored in N\MAX, so any value will do.)


```
(N\MIN
  (LAMBDA (X Y NOTX)

    (if (LISTP X) and X:OPR=MINOP or (LISTP Y) and Y:OPR=MINOP
        then (DO\NARY\OP MINOP X Y)
      else (PROG ((R (SIMP\LE X Y)))
                 (if R=TRUE
                     then (RETURN X)
                   elseif R=FALSE
                     then (RETURN Y)
                   else (RETURN (SIMP\PREF MINOP X Y))
```

(* Simplify the minimum of two expressions. NOTX is presently ignored in N\MIN, so any value will do.)


```
(N\MULT
  (LAMBDA (X Y INVX)
    (if (LISTP X) and X:OPR=MULTOP or (LISTP Y) and Y:OPR=MULTOP
        then (DO\NARY\OP MULTOP X Y)
      elseif (NUMBERP X) and (NUMBERP Y)
        then X*Y
      elseif (EQUAL INVX Y)
        then 1
      elseif Y=1
        then X
      elseif X=1
        then Y
      elseif (NEG\TERM X) and (NEG\TERM Y)
        then (DO\NARY\OP MULTOP (SIMP\NEG X)
                                (SIMP\NEG Y))
```

(* Simplify PRODUCT of two arguments)

```
    elseif (NEG\TERM X)
      then (SIMP\NEG (DO\NARY\OP MULTOP (SIMP\NEG X)
                                 Y))
    elseif (NEG\TERM Y)
      then (SIMP\NEG (DO\NARY\OP MULTOP X (SIMP\NEG Y)))
    elseif X=POSINF and Y=POSINF
      then POSINF
    elseif X=POSINF or Y=POSINF or X=NEGINF or Y=NEGINF
      then (SIMP\PREF MULTOP X Y)
    elseif X=0 or Y=0
      then 0
    elseif (LISTP X) and X:OPR=INVOP and (LISTP Y) and Y:OPR=INVOP
      then (MKINV (DO\NARY\OP MULTOP X:ARG1 Y:ARG1))
    elseif ((NLISTP X) or X:OPR=ADDOP or X:OPR=PWROP) and (LISTP Y) and Y:OPR=INVOP
      then (PROG (FACTORS NEWX NEWY)
                 (FACTORS←(COMMONFACTORS (GETFACTORS X)
                                         (GETFACTORS Y:ARG1)))
                 (if FACTORS=NIL
                     then (RETURN (SIMP\PREF MULTOP X Y)))
                 (NEWX←(REMOVEFACTORS X FACTORS))
                 (NEWY←(MKINV (REMOVEFACTORS Y:ARG1 FACTORS)))
                 (RETURN (DO\NARY\OP MULTOP NEWX NEWY)))
    elseif ((NLISTP Y) or Y:OPR=ADDOP or Y:OPR=PWROP) and (LISTP X) and X:OPR=INVOP
      then (PROG (FACTORS NEWY NEWX)
                 (FACTORS←(COMMONFACTORS (GETFACTORS Y)
                                         (GETFACTORS X:ARG1)))
                 (if FACTORS=NIL
                     then (RETURN (SIMP\PREF MULTOP X Y)))
                 (NEWY←(REMOVEFACTORS Y FACTORS))
                 (NEWX←(MKINV (REMOVEFACTORS X:ARG1 FACTORS)))
                 (RETURN (DO\NARY\OP MULTOP NEWX NEWY)))
    elseif (LISTP X) and X:OPR=ADDOP
      then (MKLISTPREF ADDOP (SORT*XEV (DISTRIBUTE MULTOP Y X:ARGS)))
    elseif (LISTP Y) and Y:OPR=ADDOP
      then (MKLISTPREF ADDOP (SORT*XEV (DISTRIBUTE MULTOP X Y:ARGS)))
    elseif (LISTP X) and X:OPR=PWROP and (LISTP Y) and Y:OPR=PWROP and (EQUAL X:ARG1 Y:ARG1)
      then (SIMP\POWER X:ARG1 (DO\NARY\OP ADDOP X:ARG2 Y:ARG2))
    elseif (LISTP X) and X:OPR=PWROP and (EQUAL X:ARG1 Y)
      then (SIMP\POWER Y (DO\NARY\OP ADDOP X:ARG2 1))
    elseif (LISTP Y) and Y:OPR=PWROP and (EQUAL Y:ARG1 X)
      then (SIMP\POWER X (DO\NARY\OP ADDOP Y:ARG2 1))
    elseif (LISTP X) and (X:OPR=MAXOP or X:OPR=MINOP) and (LISTP Y) and Y:OPR=INVOP
      then (PROG (FACTORS)
                 (FACTORS←(GETMAXCONSTS X:ARGS))
                 (FACTORS←(COMMONFACTORS (GETFACTORS Y:ARG1)
                                         FACTORS))
                 (RETURN (MKLISTPREF X:OPR (REMFROMALL X:ARGS FACTORS)))
    elseif (LISTP Y) and (Y:OPR=MAXOP or Y:OPR=MINOP) and (LISTP X) and X:OPR=INVOP
      then (PROG (FACTORS)
                 (FACTORS←(GETMAXCONSTS Y:ARGS))
                 (FACTORS←(COMMONFACTORS (GETFACTORS X:ARG1)
                                         FACTORS))
                 (RETURN (MKLISTPREF Y:OPR (REMFROMALL Y:ARGS FACTORS)))
    elseif (EQUAL X Y)
      then (SIMP\POWER X 2)
    elseif (LISTP Y) and Y:OPR=MAXOP
      then (PROG (ZLEX)
                 (ZLEX←(SIMP\LE 0 X))
                 (if ZLEX=TRUE
                     then (RETURN (MKLISTPREF MAXOP (SORT*XEV (DISTRIBUTE MULTOP X Y:ARGS)))
                 (if ZLEX=FALSE
                     then (RETURN (MKLISTPREF MINOP (SORT*XEV (DISTRIBUTE MULTOP X Y:ARGS)))
                 (RETURN (SIMP\PREF MULTOP X Y)))
    elseif (LISTP Y) and Y:OPR=MINOP
      then (PROG (ZLEX)
                 (ZLEX←(SIMP\LE 0 X))
                 (if ZLEX=TRUE
                     then (RETURN (MKLISTPREF MINOP (SORT*XEV (DISTRIBUTE MULTOP X Y:ARGS)))
                 (if ZLEX=FALSE
                     then (RETURN (MKLISTPREF MAXOP (SORT*XEV (DISTRIBUTE MULTOP X Y:ARGS)))
                 (RETURN (SIMP\PREF MULTOP X Y)))
    elseif (LISTP X) and X:OPR=MAXOP
```

```
      then (PROG (ZLEY)
                 (ZLEY←(SIMP\LE 0 Y))
                 (if ZLEY=TRUE
                     then (RETURN (MKLISTPREF MAXOP (SORT*XEV (DISTRIBUTE MULTOP Y X:ARGS)
                 (if ZLEY=FALSE
                     then (RETURN (MKLISTPREF MINOP (SORT*XEV (DISTRIBUTE MULTOP Y X:ARGS)
                 (RETURN (SIMP\PREF MULTOP X Y)))
    elseif (LISTP X) and X:OPR=MINOP
      then (PROG (ZLEY)
                 (ZLEY←(SIMP\LE 0 Y))
                 (if ZLEY=TRUE
                     then (RETURN (MKLISTPREF MINOP (SORT*XEV (DISTRIBUTE MULTOP Y X:ARGS)
                 (if ZLEY=FALSE
                     then (RETURN (MKLISTPREF MAXOP (SORT*XEV (DISTRIBUTE MULTOP Y X:ARGS)
                 (RETURN (SIMP\PREF MULTOP X Y)))
    else (SIMP\PREF MULTOP X Y))
```

