<AFFIRM>INFIXPRINT..19 17-Jun-81 15:08:13

```
(FILECREATED "31-Mar-81 18:53:00" <AFFIRM>INFIXPRINT..19 73813

     changes to:  SeparateWithBlanks? INFIXPRINTCOMS INFIX\PRINTFNS

     previous date: "28-Mar-81 14:55:31" <AFFIRM>INFIXPRINT..18)


(PRETTYCOMPRINT INFIXPRINTCOMS)

(RPAQQ INFIXPRINTCOMS [(* ↑ The infix print initialization function.)
                       (FNS #INIT\INFIX\PRINT EmbeddedQuantifier MakeProgramLevels OperatorPrecedence)
                       (* ↑ The main entry points for infix printing.)
                       (FNS INFIX\PRINT INFIX\PRINT3 INFIX\PRINT4 PRINT\PROGRAM)
                       (* ↑ The internal functions for infix printing.)
                       (FNS * INFIX\PRINTFNS)
                       (FNS * DeCanonicalizeFNS)
                       (P (#INIT\INFIX\PRINT))
                       (VARS (MagicCount 5))
                       (* ↑ The information for compiling and block compiling.)
                       (MACROS * INFIXPRINTMACROS)
                       (DECLARE: DOEVAL@COMPILE (PROP MACRO CHRCT))
                       (BLOCKS * INFIXPRINTBLOCKS)
                       (DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
                                 (ADDVARS (NLAMA)
                                          (NLAML)
                                          (LAMA Max])
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* ↑ The infix print initialization function.) ]

(DEFINEQ

                                                                    1

(#INIT\INFIX\PRINT
  [LAMBDA NIL                                    (* R.Bates " 6-Aug-79 14:10")
    (/SET '#LIST\OF\SYSTEM\OPS
          <MAXIO MINIO IMPIO EQVIO NOTIO NEIO LEIO GEIO ORIO ANDIO ADDIO DIFFIO NEGIO MULTIO INVIO
            QUOTIENTIO LTIO GTIO EQIO DIVIO MODIO ALLIO SOMEIO ! '(CC CO OC OO NOP ARRAY ENTRY IN
                                                                    EXIT)
            >)

          (* List of op's for which ((A op B) associative-op C) differs from (A op (B associative-op C)) where op and
          associative-op have same hierarchy)

    (/SET '#NO\LEFT\ASSOC\OPS <IMPIO DIFFIO QUOTIENTIO DIVIO MODIO>)

          (* List of op's for which ((A associative\op B) op C) differs from (A associative\op (B op C)) where op and
          associative\op have same hierarchy)

    (/SET '#NO\RIGHT\ASSOC\OPS <IMPIO DIVIO MODIO>)

          (* Global flag CANON being NIL allows INFIX\PRINT and INFIX\PRINT3 (in fact any routine calling DeCanonicalizer) to
          simplify the canonical form of EQ. NE. LE for printing. When CANON is T. such routines will print exactly the
          expression given them including the unreadable canonicalized forms)


          (* Global flag LESS\THAN\FLAG has no effect until CANON is NIL. Then LESS\THAN\FLAG being T allows certain LE's to
          be decanonicalized to equivalent LT's. If LESS\THAN\FLAG is NIL. all LE's will remain LE's)

    (/SET 'LESS\THAN\FLAG T)

          (* Global flag FEWER\PARENS selects which way DeCanonicalize treats the form (PLUS (MINUS A) B
          (MINUS C) D). If FEWER\PARENS is NIL. INFIX\PRINT (which uses DeCanonicalize. gives B + D- (A + C). If FEWER\PARENS is
          T. INFIX\PRINT gives -A + B-C + D. This latter form necessitates a removal of the parentheses in infix printing
          (X-Y) + Z. If CANON is T then the form (PLUS (MINUS A) B (MINUS C) D) will INFIX\PRINT unchanged as -A + B + -C + D. while
          the effect on (X-Y) + Z will remain controlled by FEWER\PARENS)

    (/SET 'FEWER\PARENS NIL)

    (/SET '#ERROR\REPORTING\FLG NIL)        (* Global variable #ERROR\REPORTING\FLG controls the
                                            printing of the error message for trying to INFIX\PRINT
                                            something not on the list of functions or operators)
    (/SET 'DOTS\FLAG NIL)                   (* Global variable DOTS\FLAG. if non-NIL. causes
                                            INFIX\PRINT etc. to print "..."
                                            When the right margin is reached instead of spilling
                                            over the line)

          (* Make the appropriate calls to set up STANFORD and NORMAL terminal character sets. print out forms. and page
          length. Also initialize for a normal terminal.)

    (/SET '#TERMINALPAGELENGTH 24)

    (/SET 'DTVSCANON T)                     (* If DTVSCANON is NIL then RenameAtoms is called on
                                            almost all atoms printed by INFIX\PRINT.)
    (InitTTYCharSet)
    (InitHierarchy)
    (OnTTY)
    (INTERRUPTCHAR 11 '(PRIN1 (CHARACTER 7])

                                                                    2

(EmbeddedQuantifier
  [LAMBDA (operator)                            (* D.Thompson "20-Jun-80 11:34")
    ((Shorten operator)
     MEMB <SOMEIO ALLIO>])

                                                                    3

(MakeProgramLevels
  [LAMBDA (FORMULA)                             (* D.Thompson "23-Jun-80 11:49")
```

```
(* * Preprocesses prefix programs for formula printer.)


(PROG (LRET)
      (SELECTQ FORMULA:SYNTACTICTYPE
               ((assertStatement assumeStatement proveStatement)
                 LRET←
                 (MakeBinaryLevels <FORMULA:assertion> 'ASRT))
               (assignmentStatement LRET← (MakeBinaryLevels <FORMULA:variable FORMULA:expression>
                                                            'ASSIGN))
               (caseStatement (PROG (caseMax caseElement (LEFT (FORMULA:expression))
                                                         (RIGHT (FORMULA:statement)))
                                    (caseMax←0)
                                    (caseElement←(for (caseE caseL caseM caseS)
                                                      in FORMULA:caseElementList
                                                      collect (caseL←caseE:caseLabel)
                                                              (if caseL::1=NIL
                                                                  then caseL←(MakeLevels caseL:1)
                                                                else caseL←(MakeBinaryLevels caseL
                                                                                             'COMMA))
                                                              (caseS←(MakeLevels (if
                                                                                   caseE:statement
                                                                                     else 'NOP)))
                                                              (caseM←(IMAX 2 (ListPlus caseL caseS))
                                                                )
                                                              (caseMax←(IMAX caseMax caseM))
                                                              (<caseM 'LABEL caseL caseS>)))
                                    (if RIGHT
                                        then RIGHT←(MakeLevels RIGHT)
                                             RIGHT← <(ListAdd1 RIGHT)
                                                      'OTHERWISE <1 'otherwise > RIGHT>
                                             caseMax←(Max caseMax RIGHT)
                                             caseElement← < ! caseElement RIGHT>)
                                    (LEFT←(MakeLevels LEFT))
                                    (LRET← <(Max LEFT caseMax)← 1 'CASE LEFT ! caseElement>)))
               [compoundStatement (PROG ((RIGHT (FORMULA:statement)))
                                        (LRET←(FuncMakeLevels RIGHT 'CMPD T]
               (forStatement LRET← (FuncMakeLevels <FORMULA:identifier FORMULA:expression
                                                    (L-CASE FORMULA:direction:LEXEME)
                                                    FORMULA:expression# FORMULA:statement>
                                                   'FOR NIL))
               (goToStatement LRET← (MakeBinaryLevels <FORMULA:label> 'GOTO))
               [ifStatement (PROG ((EXPRESSION (FORMULA:expression))
                                   (LEFT (FORMULA:statement))
                                   (RIGHT (FORMULA:statement#)))
                                  (LRET←(if RIGHT
                                            then (FuncMakeLevels <EXPRESSION LEFT RIGHT. 'IFELSE
                                                                 NIL)
                                          else (FuncMakeLevels <EXPRESSION LEFT> 'IF NIL]
               (labelStatement LRET← (MakeBinaryLevels <FORMULA:label (if FORMULA:simpleStatement
                                                                         else 'NOP)
                                                                >
                                                       'LABEL))
               (procedureStatement (PROG ((LEFT (FORMULA:expression))
                                          (NAME (FORMULA:identifier)))
                                         (if LEFT=NIL
                                             then LRET← <2 'FUNCTION (MakeLevels NAME) >
                                           else (if LEFT::1=NIL
                                                    then LEFT←(MakeLevels LEFT:1)
                                                  else LEFT←(MakeBinaryLevels LEFT 'COMMA))
                                                LRET← <(ListAdd1 LEFT)
                                                       'FUNCTION
                                                       (MakeLevels NAME)
                                                       LEFT>)))
               (repeatStatement (PROG ((RIGHT (FORMULA:expression))
                                       (LEFT (FORMULA:statement)))
                                      (LEFT←(FuncMakeLevels LEFT NIL T))
                                      (RIGHT←(MakeLevels RIGHT))
                                      (LRET← <(Max RIGHT LEFT)← 1 'REPEAT LEFT::2 RIGHT>)))
               [returnStatement (PROG ((RIGHT (FORMULA:expression)))
                                      (if RIGHT
                                          then RIGHT←(MakeLevels RIGHT)
                                               LRET← <(ListAdd1 RIGHT)
                                                      'FUNCTION '(1 return)
                                                      RIGHT>
                                        else LRET←'(1 return]
               (statement LRET← <0 'NOP >)
               (whileStatement LRET← (MakeBinaryLevels <FORMULA:expression FORMULA:statement>
                                                       'WHILE))
               (Unexpected 'ProgramType))
      (RETURN LRET])
```