46

```
(N\OR
  (LAMBDA (X Y NOTX)                                   (* Simplify the logical or of two arguments)
    (if (LISTP X) and X:OPR=OROP or (LISTP Y) and Y:OPR=OROP
        then (DO\NARY\OP OROP X Y)
      elseif X=TRUE or Y=TRUE
        then TRUE
      elseif X=FALSE
        then Y
      elseif Y=FALSE
        then X
      elseif (EQUAL\QUANTIFIED X Y)
        then X
      elseif (EQUAL\QUANTIFIED NOTX Y)
        then TRUE
      else (SIMP\PREF OROP X Y))
```

47

```
(ORDERED\INSERT
  (LAMBDA (ELEM L)

          (* Inserts element ELEM into the ordered list L. Both L and the list returned are ordered on function
          ORDERING.)


    (if L=NIL
        then <ELEM>
      elseif (ORDERING ELEM L:1)
        then <ELEM ! L>
      else <L:1 !(ORDERED\INSERT ELEM L::1)
           >))
```

48

```
(ORDERING
  (LAMBDA (U V)                                        (* ORDERING defines an ordering on
                                                       s-expressions)

    (if (NUMBERP U)
        then (if (NUMBERP V)
                 then (ILEQ U V))
      elseif (NLISTP U)
        then (if (NUMBERP V)
                 then T
               elseif (NLISTP V)
                 then (ALPHORDER U V))
      elseif (NUMBERP V) or (NLISTP V)
        then T
      elseif (EQUAL U:1 V:1)
        then (ORDERING U::1 V::1)
      else (ORDERING U:1 V:1))
```

49

```
(PRINT\PROVED
  (LAMBDA (X)
    (PRINTLINES T "Proved" (INFIX\PRINT3 X)
                T))
```

50

```
(PRINT\REWRITE
  (LAMBDA (OLDFORM NEWFORM)
    (if RULETR
        then (PRINTLINES T "Rewriting" (INFIX\PRINT3 OLDFORM)
                         "as"
                         (INFIX\PRINT3 NEWFORM)
                         T))
```

51

```
(REMAFACT
  (LAMBDA (X FACT)                                   (* Removes factor FACT from expression X)
                                                     (* Returns NIL if FACT is not found)
    (if (NLISTP X)
        then (if X=FACT
                 then 1
               elseif (NUMBERP FACT) and (NUMBERP X) and (REMAINDER X FACT)=0
                 then X/FACT
               else NIL)
      elseif (EQUAL X FACT)
        then 1
      elseif X:OPR=MULTOP
        then (PROG (XTERM OLDTERMS NOTFOUND)
                   (OLDTERMS←NIL)
                   (NOTFOUND←T)
                   (X←X:ARGS)
                   (while X and NOTFOUND do (PROGN XTERM←(REMAFACT X:1 FACT)
                                                   (if XTERM=NIL
                                                       then OLDTERMS← <X:1 ! OLDTERMS>
                                                     else (PROGN NOTFOUND←NIL
                                                                 (if ~(EQUAL XTERM 1)
                                                                     then OLDTERMS← <XTERM ! OLDTERMS>)))
                                                   X←X::1))
                   (RETURN (if NOTFOUND
                               then NIL
                             else (ARG\LIST\TO\OP (SORT*XEV < !(REVERSE OLDTERMS) ! X>)
                                                  MULTOP 1)
      elseif X:OPR=PUROP and (EQUAL FACT X:ARG1)
        then (SIMP\POWER FACT (DO\NARY\OP' ADDOP X:ARG2 (- 1)))
      elseif X:OPR=PUROP and (LISTP FACT) and FACT:OPR=PUROP and (EQUAL X:ARG1 FACT:ARG1)
        then (SIMP\POWER X:ARG1 (SIMP\DIFF X:ARG2 FACT:ARG2))
      elseif X:OPR=ADDOP
        then (PROG (XTERM OLDTERMS FOUND)
                   (OLDTERMS←NIL)
                   (FOUND←T)
                   (X←X:ARGS)
                   (while X and FOUND do (PROGN XTERM←(REMAFACT X:1 FACT)
                                                (if XTERM=NIL
                                                    then FOUND←NIL
                                                  else OLDTERMS← <XTERM ! OLDTERMS>)
                                                X←X::1))
                   (RETURN (if FOUND
                               then (MKLISTPREF ADDOP (SORT*XEV OLDTERMS))
                             else NIL)))
      else NIL))
```

52

```
(REMAFACTFROMLIST
  (LAMBDA (LX FACT)
```

```
(PROG (REMLIST)                                   (* Remove factor FACT from list of factors LX
                                                     or return NIL)
      (REMLIST←(REMAFACT (MKLISTPREF MULTOP LX)
                         FACT))
      (RETURN (if REMLIST=NIL
                  then NIL
                elseif (LISTP REMLIST) and REMLIST:OPR=MULTOP
                  then REMLIST:ARGS
                else <REMLIST>))
```

53

```
(REMFROMALL
  (LAMBDA (LX FACTORS)                            (* Removes ALL items in list FACTORS from
                                                     every expression in the list LX)
    (fetch ARGS of (REMOVEFACTORS (MKLISTPREF ADDOP LX)
                                  FACTORS)))
```

54

```
(REMORMULT
  (LAMBDA (X FACTORS)                             (* Removes each factor from expression X if
                                                     possible, else multiplies X by inverse)
                                                  (* FACTORS is a list of inverse TERMS only)
    (if (LISTP X) and X:OPR=ADDOP
        then (PROG (NEWX XREM FL TERM)
                   (X←X:ARGS)
                   (NEWX←NIL)
                   (while X do (PROGN TERM←X:1
                                      FL←FACTORS
                                      (while FL do (PROGN XREM←(REMAFACT TERM FL:1)
                                                          (if XREM
                                                              then TERM←XREM
                                                            else TERM←(DO\NARY\OP MULTOP FL:1:ARG1 TERM))
                                                          FL←FL::1))
                                      NEWX← <TERM ! NEWX> X←X::1))
                   (RETURN (MKLISTPREF ADDOP (SORT*XEV NEWX)))
      else (REMOVEFACTORS X FACTORS))
```

55

```
(REMOVE*XEV
  (LAMBDA (X Y)                                   (* Delete element X from set Y)
    (if Y=NIL
        then NIL
      elseif (EQUAL X Y:EL1)
        then Y::1
      elseif (ORDERING X Y:EL1)
        then Y
      else <Y:EL1 !(REMOVE*XEV X Y::1)
           >))
```

56

```
(REMOVEFACTORS
  (LAMBDA (X FACTORS)
    (PROG (NEWX)                                  (* Removes each factor in the list FACTORS
                                                     from expression X)
          (while FACTORS do (PROGN NEWX←(REMAFACT X FACTORS:1)
                                   (if NEWX=NIL
                                       then (PROGN (INFIX\PRINT FACTORS:1)
                                                   (PRIN1 "CANNOT BE REMOVED FROM ")
                                                   (INFIX\PRINT X)
                                                   (ERROR "REMOVEFACTORS ERROR")
                                                   NIL))
                                   X←NEWX
                                   FACTORS←FACTORS::1))
          (RETURN X)))
```

57

```
(SET\DEFINITION
  (LAMBDA (FUNNAME)
    (PROG (USEFUN EXPRESS PLIST CCLAUSE FTEST FTPART TCLAUSE LPART FCLAUSE IFEXP)
          (USEFUN←FUNNAME)
          (EXPRESS←(GETP USEFUN 'EXPR))
          (if EXPRESS=NIL
              then (ERROR "function not recognized"))
          (PLIST←EXPRESS:2)
          (CCLAUSE←EXPRESS:3)
          (if CCLAUSE:1~='COND
              then (ERROR "body not if-then-else"))
          (FTEST←CCLAUSE:2)
          (FTPART←FTEST:1)
          (TCLAUSE←FTEST:2)
          (LPART←CCLAUSE::2)
          (FCLAUSE←LPART:1:2)
          (IFEXP←(SIMP\PREF IFOP FTPART TCLAUSE FCLAUSE))
          (IFEXP←(SUBLIS '((NEQ . NE)
                           (EQUAL . EQ)
                           (LESSP . LT)
                           (GREATERP . GT)
                           (LEQ . LE)
                           (GEQ . GE))
                         IFEXP))
          (RETURN (DEFINE\FUNC USEFUN PLIST IFEXP))
```

58

```
(SIMP\ALL
  (LAMBDA (X Y)                                    (* Simplify for ALL X (Y))
    (if Y=TRUE
        then TRUE
      elseif Y=FALSE
        then FALSE
      else (SIMP\PREF ALLOP X Y))
```

59

```
(SIMP\ASSIGN
  (LAMBDA (x i v)                                  (* Edited by R. Bates on 10-FEB-78;
                                                   from version 32)
    (SELECTQ i:SYNTACTICTYPE
             (ArrayAccess (if (NLISTP i:Array)
                              then (create ArrayWrite
                                           Array ← x
                                           Index ← i:Index
                                           Value ← v)
                            else (SIMP\ASSIGN x i:Array (create ArrayWrite
                                                               Array ←(XEVAL (baseNameSubst i:Array x))
                                                               Index ← i:Index
                                                               Value ← v)
             (RecordAccess (if (NLISTP i:Record)
                               then (create RecordWrite
                                            Record ← x
                                            Field ← i:Field
                                            Value ← v)
                             else (SIMP\ASSIGN x i:Record (create RecordWrite
                                                                  Record ←(XEVAL (baseNameSubst i:Record x))
                                                                  Field ← i:Field
                                                                  Value ← v)
             (HeapOrFileAccess (if (NLISTP i:HeapOrFile)
                                   then (create HeapOrFileWrite
                                                HeapOrFile ← x
                                                Pointer ← i:Pointer
                                                Value ← v)
                                 else (SIMP\ASSIGN x i:HeapOrFile (create HeapOrFileWrite
                                                                          HeapOrFile ←(XEVAL (baseNameSubst
                                                                                              i:HeapOrFile x)
```

```
                                                                    )
                                                  Pointer ← i:Pointer
                                                  Value ← v]

        (PRINTLINES T "*** Unrecognized variable" i T])
```

60

```
(SIMP\A\SET
  [LAMBDA (A I X)                                  (* Simplify set A sub I to X)
    (if (NLISTP A)
        then (SIMP\PREF ASETOP A I X)
      elseif A:OPR=ASETOP
        then (if (SIMP\EQ A:ARG2 I)=TRUE
                 then (SIMP\A\SET A:ARG1 I X)
               else (SIMP\PREF ASETOP A I X))
      else (SIMP\PREF ASETOP A I X))
```

61

```
(SIMP\A\SUB
  [LAMBDA (A I)                                    (* Simplify the value of the Ith element of
                                                   the array A)
    (if (NLISTP A)
        then (SIMP\PREF ASUBOP A I)
      elseif A:OPR=ASETOP
        then [PROG ((INDEQ (SIMP\EQ I A:ARG2)))
                   (if INDEQ=TRUE
                       then (RETURN A:ARG3)
                     elseif INDEQ=FALSE
                       then (RETURN (SIMP\A\SUB A:ARG1 I))
                     else (RETURN (SIMP\PREF ASUBOP A I]
      elseif A:OPR=SWAPOP
        then (if (SIMP\EQ I A:ARG2)=TRUE
                 then (SIMP\A\SUB A:ARG1 A:ARG3)
               elseif (SIMP\EQ I A:ARG3)=TRUE
                 then (SIMP\A\SUB A:ARG1 A:ARG2)
               else (SIMP\PREF ASUBOP A I))
      else (SIMP\PREF ASUBOP A I])
```

62

```
(SIMP\DIFF
  [LAMBDA (X Y)                                    (* Simplify the difference of X and Y by
                                                   adding X to minus Y)
    (DO\NARY\OP ADDOP X (SIMP\NEG Y))
```