4

```
(OperatorPrecedence
  [LAMBDA (operator)                                    (* D.Thompson "19-Aug-80 16:47")

          (* * This routine returns the operator precedence of its parameter  Currently the information is stored on the
          property list of atom HIERARCHY.)


    (if (GETP 'HIERARCHY (Shorten operator))
      else 0])
)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* ↑ The main entry points for infix printing.) ]

(DEFINEQ

                                                                    5

(INFIX\PRINT
  [LAMBDA (INFIX\PRINTARG)                                  (* R.Bates "31-Oct-79 14:05")
                                                            (* Infix printout of a VC or program)
    CONTROLKFLG←NIL
    (if ~[PROG (LISPXHIST)
               (RETURN (RESETFORM (INTERRUPTCHAR 11 '(AbortInfixPrint)
                                  T)
                                  (NLSETQ (PROGN (TERPRI)
                                                 #CURRENT\LINELENGTH←(LINELENGTH)
                                                 #NUMBERLINES←1
                                                 [if INFIX\PRINTARG=NIL
                                                     then (PRIN1 NIL)
                                                   else (PROG (LEVD PRINT\SWITCH)
                                                              (LEVD←(MakeLevels INFIX\PRINTARG))
                                                              (if (EQCAR INFIX\PRINTARG IMPIO)
                                                                    or (EQCAR INFIX\PRINTARG EQVIO)
                                                                  then PRINT\SWITCH←5
                                                                elseif (EQCAR INFIX\PRINTARG ANDIO)
                                                                  then PRINT\SWITCH←8
                                                                  else PRINT\SWITCH←0)
                                                              (RESETFORM (GCGAG NIL)
                                                                         (PrintFormula LEVD 0 12
                                                                                       PRINT\SWITCH]
                                                 (TERPRI]
         then (if CONTROLKFLG
                  then NIL
                else (ERROR!))
       else NIL])

                                                                    6

(INFIX\PRINT3
  [LAMBDA (INFIX\PRINTARG)                                  (* R.Bates "31-Oct-79 14:06")
                                                            (* Same as INFIX\PRINT except it starts and ends without
                                                            TERPRIs)
    CONTROLKFLG←NIL
    (if ~[PROG (LISPXHIST)
               (RETURN (RESETFORM (INTERRUPTCHAR 11 '(AbortInfixPrint)
                                  T)
                                  (NLSETQ (PROGN #CURRENT\LINELENGTH←(LINELENGTH)
                                                 (if INFIX\PRINTARG=NIL
                                                     then (PRIN1 NIL)
                                                   else (PROG (LEVD PRINT\SWITCH)
                                                              (#NUMBERLINES←0)
                                                              (LEVD←(MakeLevels INFIX\PRINTARG))
                                                              (if (EQCAR INFIX\PRINTARG IMPIO)
                                                                    or (EQCAR INFIX\PRINTARG EQVIO)
                                                                  then PRINT\SWITCH←5
                                                                elseif (EQCAR INFIX\PRINTARG ANDIO)
                                                                  then PRINT\SWITCH←8
                                                                  else PRINT\SWITCH←0)
                                                              (RESETFORM
                                                                (GCGAG NIL)
                                                                (NLSETQ (PrintFormula LEVD (POSITION)
                                                                                      12 PRINT\SWITCH]
         then (if CONTROLKFLG
                  then NIL
                else (ERROR!))
       else NIL])

                                                                    7

(INFIX\PRINT4
  [LAMBDA (INFIX\PRINTARG)                                  (* R.Bates "31-Oct-79 14:07")
                                                            (* Same as INFIX\PRINT3 except it doesn't reset the
                                                            linecounter)
    CONTROLKFLG←NIL
    (if ~[PROG (LISPXHIST)
               (RETURN (RESETFORM (INTERRUPTCHAR 11 '(AbortInfixPrint)
                                  T)
                                  (NLSETQ (PROGN #CURRENT\LINELENGTH←(LINELENGTH)
                                                 (if INFIX\PRINTARG=NIL
```

```
                                                    then (PRIN1 NIL)
                                                  else (PROG (LEVD PRINT\SWITCH)
                                                             (LEVD←(MakeLevels INFIX\PRINTARG))
                                                             (if (EQCAR INFIX\PRINTARG IMPIO)
                                                                   or (EQCAR INFIX\PRINTARG EQVIO)
                                                                 then PRINT\SWITCH←5
                                                                elseif (EQCAR INFIX\PRINTARG ANDIO)
                                                                  then PRINT\SWITCH←8
                                                                else PRINT\SWITCH←0)
                                                             (RESETFORM
                                                               (GCGAG NIL)
                                                               (NLSETQ (PrintFormula LEVD (POSITION)
                                                                                     12 PRINT\SWITCH]
        then (if CONTROLKFLG
                 then NIL
               else (ERROR!))
      else NIL])
```



```
(PRINT\PROGRAM
  [LAMBDA (TREE PIECE)                                    (* R.Bates "13-Apr-79 17:02")
    (PROG (part (BLOCK (TREE:block)))

          (* * Pretty prints procedures or parts thereof)


          (#CURRENT\LINELENGTH←(LINELENGTH))
          (if PIECE=NIL
              then PIECE←'ALL)
          (if PIECE ~MEMB '(ALL HEAD ARGS ENTRY EXIT BODY)
              then (ERROR PIECE
                          "illegal PRINT\PROGRAM argument. Use ALL, HEAD, ARGS, ENTRY, EXIT, or BODY."
                          T)
            else NIL)
          (#NUMBERLINES←0)
          (EndLine 0)
          (if PIECE MEMB '(ALL HEAD ARGS)
              then (PRIN1 (if (L-CASE TREE:unitKind:LEXEME)
                              else 'program))
                   (PRIN1 " ")
                   (PRIN1 TREE:identifier)
                   (EndLine 0))
          (if PIECE MEMB '(ALL HEAD ENTRY) and part←BLOCK:assertion
              then (INFIX\PRINT4 <'ENTRY part>)
                   (PRIN1 ":")
                   (EndLine 0))
          (if PIECE MEMB '(ALL HEAD EXIT) and part←BLOCK:assertion#
              then (INFIX\PRINT4 <'EXIT part>)
                   (PRIN1 ":")
                   (EndLine 0))
          (if PIECE MEMB '(ALL BODY) and part←BLOCK:compoundStatement
              then (INFIX\PRINT4 part)
                   (EndLine 0])
)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* ↑ The internal functions for infix printing.)]