63

```
(SIMP\DIV
  [LAMBDA (X Y)                                    (* Simplify integer division of X by Y.)
    (if Y=0
        then (SIMP\PREF DIVOP X 0)
      elseif X=0
        then 0
      elseif Y=1 and .(ISINT X)
        then X
      elseif (NUMBERP X) and (NUMBERP Y)
        then X/Y
      elseif (NEG\TERM X) and (NEG\TERM Y)
        then (SIMP\DIV (SIMP\NEG X)
                       (SIMP\NEG Y))
      elseif (NEG\TERM X)
        then (SIMP\NEG (SIMP\DIV (SIMP\NEG X)
                                 Y))
      elseif (NEG\TERM Y)
        then (SIMP\NEG (SIMP\DIV X (SIMP\NEG Y)))
      elseif X~=POSINF and Y=POSINF
        then 0
```

```
    elseif X=POSINF or Y=POSINF
      then (SIMP\PREF DIVOP X Y)
    elseif (EQUAL X Y)
      then 1
    elseif (SIMP\LT X Y)=TRUE and (SIMP\LE 0 X)=TRUE
      then 0
    elseif (SIMP\LT Y X)=TRUE and (SIMP\LE X 0)=TRUE
      then 0
    else (SIMP\PREF DIVOP X Y))
```

64

```
(SIMP\EQ
  (LAMBDA (X Y)                                   (* Simplify X equal Y. Numeric denominator of
                                                  XEVALed form is positive.)
    (if (EQUAL X Y)
        then TRUE
      else (PROG (DIFF)
                 (DIFF←(if (ORDERING X Y)
                           then (DO\NARY\OP ADDOP X (SIMP\NEG Y))
                         else (DO\NARY\OP ADDOP Y (SIMP\NEG X)
                 (if DIFF=0
                     then (RETURN TRUE)
                   elseif (NUMERICAL DIFF)
                     then (RETURN FALSE)
                   elseif (NLISTP DIFF)
                     then (RETURN (SIMP\PREF EQOP 0 DIFF))
                   elseif (NEG\TERM DIFF)
                     then (RETURN (SIMP\EQ 0 (SIMP\NEG DIFF)))
                   elseif DIFF:OPR=MULTOP
                     then (RETURN (SIMP\PREF EQOP 0 (CONSTANT\FACTORS DIFF):EL2))
                   elseif DIFF:OPR=DIVOP
                     then (RETURN (if (NUMBERP DIFF:ARG2)
                                      then (DO\NARY\OP ANDOP (SIMP\LE (-DIFF:ARG2)
                                                                      DIFF:ARG1)
                                                             (SIMP\LE DIFF:ARG1 DIFF:ARG2))
                                    else (SIMP\PREF EQOP 0 DIFF)))
                   else (RETURN (SIMP\PREF EQOP 0 DIFF))
```

65

```
(SIMP\EQUIVALENT
  (LAMBDA (X Y)                                   (* Simplify X logically equivalent to Y)
    (if X=TRUE
        then Y
      elseif X=FALSE
        then (SIMP\NOT Y)
      elseif Y=TRUE
        then X
      elseif Y=FALSE
        then (SIMP\NOT X)
      elseif (EQUAL X Y)
        then TRUE
      elseif (EQUAL X (SIMP\NOT Y))
        then FALSE
      else (MKLISTPREF EQVOP (ORDERED\INSERT X <Y>))
```

66

```
(SIMP\GE
  (LAMBDA (X Y)                                   (* Simplify X greater or equal Y)
    (SIMP\LE Y X))
```

67

```
(SIMP\GT
  (LAMBDA (X Y)                                   (* Simplify X > Y)
    (SIMP\LT Y X))
```

68

```
(SIMP\IMP
  (LAMBDA (X Y)                                   (* Simplify X implies Y.
                                                     X and Y have been XEVALed)
    Y←(UNMATCHED\YS CONTEXT Y)
    (if X=TRUE
        then Y
      elseif X=FALSE or Y=TRUE
        then TRUE
      elseif Y=FALSE
        then (SIMP\NOT X)
      elseif (LISTP Y) and Y:OPR=IMPOP
        then (SIMP\IMP (DO\NARY\OP ANDOP X Y:ARG1)
                       Y:ARG2)
      else (SIMP\PREF IMPOP X Y)))
```

69

```
(SIMP\LE
  (LAMBDA (X Y)                                   (* Simplify X less or equal Y)
    (if X=NEGINF or Y=POSINF
        then TRUE
      elseif X=POSINF or Y=NEGINF
        then FALSE
      elseif (EQUAL X Y)
        then TRUE
      else (PROG (DIFF CONST)
                 (DIFF←(DO\NARY\OP ADDOP Y (SIMP\NEG X)))
                 (if (NUMERICAL DIFF)
                     then (RETURN (if (NEG\TERM DIFF)
                                      then FALSE
                                    else TRUE)))
                 (if (LISTP DIFF) and DIFF:OPR=MAXOP
                     then (PROGN CONST←(CONSTANT\TERM DIFF:ARGS)
                                 (if CONST
                                     then (RETURN (if ~(NEG\TERM CONST)
                                                      then TRUE
                                                    else (SIMP\PREF LEOP 0 (ARG\LIST\TO\OP (REMOVE CONST
                                                                                                   DIFF:ARGS)
                                                                                           MAXOP NEGINF]
                 (if (LISTP DIFF) and DIFF:OPR=MINOP
                     then (PROGN CONST←(CONSTANT\TERM DIFF:ARGS)
                                 (if CONST
                                     then (RETURN (if (NEG\TERM CONST)
                                                      then FALSE
                                                    else (SIMP\PREF LEOP 0 (ARG\LIST\TO\OP (REMOVE CONST
                                                                                                   DIFF:ARGS)
                                                                                           MINOP POSINF]
                 (RETURN (SIMP\PREF LEOP 0 DIFF)))
```

70

```
(SIMP\LT
  (LAMBDA (X Y)                                   (* Simplify X < Y)
    (SIMP\LE (DO\NARY\OP ADDOP 1 X)
             Y)))
```

71

```
(SIMP\MOD
  (LAMBDA (X Y)                                   (* Simplify X modulo Y)
    (DO\NARY\OP ADDOP X (SIMP\NEG (DO\NARY\OP MULTOP Y (SIMP\DIV X Y)))
```

72

```
(SIMP\NE
  (LAMBDA (X Y)                                   (* Simplify X NE Y as "(not (X equal Y))."
```

```
(SIMP\NOT (SIMP\EQ X Y))
```

NOTE: that SIMP\NOT usually changes it back to NE.)