```
(RPAQQ INFIX\PRINTFNS (AbortInfixPrint ArgsMakeLevels Convert DetermineQuantifierOrder EndLine
                                       FuncMakeLevels GetOpLength InitHierarchy InitTTYCharSet
                                       ListAdd1 ListGT ListPlus Magic MakeBinaryLevels MakeLevels Max
                                       NewMakeLevels NewPrintFormula OldMakeLevels OldPrintFormula
                                       OnTTY PrintArray PrintCase PrintCmpd PrintCommaDot PrintFor
                                       PrintFormula PrintIf PrintInterval PrintLabel PrintMisc1Line
                                       PrintMisc2Lines PrintPrefixFunction PrintProgramPiece
                                       PrintQexpression PrintQuantifierSequences PrintQuantifiers
                                       PrintRepeat PrintSequence PrintSimple PrintSomeAll PrintWhile
                                       PrintingSwitchValue SeparateWithBlanks? SetHierarchy))
(DEFINEQ
```

9

```
(AbortInfixPrint
  [LAMBDA NIL                                          (* R.Bates "11-JAN-79 09:15")
    (if (STKPOS 'INFIX\PRINT\BLOCK)
        then CONTROLKFLG←T
             (ERROR!])
```

10

```
(ArgsMakeLevels
  [LAMBDA (ARGLIST)                                    (* Applies MakeLevels to items in ARGLIST)
    (if ARGLIST
        then <(MakeLevels ARGLIST:1) !(ArgsMakeLevels ARGLIST::1)
             >])
```

11

```
(Convert
  [LAMBDA (object)                                     (* D Thompson "20-Jun-80 12:21")

          (* * Returns printout form of operators)


    (PROG (printForm)
          (if (NUMBERP object) or (STRINGP object)
              then (RETURN object))
          (object←(Shorten object))
          (printForm←(GETPROP CHARSET object))
          (if printForm=NIL
              then (if object MEMB #LIST\OF\SYSTEM\OPS
                       then printForm←(L-CASE object)

          (* Could store the result of the L-CASE but we need all the space we can get and there just are not that many cells
          to L-CASE.)


                  else printForm←object)
               printForm←(if DTVSCANON
                             then printForm
                           else (RenameAtoms printForm)))
          (RETURN printForm])
```

12

```
(DetermineQuantifierOrder
  [LAMBDA (givenList findList)                         (* D.Thompson " 1-Apr-80 14:35")

          (* * This routine determines the order of quantifiers of a Q-expression. using the dependency lists of the given
          variables.)


    (PROG (findSegment givenSegment outputSequence)    (* The output sequence is a list of lists of vars
                                                          alternating between given and find vars.)
          (outputSequence← <(for x in givenList unless (LISTP x) collect x)
                           >)
          (givenList←(for x in givenList when (LISTP x) collect x))
          (while givenList do (findSegment←givenList:1::1)
                              (for x in givenList::1 do findSegment←(INTERSECTION findSegment x::1))
                              (if findSegment
                                  then givenSegment←(for x in givenList collect x:1
                                                                unless (LDIFFERENCE x::1 findSegment
```

```
                    givenList←(for x in givenList collect x
                                  unless x:1 MEMB givenSegment)
                    findList←(LDIFFERENCE findList findSegment)
                    outputSequence← < ! outputSequence findSegment givenSegment>
               else                          (* oops' dependency lists of givens should never be
                                                independent.)
                    (Unexpected 'DependencyLists T)))
    (if findList
        then outputSequence← < ! outputSequence findList>)
    (RETURN outputSequence])
```

13

```
(EndLine
  [LAMBDA (INDENT)                          (* Does a TERPRI and indents on the new line.)
    (if (IGEQ #NUMBERLINES #TERMINALPAGELENGTH)
        then #NUMBERLINES←0)
    (TERPRI)
    #NUMBERLINES←#NUMBERLINES+1
    (SPACES INDENT])
```

14

```
(FuncMakeLevels
  [LAMBDA (ARGLIST NAME MAXFLAG)            (* R.Bates "30-Jul-79 15:34")
    (PROG ((DEPTH 0))

          (* Applies MakeLevels to n-ary function, leaving it n-ary. Applies MakeLevels to items in ARGLIST, then appends
          level and NAME to the result. If MAXFLAG is non NIL then level is the IMAX level of the arguments + 1, else the sum
          of the arguments + 1)


          (ARGLIST←(ArgsMakeLevels ARGLIST))
          (if MAXFLAG
              then (for X in ARGLIST do DEPTH←(Max X DEPTH))
            else (for X in ARGLIST do DEPTH←(if (LISTP X)
                                                then X:1
                                              else 1)+DEPTH))
          (DEPTH←(IMAX 2 DEPTH+1))
          (RETURN <DEPTH NAME ! ARGLIST>])
```

15

```
(GetOpLength
  [LAMBDA (OPGETLEN)                        (* Gets number of characters in the printed name of an
                                               operator.)
    (NCHARS (Convert OPGETLEN])
```

16

```
(InitHierarchy
  [LAMBDA NIL                               (* D Thompson "18-Oct-79 17:01")
                                            (* SETS the hierarchy of all STANDARD operators)
    (SetHierarchy '(ASSIGN GOTO ASRT ENTRY EXIT)
                  11)
    (SetHierarchy <IMPIO EQVIO> 10)
    (SetHierarchy '(, COMMA)
                  9)
    (SetHierarchy <ALLIO SOMEIO EQIO NEIO LTIO GTIO LEIO GEIO MAXIO MINIO 'IN > 8)
    (SetHierarchy <ADDIO DIFFIO ORIO> 6)
    (SetHierarchy <MULTIO QUOTIENTIO ANDIO DIVIO MODIO> 4)
    (SetHierarchy <INVIO 'DOT QOP> 3)
    (SetHierarchy <NOTIO NEGIO> 1])
```

17

```
(InitTTYCharSet
  [LAMBDA NIL                               (* R.Bates "26-Jun-80 19:07")
                                            (* Call after InitTTYPrintOut to initialize tty
                                               characters)

    (PUT 'TTYCHARSET IMPIO " imp ")
    (PUT 'TTYCHARSET EQVIO " eqv ")
    (PUT 'TTYCHARSET NOTIO "~")
    (PUT 'TTYCHARSET NEIO " ~= ")
    (PUT 'TTYCHARSET LEIO " <= ")
    (PUT 'TTYCHARSET GEIO " >= ")
```