73

```
(SIMP\NEG
  (LAMBDA (X)
    (if (NUMBERP X)
        then (-X)
      elseif X=NEGINF
        then POSINF
      elseif X=POSINF
        then NEGINF
      elseif (NLISTP X)
        then (SIMP\PREF NEGOP X)
      elseif X:OPR=NEGOP
        then X:ARG1
      elseif X:OPR=ADDOP
        then (MKLISTPREF ADDOP (SIMP\NEG\LIST X:ARGS))
      elseif X:OPR=MAXOP
        then (MKLISTPREF MINOP (SIMP\NEG\LIST X:ARGS))
      elseif X:OPR=MINOP
        then (MKLISTPREF MAXOP (SIMP\NEG\LIST X:ARGS))
      elseif X:OPR=MULTOP
        then (SIMP\NEG\MULT NIL X:ARGS)
      elseif X:OPR=INVOP
        then (SIMP\PREF INVOP (SIMP\NEG X:ARG1))
      elseif X:OPR=DIVOP
        then (SIMP\PREF DIVOP (SIMP\NEG X:ARG1)
                              X:ARG2)
      elseif X:OPR=PWROP and (NUMBERP X:ARG2) and ~(EVENP X:ARG2)
        then (SIMP\PREF PWROP (SIMP\NEG X:ARG1)
                              X:ARG2)
      else (SIMP\PREF NEGOP X)))
```

(* Simplify the negative of an arithmetic expression)

74

```
(SIMP\NEG\LIST
  (LAMBDA (LX)
    (if LX
        then (ORDERED\INSERT (SIMP\NEG LX:1)
                             (SIMP\NEG\LIST LX::1)))
```

(* Make each element of the list LX negative)

75

```
(SIMP\NEG\MULT
  (LAMBDA (HLX TLX)
    (if TLX=NIL
        then (PROG (ARGS)
                   (ARGS←(REVERSE HLX))
                   (RETURN <MULTOP !(ORDERED\INSERT (SIMP\NEG ARGS:EL1)
                                                    ARGS::1)
                                   >))
      else (if (NEG\TERM TLX:1)
               then <MULTOP !(ORDERED\INSERT (SIMP\NEG TLX:1)
                                             <!(REVERSE HLX) ! TLX::1>)
                    >
             else (SIMP\NEG\MULT <TLX:1 ! HLX> TLX::1)))
```

(* Negate product by changing a NEG\TERM (if present) or by adding a neg sign)

76

```
(SIMP\NOT
  (LAMBDA (X)
    (if X=TRUE
```

(* Simplify logical negation)
(* X is canonical. Returns canonical)

```
      then FALSE
    elseif X=FALSE
      then TRUE
    elseif (NLISTP X)
      then (SIMP\PREF NOTOP X)
    elseif X:OPR=NOTOP
      then X:ARG1
    elseif X:OPR=LEOP
      then (SIMP\LT X:ARG2 X:ARG1)
    elseif X:OPR=LTOP
      then (SIMP\LE X:ARG2 X:ARG1)
    elseif X:OPR=NEOP
      then (MKLISTPREF EQOP X:ARGS)
    elseif X:OPR=EQOP
      then (MKLISTPREF NEOP X:ARGS)
    elseif X:OPR=ALLOP
      then (SIMP\SOME X:ARG1 (SIMP\NOT X:ARG2))
    elseif X:OPR=SOMEOP
      then (SIMP\ALL X:ARG1 (SIMP\NOT X:ARG2))
    elseif X:OPR=OROP
      then (MKLISTPREF ANDOP (SIMP\NOT\LIST X:ARGS))
    elseif X:OPR=ANDOP
      then (MKLISTPREF OROP (SIMP\NOT\LIST X:ARGS))
    else (SIMP\PREF NOTOP X))
```

77

```
(SIMP\NOT\LIST
  (LAMBDA (LX)                                   (* Not's all members of and/or argument list
                                                    LX which is already XEVALed)
    (if LX
        then (ORDERED\INSERT (SIMP\NOT LX:1)
                             (SIMP\NOT\LIST LX::1))
```

78

```
(SIMP\PAIR
  (LAMBDA (X Y)
    <X ! Y>))
```

79

```
(SIMP\POWER
  (LAMBDA (X Y)                                  (* Simplify X to the Y power)
    (PROG (FACTORS)
          (RETURN (if Y=1
                      then X
                    elseif (NEG\TERM Y)
                      then (MKINV (SIMP\POWER X (SIMP\NEG Y)))
                    elseif Y=POSINF and X=POSINF
                      then POSINF
                    elseif X=POSINF or X=NEGINF or Y=POSINF or Y=NEGINF
                      then (SIMP\PREF PWROP X Y)
                    elseif Y=0
                      then 1
                    elseif X=0 and (NUMBERP Y)
                      then 0
                    elseif X=1
                      then 1
                    elseif (LISTP X) and X:OPR=PWROP
                      then (SIMP\POWER X:ARG1 (DO\NARY\OP MULTOP X:ARG2 Y))
                    elseif (LISTP X) and X:OPR=INVOP
                      then (MKINV (SIMP\POWER X:ARG1 Y))
                    elseif (LISTP X) and FACTORS~(GETFACTORS X) and ~(EQUAL FACTORS:1 X)
                      then
```

(* The elseif test before this, protects from the case where (GETFACTORS X) = <X> which causes MKPWRL to loop forever.)

```
                (MKLISTPREF MULTOP (MKPWRL <(REMOVEFACTORS X FACTORS) ! FACTORS> Y))
        elseif (NUMBERP Y)
          then (if (NUMBERP X)
                   then (DO\POWER X Y)
                 elseif (LISTP X) and X:OPR=ADDOP
                   then (DO\POWER X Y)
                 elseif (NEG\TERM X) and (EVENP Y)
                   then (SIMP\PREF PWROP (SIMP\NEG X)
                                Y)
                 else (SIMP\PREF PWROP X Y))
        else (SIMP\PREF PWROP X Y))
```

80

```
(SIMP\PREF
  (LAMBDA X

    (for I from 1 to X collect (ARG X I))
```

(* Makes a list out of NOSPREAD argument X via the INTERLISP ARG function.)