```
    (PUT 'TTYCHARSET 'IN " in ")
    (PUT 'TTYCHARSET ORIO " or ")
    (PUT 'TTYCHARSET ANDIO " and ")
    (PUT 'TTYCHARSET ADDIO "+")
    (PUT 'TTYCHARSET NEGIO "-")
    (PUT 'TTYCHARSET DIFFIO "-")
    (PUT 'TTYCHARSET MULTIO "*")
    (PUT 'TTYCHARSET QUOTIENTIO "/")
    (PUT 'TTYCHARSET INVIO "1/")
    (PUT 'TTYCHARSET LTIO "<")
    (PUT 'TTYCHARSET GTIO ">")
    (PUT 'TTYCHARSET EQIO "=")
    (PUT 'TTYCHARSET DIVIO " div ")
    (PUT 'TTYCHARSET MODIO " mod ")
    (PUT 'TTYCHARSET ALLIO "all ")
    (PUT 'TTYCHARSET SOMEIO "some ")
    (PUT 'TTYCHARSET '. ". ")
    (PUT 'TTYCHARSET 'COMMA ". ")
    (PUT 'TTYCHARSET 'ASSIGN ":=")
    (PUT 'TTYCHARSET 'GOTO "go to ")
    (PUT 'TTYCHARSET 'ASRT "assert ")
    (PUT 'TTYCHARSET 'ENTRY "entry ")
    (PUT 'TTYCHARSET 'EXIT "exit ")
    (PUT 'TTYCHARSET 'DOT ".")
    (PUT 'TTYCHARSET MAXIO " max ")
    (PUT 'TTYCHARSET MINIO " min "])
```

18

```
(ListAdd1
  [LAMBDA (X)                                   (* R.Bates "30-Jul-79 15:32")
    (if (LISTP X)
        then X:1
      else 1)+ 1])
```

19

```
(ListGT
  [LAMBDA (X Y)                                 (* R.Bates "31-Jul-79 13:20")
    ((if (LISTP X)
         then X:1
       else 1)
     gt Y])
```

20

```
(ListPlus
  [LAMBDA (X Y)                                 (* R.Bates " 6-Aug-79 12:43")
    (if (LISTP X)
        then X:1
      else 1)+(if (LISTP Y)
                  then Y:1
                else 1])
```

21

```
(Magic
  [LAMBDA (ARG)                                 (* R.Bates "16-JAN-79 10:10")
                                                (* Estimates roughly what will fit on REMAINDER of line)
    ARG/MagicCount])
```

22

```
(MakeBinaryLevels
  [LAMBDA (ARGLIST NAME)                        (* R.Bates " 1-Nov-79 10:15")
                                                (* Applies MakeLevels to items in ARGLIST, binarying the
                                                items)
    (if ARGLIST=NIL
        then <2 NAME>
      else (PROG ((LEFT (MakeLevels ARGLIST:1))
                  RIGHT)
                 (if ARGLIST::1=NIL
                     then (RETURN <(ListAdd1 LEFT)
                                   NAME LEFT>)
                   elseif (ARGLIST::2=NIL) or (EQUAL ARGLIST::2 '(NIL))
                     then RIGHT←(MakeLevels ARGLIST:2)
```

```
                (RETURN <(1+(ListPlus LEFT RIGHT))
                        NAME LEFT RIGHT>)
            else RIGHT←(MakeBinaryLevels ARGLIST::1 NAME)
                 (RETURN <(1+(ListPlus LEFT RIGHT))
                         NAME LEFT RIGHT>])
```

23

```
(MakeLevels
  [LAMBDA (formula)                                    (* D.Thompson " 1-Apr-80 13:52")
    (if PFFlag
        then (NewMakeLevels formula)
      else (OldMakeLevels formula])
```

24

```
(Max
  [LAMBDA X                                            (* R.Bates "30-Jul-79 15:31")
    (for (i (SSVAL ←(MAX))
            temp)
       from 1 to X do (temp←(ARG X i))
                      (if (LISTP temp)
                          then temp←temp:1
                        else temp←1)
                      (if temp gt SSVAL
                          then SSVAL←temp])
```

25

```
(NewMakeLevels
  [LAMBDA (formula)                                    (* D.Thompson "19-Aug-80 16:50")

        (* * Preprocesses prefix programs or formulas for formula printer. Inserts level numbers into a prefix formula.
        Also inserts commas into argument lists of user functions and user procedures. *
        Method: *
        ATOMS = level 1 *
        REPEAT & COMPOUND = 1 + MAX of levels of arguments -- minimum 2 *
        ALL OTHERS = 1 + sum of levels of arguments -- minimum 2!)


    (PROG (agiven afind afree aexpr leftPart leveledExpression operatorName)
          [if formula=NIL or formula='NOP
              then leveledExpression←'(0 formula)
            elseif (NLISTP formula)
              then leveledExpression←formula
            else (SELECTQ formula:SYNTACTICTYPE
                          ((assertStatement assignmentStatement assumeStatement caseStatement
                                            compoundStatement forStatement goToStatement ifStatement
                                            labelStatement procedureStatement proveStatement
                                            repeatStatement returnStatement statement whileStatement)
                            leveledExpression←
                            (MakeProgramLevels formula))
                          (Qexpression afind←formula:find agiven←formula:given afree←NIL aexpr←
                                       (if formula:expr
                                         else '(NOP))
                                       (if agiven=NIL and afind=NIL and afree=NIL
                                           then leveledExpression←(MakeLevels aexpr)
                                         else aexpr←(MakeLevels aexpr)
                                              leveledExpression← <(if (LISTP aexpr)
                                                                      then aexpr:1
                                                                    else 2)
                                                                 'Qexpression agiven afind afree
                                                                 aexpr>))
                          (PROGN operatorName←formula:1
                                 leftPart←formula::1
                                 (if operatorName='ArrayAccess
                                     then operatorName←'ARRAY)
                                 (if operatorName='RecordAccess
                                     then leveledExpression←(MakeBinaryLevels leftPart 'DOT)
                                   elseif operatorName MEMB <IFOP ! '(IF IFELSE CONDEXP) >
                                     then leveledExpression←(FuncMakeLevels leftPart IFOP NIL)
                                   elseif (Shorten operatorName) MEMB #LIST\OF\SYSTEM\OPS
                                             and (leftPart::1=NIL or leftPart::2=NIL)
                                     then leveledExpression←(MakeBinaryLevels leftPart operatorName)
                                   elseif leftPart=NIL
                                     then leveledExpression← <2 'FUNCTION (MakeLevels operatorName) >
                                   else (if leftPart::1=NIL
                                             then leftPart←(MakeLevels leftPart:1)
```

```
                    else leftPart←(MakeBinaryLevels leftPart 'COMMA))
                  leveledExpression← <(ListAdd1 leftPart)
                                      'FUNCTION
                                      (MakeLevels operatorName)
                                      leftPart>]
    (RETURN leveledExpression])
```

26

```
(NewPrintFormula
  [LAMBDA (formula currentIndentation parentOpPrecedence printSwitch)
                                                    (* D.Thompson "22-Jun-80 11:14")

    (* * This routine pretty-prints programs or formulae -- it requires preprocessing by routine MakeLevels.)


    (* * The parameters are :
    formula -- already processed by MakeLevels .
    currentIndentation -- the number of spaces already indented on the present line .
    parentOpPrecedence -- precedence or hierarchy of the next higher level of the formula .
    printSwitch .
    = 0 -- normal .
    = 1 -- this is a left son, so parenthesize if the present operator has precedence equal to the parent operator and is
    in # NO\LEFT\ASSOC\OPS. .
    = 7 -- this is a right son, so parenthesize if the present operator has precedence equal to the parent operator and
    is in # NO\RIGHT\ASSOC\OPS. .
    = 2 -- this is a left son and the parent operator is in # NO RIGHT\ASSOC\OPS -- or this is a right son and the parent
    operator is in # NO\LEFT\ASSOC\OPS -- parenthesize if operator is of precedence equal to the parent.
    .
    = 3 -- the parent wants this formula to be printed on 1 line .
    = 5 -- top level IMP or EQV call. .
    = 6 -- connected to hypotheses or conclusions by ANDs. Stack operands vertically .
    = 8 -- top level of hypotheses or conclusions ANDed together. Begin stacking .
    = 9 -- this level should be stacked vertically if it is printed on two or more lines.)


    (PROG (A1 A2 level operator currentLineWidth)
          (if (PrintSimple formula)
              then (RETURN))
          (level←formula:1)
          (operator←formula:2)
          (A1←formula:3)
          (A2←formula:4)
          (currentLineWidth←(CHRCT))
          (if (PrintProgramPiece operator A1 A2 formula level parentOpPrecedence currentIndentation
                                 currentLineWidth)
            elseif operator='FUNCTION
              then (PrintPrefixFunction A1 A2 currentIndentation currentLineWidth)
            elseif operator=QOP
              then (PrintQexpression operator formula parentOpPrecedence currentIndentation
                                     currentLineWidth)
            elseif operator=IFOP
              then (PrintIf operator formula level parentOpPrecedence currentIndentation
                            currentLineWidth)
            elseif (EmbeddedQuantifier operator)
              then (PrintSomeAll operator A1 A2 currentIndentation currentLineWidth)
            elseif A2=NIL
              then (if operator=NOTOP and (Shorten A1:2)=EQOP
                       then (PrintFormula <A1:1 NEOP A1:3 A1:4> currentIndentation parentOpPrecedence
                                          printSwitch)
                    else (printout NIL (Convert operator))
                         (PrintFormula A1 currentIndentation+(currentLineWidth-(CHRCT))
                                       (OperatorPrecedence operator)
                                       0))
            elseif ((ILEQ level (Magic (CHRCT))) or printSwitch=3) and printSwitch~=6
                    and printSwitch~=5 and printSwitch~=8
              then (PrintMisc1Line operator A1 A2 printSwitch level parentOpPrecedence
                                   currentIndentation currentLineWidth)
            elseif operator ~MEMB '(, COMMA DOT)
              then (PrintMisc2Lines operator A1 A2 printSwitch currentIndentation currentLineWidth)
            else (PrintCommaDot operator A1 A2 currentIndentation])
```

27

```
(OldMakeLevels
  [LAMBDA (FORMULA)                                 (* R.Bates "1-Apr-80 13:52")

    (* * Preprocesses prefix programs or formulas for PrintFormula)
```

```
(PROG (LRET)

    (* Inserts level numbers into a prefix formula. Also inserts commas into argument lists of user functions and user
    procedures. *
    *
    ATOMS = level 1 *
    REPEAT & COMPOUND = 1 + IMAX of levels of arguments -- minimum 2 *
    ALL OTHERS = 1 + sum of levels of arguments -- minimum 2)