81

```
(SIMP\R\ACCESS
  (LAMBDA (R F)
    (if (NLISTP R)
        then (SIMP\PREF RECACCESSOP R F)
      elseif R:OPR-RECSETOP
        then (if R:ARG2=F
                 then R:ARG3
               else (SIMP\R\ACCESS R:ARG1 F))
      else (SIMP\PREF RECACCESSOP R F))
```

82

```
(SIMP\SOME
  (LAMBDA (X Y)

    (if Y-TRUE
        then TRUE
      elseif Y-FALSE
        then FALSE
      else (SIMP\PREF SOMEOP X Y))
```

(* Simplify for some X (Y))

83

```
(SIMP\SWAP
  (LAMBDA (A I J)

    (if (EQUAL I J)
        then A
      else (SIMP\PREF SWAPOP A I J))
```

(* Simplify SWAP of Ith and Jth element of array A)

84

```
(SIMP\USER\FUNC
  (LAMBDA (F ACTARGS)
    (PROG (ENTRYSTATE FORMARGS FVAL)
          (ENTRYSTATE-STATE)
          (FORMARGS-(FORMAL\PARMS F))
          (STATE-(BIND\ARGS NIL FORMARGS ACTARGS))
          (FVAL-(XEVAL2 (FUNC\BODY F)))
          (STATE-ENTRYSTATE)
          (if (LISTP FVAL) and FVAL:OPR=IFOP
              then (RETURN (MKLISTPREF F ACTARGS))
            else (PRINT\REWRITE (MKLISTPREF F ACTARGS)
                                FVAL)
                 (RETURN FVAL))
```

(* Evaluate user defined function.)

85

```
(SORT*XEV
  (LAMBDA (LX)                                    (* Sorts the list LX into the order defined
                                                     by ORDERING)
    (if LX
        then (ORDERED\INSERT LX:1 (SORT*XEV LX::1))
```

86

```
(SUPER\PUT\IN
  (LAMBDA (LX YL OPCODE)

        (* Applies n-ary op to operands pairwise and ordered. LX is a nary'd and individually XEVALed list of items to
        be put into the already pairwise compared XEVALed (but not necessarily ordered) AND (or OR, ADD, MULT, MAX,
        MIN) list YL)


    (PROG (X R Y ZL NOTX MKFCN TOSSVALUE NEGFCN)
          (X←(GETP OPCODE 'NARYS))
          (if X=NIL
              then (PRINTLINES OPCODE "not defined as nary op ****"))
          (MKFCN←X:1)
          (TOSSVALUE←X:2)
          (NEGFCN←X:3)
          (for X in LX do (if ~(EQUAL X TOSSVALUE)
                              then NOTX←(BLKAPPLY NEGFCN <X>)
                                   ZL←NIL
                                   (while YL do (Y←YL:1)
                                                (R←(BLKAPPLY MKFCN <X Y NOTX>))
                                                (if (EQUAL R (SIMP\PREF OPCODE X Y))
                                                    or (EQUAL R (SIMP\PREF OPCODE Y X))
                                                   then ZL← <Y ! ZL> YL←YL::1
                                                  else X←R
                                                       NOTX←(BLKAPPLY NEGFCN <X>)
                                                       YL← < ! ZL ! YL::1> ZL←NIL))
                                   YL← <X ! ZL>))
          (RETURN (ARG\LIST\TO\OP (SORT*XEV YL)
                                  OPCODE TOSSVALUE))
```

87

```
(UNION*XEV
  (LAMBDA (X Y)                                   (* Unions element X into the set Y)
    (if Y=NIL
        then <X>
      elseif (EQUAL X Y:EL1)
        then Y
      elseif (ORDERING X Y:EL1)
        then <X ! Y>
      else <Y:EL1 ! (UNION*XEV X Y::1)
           >))
```

88

```
(UNMATCHED\YS
  (LAMBDA (X Y)
    (PROG (HL CL SCL H C)                         (* Finds elements of AND LIST Y not matching
                                                     elements of AND LIST X)

          (HL←(AND\TO\ARG\LIST X))
          (CL←(AND\TO\ARG\LIST Y))
          (H←HL:1)
          (C←CL:1)
          (while CL and HL do (if (EQUAL\QUANTIFIED H C)
                                  then (if (IN\SET? IMPOP XEVALTRACESET)
                                           then (PRINT\PROVED C))
                                       CL←CL::1
                                       (if CL
                                           then C←CL:1)
```

```
                elseif (ORDERING H C)
                  then HL←HL::1
                       (if HL
                          then H←HL:1)
                else SCL← <C | SCL> CL←CL::1
                     (if CL
                        then C←CL:1)))
        (if SCL=NIL and CL=NIL
           then (RETURN TRUE)
         else (RETURN (ARG\LIST\TO\AND < |(REVERSE SCL) | CL>)))
```

89

```
(XEVAL
  (LAMBDA (PREFIX)
    (SIMPLIFY\BREAK PREFIX)
    (XEVAL2 PREFIX)))
```

90

```
(XEVAL2
  (LAMBDA (PREFIX)
```

(* Simplify expressions in prefix form by finding each operand and its args. STATE and CNDS are global variables that contain the current program state and evaluation conditions. The operators processed by functions preceded with "X\" control their own order of XEVALing their arguments. For the operators processed by functions preceded with "SIMP\", all arguments are XEVALed before the "SIMP\" function is applied.)

```
    (PROG (INTTH)
          (INTTH←PREFIX)
          (if (NLISTP PREFIX)
              then (if (CONSTANT? PREFIX)
                       then (RETURN PREFIX)
                     else (RETURN (APPLY\STATE STATE PREFIX)))
            else (RETURN (PROG (OP OP2 OP3 EVLPREF XARGS)
                               (OP←PREFIX:OPR)
                               (if (IN\SET? OP XEVALTRACESET)
                                   then (PRINTLINES T "==>" (INFIX\PRINT3 PREFIX)
                                                    T))
                               (EVLPREF←(if (DEFINED? UFUNS OP)
                                            then (SIMP\USER\FUNC OP (for X in PREFIX:ARGS
                                                                       collect (XEVAL2 X)))
                                          elseif OP3←(GETP OP 'EVALARGS)
                                            then (if (EQUAL OP3 TRUE)
                                                     then XARGS←(for X in PREFIX:ARGS collect (XEVAL2 X))
                                                   else XARGS←PREFIX:ARGS)
                                                 OP2←(GETP OP 'EVFUN)
                                                 (if (EQUAL OP2:MEM2 'NARGS)
                                                     then (BLKAPPLY OP2:MEM1 <XARGS>)
                                                   else (BLKAPPLY OP2:MEM1 XARGS))
                                          else (MKLISTPREF OP (for X in PREFIX:ARGS collect (XEVAL2 X)))))
                               (if (IN\SET? OP XEVALTRACESET)
                                   then (PRINTLINES T "<==" (INFIX\PRINT3 EVLPREF)
                                                    T))
                               (RETURN EVLPREF)))
```

91

```
(XEVAL\TURN\OFF
  (LAMBDA (LX)
    (if LX='ALL
        then XEVALTRACESET←NIL
      elseif LX=NIL
        then XEVALTRACESET
      else XEVALTRACESET←(REMOVE*XEV LX:1 XEVALTRACESET)
           (XEVAL\TURN\OFF LX::1))
```