    [if FORMULA=NIL or FORMULA='NOP
        then LRET← <0 FORMULA>
      elseif (NLISTP FORMULA)
        then LRET←FORMULA
      else (SELECTQ FORMULA:SYNTACTICTYPE
                    ((assertStatement assumeStatement proveStatement)
                      LRET←
                      (MakeBinaryLevels <FORMULA:assertion> 'ASRT))
                    (assignmentStatement LRET← (MakeBinaryLevels <FORMULA:variable
                                                                  FORMULA:expression>
                                                                 'ASSIGN))
                    (caseStatement (PROG (caseMax caseElement (LEFT (FORMULA:expression))
                                                              (RIGHT (FORMULA:statement)))
                                         (caseMax←0)
                                         (caseElement←(for (caseE caseL caseM caseS)
                                                           in FORMULA:caseElementList
                                                           collect
                                                            (caseL←caseE:caseLabel)
                                                            (if caseL:.1=NIL
                                                                then caseL←(MakeLevels caseL:1)
                                                              else caseL←(MakeBinaryLevels caseL
                                                                                           'COMMA))
                                                            (caseS←(MakeLevels (if
                                                                                  caseE:statement
                                                                                  else 'NOP)))
                                                            (caseM←(IMAX 2 (ListPlus caseL caseS))
                                                              )
                                                            (caseMax←(IMAX caseMax caseM))
                                                            (<caseM 'LABEL caseL caseS>)))
                                         (if RIGHT
                                             then RIGHT←(MakeLevels RIGHT)
                                                  RIGHT← <(ListAdd1 RIGHT)
                                                           'OTHERWISE <1 'otherwise > RIGHT
                                                  caseMax←(Max caseMax RIGHT)
                                                  caseElement← < ! caseElement RIGHT>)
                                         (LEFT←(MakeLevels LEFT))
                                         (LRET← <(Max LEFT caseMax)+ 1 'CASE LEFT
                                                  ! caseElement>)))
                    [compoundStatement (PROG ((RIGHT (FORMULA:statement)))
                                             (LRET←(FuncMakeLevels RIGHT 'CMPD T]
                    (forStatement LRET← (FuncMakeLevels <FORMULA:identifier FORMULA:expression
                                                         (L-CASE FORMULA:direction:LEXEME)
                                                         FORMULA:expression#
                                                         FORMULA:statement>
                                                        'FOR NIL))
                    (goToStatement LRET← (MakeBinaryLevels <FORMULA:label> 'GOTO))
                    [ifStatement (PROG ((EXPRESSION (FORMULA:expression))
                                        (LEFT (FORMULA:statement))
                                        (RIGHT (FORMULA:statement#)))
                                       (LRET←(if RIGHT
                                                 then (FuncMakeLevels <EXPRESSION LEFT RIGHT>
                                                                      'IFELSE NIL)
                                               else (FuncMakeLevels <EXPRESSION LEFT> 'IF NIL]
                    (labelStatement LRET← (MakeBinaryLevels
                                           <FORMULA:label (if FORMULA:simpleStatement
                                                              else 'NOP)
                                            >
                                           'LABEL))
                    (procedureStatement (PROG ((LEFT (FORMULA:expression))
                                               (NAME (FORMULA:identifier)))
                                              (if LEFT=NIL
                                                  then LRET← <2 'FUNCTION (MakeLevels NAME) >
                                                else (if LEFT::1=NIL
                                                         then LEFT←(MakeLevels LEFT:1)
                                                       else LEFT←(MakeBinaryLevels LEFT 'COMMA))
                                                     LRET← <(ListAdd1 LEFT)
                                                            'FUNCTION
                                                            (MakeLevels NAME)
                                                            LEFT>)))
```

```
(Qexpression (PROG (agiven afind afree aexpr)
                   (afind←(if FORMULA:find
                              else '(NOP)))
                   (agiven←(if FORMULA:given
                               else '(NOP)))
                   (afree←'(NOP))
                   (aexpr←(if FORMULA:expr
                              else '(NOP)))
                   (if (EQUAL agiven '(NOP)) and (EQUAL afind '(NOP))
                       and (EQUAL afree '(NOP))
                     then LRET←(MakeLevels aexpr)
                          (RETURN))
                   (if agiven::1=NIL
                       then agiven←(MakeLevels agiven:1)
                     else agiven←(MakeBinaryLevels agiven 'COMMA))
                   (if afind::1=NIL
                       then afind←(MakeLevels afind:1)
                     else afind←(MakeBinaryLevels afind 'COMMA))
                   (if afree::1=NIL
                       then afree←(MakeLevels afree:1)
                     else afree←(MakeBinaryLevels afree 'COMMA))
                   (aexpr←(MakeLevels aexpr))
                   (LRET← <(IMAX (if (LISTP agiven)
                                     then agiven:1
                                   else 1)←(ListPlus afind afree)
                                 (if (LISTP aexpr)
                                     then aexpr:1
                                   else 1))
                          'Qexpression agiven afind afree aexpr>)))
(repeatStatement (PROG ((RIGHT (FORMULA:expression))
                        (LEFT (FORMULA:statement)))
                       (LEFT←(FuncMakeLevels LEFT NIL T))
                       (RIGHT←(MakeLevels RIGHT))
                       (LRET← <(Max RIGHT LEFT)+ 1 'REPEAT LEFT::2 RIGHT>)))
[returnStatement (PROG ((RIGHT (FORMULA:expression)))
                       (if RIGHT
                           then RIGHT←(MakeLevels RIGHT)
                                LRET← <(ListAdd1 RIGHT)
                                       'FUNCTION '(1 return)
                                       RIGHT>
                         else LRET←'(1 return]
(statement LRET← <0 'NOP >)
(whileStatement LRET← (MakeBinaryLevels <FORMULA:expression
                                         FORMULA:statement
                                         'WHILE))

(PROG ((NAME (Shorten FORMULA:1))
       (LEFT (FORMULA::1)))
      (if NAME='RecordAccess
          then LRET←(MakeBinaryLevels LEFT 'DOT)
               (RETURN NIL))
      (if NAME='ArrayAccess
          then NAME←'ARRAY)
      (if NAME MEMB <IFOP ! '(IF IFELSE CONDEXP) >
          then LRET←(FuncMakeLevels LEFT NAME NIL)
        elseif (MEMBER NAME #LIST\OF\SYSTEM\OPS)
               and (LEFT::1=NIL or LEFT::2=NIL)
          then LRET←(MakeBinaryLevels LEFT NAME)
        else (if LEFT=NIL
                 then LRET← <2 'FUNCTION (MakeLevels NAME) >
               else (if LEFT::1=NIL
                        then LEFT←(MakeLevels LEFT:1)
                      else LEFT←(MakeBinaryLevels LEFT 'COMMA))
                    LRET← <(ListAdd1 LEFT)
                           'FUNCTION
                           (MakeLevels NAME)
                           LEFT>]

(RETURN LRET])
```

28

```
(OldPrintFormula
  [LAMBDA (FORMULA INDENT PREC PRINTING\SWITCH)          (* D.Thompson "18-Aug-80 14:01")

    (* * Pretty prints program or formula -- requires MakeLevels preprocessing)


    (* The values of the parameters are -
    FORMULA -- already processed by MakeLevels -
    INDENT -- the number of spaces already indented on the present line -
```