92

```
(XEVAL\TURN\ON
  (LAMBDA (LX)
    (if LX=NIL
        then XEVALTRACESET
      else XEVALTRACESET←(UNION*XEV LX;1 XEVALTRACESET)
           (XEVAL\TURN\ON LX;;1)))
```

93

```
(X\ADD\N
  (LAMBDA (LX)

    (SUPER\PUT\IN (NARY\X LX ADDOP)
                  NIL ADDOP))
```

(* XEVAL's and n-ary's list LX and pairwise adds and orders elements)

94

```
(X\ALPHA
  (LAMBDA (F X Y)

        (* If F is defined at X, then set F(X) to Y, else add the PAIR (x,y) to F)

    (if (NULL\SET? F)
        then <(SIMP\PAIR X Y) >
      elseif (EQUAL X F;1;MEM1)
        then <(SIMP\PAIR X Y) ! F;;1>
      elseif (ORDERING X F;1;MEM1)
        then <(SIMP\PAIR X Y) ! F>
      else <F;1 ! (X\ALPHA F;;1 X Y)
           >))
```

95

```
(X\AND\N
  (LAMBDA (LX)

    (SUPER\PUT\IN (NARY\X LX ANDOP)
                  NIL ANDOP))
```

(* XEVAL's and n-ary's list LX and pairwise and's and orders elements)

96

```
(X\IF
  (LAMBDA (TEST TPART FPART)
    (PROG (X Y Z OLDCONTEXT X\IFRESULT)

          (X←(XEVAL2 TEST))
          (OLDCONTEXT←CONTEXT)
          (X\IFRESULT←(if (SIMP\IMP CONTEXT X)=TRUE
                          then (XEVAL2 TPART)
                        elseif (SIMP\IMP CONTEXT (SIMP\NOT X))=TRUE
                          then (XEVAL2 FPART)
                        else (RESETVAR UFUNS NIL (PROGN CONTEXT←(DO\NARY\OP ANDOP OLDCONTEXT X)
                                                        Y←(XEVAL2 TPART)
                                                        CONTEXT←(DO\NARY\OP ANDOP OLDCONTEXT (SIMP\NOT X))
                                                        Z←(XEVAL2 FPART)
                                                        (if (EQUAL Y Z)
                                                            then Y
                                                          elseif (LISTP X) and X:OPR=NOTOP
                                                            then (SIMP\PREF IFOP X:ARG1 Z Y)
                                                          else (SIMP\PREF IFOP X Y Z)
          (CONTEXT←OLDCONTEXT)
          (RETURN X\IFRESULT))
```

(* Simplify (if X then Y else Z). TEST to avoid evaluating both Y and Z)

97

```
(X\IMP
  (LAMBDA (X Y)
    (PROG (OLDCONTEXT H C X\IMPY)

          (OLDCONTEXT←CONTEXT)
          (H←(XEVAL2 X))
          (CONTEXT←(DO\NARY\OP ANDOP OLDCONTEXT H))
          (X\IMPY←(if CONTEXT=FALSE
                      then TRUE
                    else (SIMP\IMP H (XEVAL2 Y)
          (CONTEXT←OLDCONTEXT)
          (RETURN X\IMPY))
```

(* Simplify X implies Y.
Unnecessary to XEVAL2 Y if X is FALSE)

98

```
(X\MAX\N
  (LAMBDA (LX)

    (SUPER\PUT\IN (NARY\X LX MAXOP)
                  NIL MAXOP))
```

(* XEVAL's and n-ary's list LX and pairwise MAX's and orders elements)

99

```
(X\MIN\N
  (LAMBDA (LX)

    (SUPER\PUT\IN (NARY\X LX MINOP)
                  NIL MINOP))
```

(* XEVAL's and n-ary's list LX and pairwise MIN's and orders elements)

100

```
(X\MULT\N
  (LAMBDA (LX)

    (SUPER\PUT\IN (NARY\X LX MULTOP)
                  NIL MULTOP))
```

(* XEVAL's and n-ary's list LX and pairwise multiplies and orders elements)

101

```
(X\ORDERED\MEMBER?
  (LAMBDA (ELEM L)

    (if L=NIL
        then NIL
      elseif (EQUAL ELEM L:1)
        then T
      elseif (ORDERING ELEM L:1)
        then NIL
      else (X\ORDERED\MEMBER? ELEM L::1))
```

(* TESTS if the element ELEM is a member of the ordered list L)

102

```
(X\OR\N
  (LAMBDA (LX)

    (SUPER\PUT\IN (NARY\X LX OROP)
                  NIL OROP))
)
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY
(BLOCK: XEVALBLOCK SIMP\ASSIGN SIMP\R\ACCESS ALPHA AND\TO\ARG\LIST APPLY\STATE APPLY\F\OF\X ARG\LIST\TO\AND
        ARG\LIST\TO\OP BIND\ARGS CONSTANT\TERM CONSTANT\FACTORS CONSTANT\MULT\FACTORS DEFINE\FUNC DEFINED?
        DISTRIBUTE DOMAIN DO\NARY\OP DO\POWER EQUAL\QUANTIFIED UNION#XEV EVENP FUNC\BODY FORMAL\PARMS IN\SET?
        SIMP\ALL SIMP\A\SET SIMP\A\SUB SIMP\DIFF SIMP\DIV SIMP\EQ SIMP\EQUIVALENT SIMP\USER\FUNC SIMP\GE
        SIMP\GT SIMP\IMP SIMP\LE SIMP\LT SIMP\MOD SIMP\NE SIMP\NEG SIMP\NEG\LIST SIMP\NEG\MULT SIMP\NOT
        SIMP\NOT\LIST SIMP\POWER SIMP\SOME SIMP\SUMP N\ADD N\AND NARY\X NEG\MAX NEG\MULT NEG\TERM N\MAX N\MIN
        N\MULT N\OR ORDERED\INSERT ORDERING PRINT\PROVED REMOVE#XEV PRINT\REWRITE SET\DEFINITION SORT#XEV
        SUPER\PUT\IN UNMATCHED\YS X\ADD\N X\ALPHA X\AND\N XEVAL XEVAL2 X\IF X\IMP X\MAX\N X\MIN\N X\MULT\N
```

```
        X\ORDERED\MEMBER? X\OR\N XEVAL\TURN\OFF XEVAL\TURN\ON COMMONFACTORS FACTORIZE FOUNDIN GETFACTORS
        GETMAXCONSTS GETNEGPWR GETNUMDEN ISINT MKINV MKPWRL MKQUOT NUMERICAL REMAFACT REMAFACTFROMLIST
        REMFROMALL REMORMULT REMOVEFACTORS
        (BLOCK APPLYFNS SIMP\ASSIGN SIMP\ALL SIMP\A\SET SIMP\R\ACCESS SIMP\A\SUB SIMP\DIFF SIMP\DIV SIMP\EQ
                SIMP\EQUIVALENT SIMP\GE SIMP\GT SIMP\LE SIMP\LT SIMP\MOD SIMP\NE SIMP\NEG SIMP\NOT
                SIMP\POWER SIMP\SOME SIMP\SWAP N\ADD N\AND N\MAX N\MIN N\MULT N\OR X\ADD\N X\AND\N X\IF
                X\IMP X\MAX\N X\MIN\N X\MULT\N X\OR\N MKINV MKQUOT)
        (GLOBALVARS CONTEXT RULETR STATE UFUNS UNDEFINED XEVALTRACESET)
        (ENTRIES XEVAL XEVAL\TURN\OFF XEVAL\TURN\ON SET\DEFINITION SIMP\NEG NEG\TERM GETNUMDEN ISINT)
        (NOLINKFNS . T))
]
(DECLARE: DOEVAL@COMPILE

(PUTPROPS NULL\SET? MACRO (X (CONS (QUOTE NULL)
                                   X)))

(PUTPROPS SIMP\PAIR MACRO (X (CONS (QUOTE CONS)
                                   X)))

(PUTPROPS SIMP\PREF MACRO (X (CONS (QUOTE LIST)
                                   X)))

(PUTPROPS CONSTANT? MACRO (LAMBDA (A)
                            (COND
                              ((OR (NUMBERP A)
                                   (EQ A NEGINF)
                                   (EQ A POSINF)
                                   (EQ A TRUE)
                                   (EQ A FALSE))

(PUTPROPS MKLISTPREF MACRO (X (CONS (QUOTE CONS)
                                    X)))
)
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA SIMP\PREF)
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (3396 67974 (ALPHA 3408 . 3833) (AND\TO\ARG\LIST 3837 . 4157) (APPLY\F\OF\X 4161 . 4518) (
APPLY\STATE 4522 . 4744) (ARG\LIST\TO\AND 4748 . 5071) (ARG\LIST\TO\OP 5075 . 5400) (BIND\ARGS 5404 . 5730) (
COMMONFACTORS 5734 . 6655) (CONSTANT? 6659 . 6883) (CONSTANT\FACTORS 6887 . 7467) (CONSTANT\MULT\FACTORS 7471
 . 8308) (CONSTANT\TERM 8312 . 8645) (DEFINED? 8649 . 8951) (DEFINE\FUNC 8995 . 9239) (DISTRIBUTE 9243 . 9536)
(DOMAIN 9540 . 9859) (DO\NARY\OP 9863 . 11181) (DO\POWER 11185 . 11881) (EQUAL\QUANTIFIED 11885 . 12409) (
EVENP 12413 . 12566) (FACTORIZE 12570 . 13352) (FORMAL\PARMS 13356 . 13635) (FOUNDIN 13639 . 14163) (FUNC\BODY
 14167 . 14388) (GETFACTORS 14392 . 15652) (GETMAXCONSTS 15656 . 16179) (GETNEGPWR 16183 . 17153) (GETNUMDEN
17157 . 18239) (IN\SET? 18243 . 18445) (ISINT 18449 . 19246) (MKINV 19250 . 20186) (MKLISTPREF 20190 . 20380)
(MKPWRL 20384 . 20697) (MKQUOT 20701 . 20915) (NARY\X 20919 . 21397) (NEG\MAX 21401 . 22047) (NEG\MULT 22051 .
 22321) (NEG\TERM 22325 . 23210) (NULL\SET? 23214 . 23258) (NUMERICAL 23262 . 23806) (N\ADD 23810 . 26411) (
N\AND 26415 . 27316) (N\MAX 27320 . 27940) (N\MIN 27944 . 28564) (N\MULT 28568 . 34604) (N\OR 34608 . 35207) (
ORDERED\INSERT 35211 . 35578) (ORDERING 35582 . 36230) (PRINT\PROVED 36234 . 36326) (PRINT\REWRITE 36330 .
36516) (REMAFACT 36520 . 38643) (REMAFACTFROMLIST 38647 . 39142) (REMFROMALL 39146 . 39435) (REMORMULT 39439 .
 40382) (REMOVE#XEV 40386 . 40733) (REMOVEFACTORS 40737 . 41333) (SET\DEFINITION 41337 . 42258) (SIMP\ALL
42262 . 42515) (SIMP\ASSIGN 42519 . 43916) (SIMP\A\SET 43920 . 44353) (SIMP\A\SUB 44357 . 45265) (SIMP\DIFF
45269 . 45497) (SIMP\DIV 45501 . 46689) (SIMP\EQ 46693 . 48002) (SIMP\EQUIVALENT 48006 . 48547) (SIMP\GE 48551
 . 48702) (SIMP\GT 48706 . 48842) (SIMP\IMP 48846 . 49304) (SIMP\LE 49398 . 50824) (SIMP\LT 50828 . 50998) (
SIMP\MOD 51002 . 51214) (SIMP\NE 51218 . 51489) (SIMP\NEG 51493 . 52725) (SIMP\NEG\LIST 52729 . 52985) (
SIMP\NEG\MULT 52989 . 53642) (SIMP\NOT 53646 . 54825) (SIMP\NOT\LIST 54829 . 55135) (SIMP\PAIR 55139 . 55187)
(SIMP\POWER 55191 . 57003) (SIMP\PREF 57007 . 57271) (SIMP\R\ACCESS 57275 . 57585) (SIMP\SOME 57589 . 57867) (
SIMP\SWAP 57871 . 58136) (SIMP\USER\FUNC 58140 . 58764) (SORT#XEV 58768 . 59021) (SUPER\PUT\IN 59025 . 60263)
(UNION#XEV 60267 . 60619) (UNMATCHED\YS 60623 . 61607) (XEVAL 61611 . 61706) (XEVAL2 61710 . 63606) (
XEVAL\TURN\OFF 63610 . 63860) (XEVAL\TURN\ON 63864 . 64052) (X\ADD\N 64056 . 64305) (X\ALPHA 64309 . 64752) (
X\AND\N 64756 . 65006) (X\IF 65010 . 66035) (X\IMP 66039 . 66546) (X\MAX\N 66550 . 66800) (X\MIN\N 66804 .
67054) (X\MULT\N 67058 . 67316) (X\ORDERED\MEMBER? 67320 . 67721) (X\OR\N 67725 . 67971)))))
STOP
```