```
    PREC -- precedence or hierarchy of the next higher level of the FORMULA -
    -
    PRINTING\SWITCH -
    = 0 -- normal -
    = 1 -- this is a left son. so parenthesize if the present operator is of precedence equal to the parent operator and
    is in #NO\LEFT\ASSOC\OPS. This mode is turned off when global flag FEWER PARENS is T. -
    = 7 -- this is a right son. so parenthesize if the present operator is of precedence equal to the parent operator and
    is in #NO\RIGHT\ASSOC\OPS. -
    = 2 -- this is a left son and the parent operator is in #NO\RIGHT\ASSOC\OPS -- only if FEWER\PARENS is NIL -- or this
    is a right son and the parent operator is in #NO\LEFT\ASSOC\OPS -- regardless of FEWER\PARENS -- so parenthesize if
    operator is of precedence equal to the parent. -
    = 3 -- the parent wants this FORMULA to be printed on 1 line. -
    = 5 -- top level IMP or EQV call. -
    = 6 -- connected to hypotheses or conclusions by ANDs. Stack operands vertically -
    = 8 -- top level of hypotheses or conclusions ANDed together. Begin stacking. -
    = 9 -- this level should be stacked vertically if it is printed on two or more lines.)

(if FORMULA=NIL
    then NIL
  elseif (NLISTP FORMULA)
    then (PRIN1 (Convert FORMULA))
  elseif FORMULA:1=1
    then (PRIN1 (Convert FORMULA:2))
  elseif FORMULA:2='NOP
    then NIL
  elseif DOTS\FLAG and (CHRCT) lt 10
    then (PRIN1 "...")
  else (PROG (A1 A2 LEVEL OPF DIST)
             (LEVEL←FORMULA:1)
             (OPF←FORMULA:2)
             (DIST←(CHRCT))
             (A1←FORMULA::2)
             (A2←A1::1)
             (A1←A1:1)
             (if A2
                 then A2←A2:1)
             (if OPF MEMB '(ARRAY FUNCTION)
                 then [PROG (PAREN)
                            (PrintFormula A1 INDENT 12 0)
                            (if A2
                                then (if OPF='ARRAY
                                         then PAREN←'("[" "]")
                                       else PAREN←'("(" ")")
                                     (PRIN1 (Convert PAREN:1))
                                     (PrintFormula A2 INDENT←(DIST-(CHRCT))
                                                    12 0)
                                     (PRIN1 (Convert PAREN:2]
       elseif OPF='CMPD
         then (PRIN1 "begin ")
              (for X in FORMULA::2 do (if X:2~='NOP
                                          then (PrintFormula X INDENT←(DIST-(CHRCT))
                                                             12 0)
                                               (PRIN1 ";")
                                               (EndLine INDENT)))
              (PRIN1 "end")
       elseif A2=NIL
         then (PRIN1 (Convert OPF))
              (PrintFormula A1 INDENT+(DIST-(CHRCT))
                            (GETP 'HIERARCHY OPF)
                            0)
       elseif OPF MEMB <IFOP ! '(IF IFELSE CONDEXP) >
         then (PROG (PAREN)
                    (if (ILEQ LEVEL (Magic (CHRCT))) and (ILEQ PREC 10)
                        then INDENT←INDENT+1
                             PAREN←T
                             (PRIN1 "(")
                      else PAREN←NIL)
                    (if OPF='CONDEXP
                        then (PRIN1 " if ")
                      else (PRIN1 "if "))
                    (PrintFormula A1 INDENT+(DIST-(CHRCT))
                                  12 0)
                    (if (ListGT A2 (Magic (CHRCT)
                                          - 6))
                        then (EndLine INDENT))
                    (PRIN1 " then ")
                    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                                  12 0)
                    (if (FLENGTH FORMULA) gt 4
                        then A2←FORMULA:5
```

```
                        (if (ListGT A2 (Magic (CHRCT)
                                              - 6))
                            then (EndLine INDENT))
                        (PRIN1 " else ")
                        (PrintFormula A2 INDENT+(DIST-(CHRCT))
                                      12 0))
                (if PAREN
                    then (PRIN1 "%"))))
elseif OPF MEMB <SOMEIO ALLIO>
  then (PRIN1 (Convert OPF))
       (PrintFormula A1 INDENT+(DIST-(CHRCT))
                     12 0)
       (PRIN1 " (")
       (PrintFormula A2 INDENT+(DIST-(CHRCT))
                     12 0)
       (PRIN1 ")")
elseif OPF MEMB '(LABEL OTHERWISE)
  then (PrintFormula A1 INDENT+(DIST-(CHRCT))
                     12 0)
       (if OPF='LABEL
           then (PRIN1 ": ")
         else (PRIN1 " "))
       (PrintFormula A2 INDENT-(DIST-(CHRCT))
                     12 0)
elseif OPF='WHILE
  then (PRIN1 "while ")
       (PrintFormula A1 INDENT+(DIST-(CHRCT))
                     12 0)
       (PRIN1 " do ")
       (if (ListGT A2 (Magic (CHRCT)))
           then (EndLine INDENT+2))
       (PrintFormula A2 INDENT-(DIST-(CHRCT))
                     12 0)
elseif OPF='Qexpression
  then (PROG (HIER PAREN)
             (HIER←(GETP 'HIERARCHY OPF))
             (if PREC lt HIEP
                 then PAREN←T)
             (if PAREN
                 then (PRIN1 "("))
             (if (ListGT A1 0)
                 then (PRIN1 "forall ")
                      (PrintFormula A1 INDENT+(DIST-(CHRCT))
                                    12 0))
             (A1←FORMULA:4)
             (A2←FORMULA:5)
             (FORMULA←FORMULA::5)
             (if (ListGT A1 (Magic (CHRCT)))
                 then (EndLine INDENT+2))
             (if (ListGT A1 0)
                 then (PRIN1 " find ")
                      (PrintFormula A1 INDENT+(DIST-(CHRCT))
                                    12 0))
             (if (ListGT A2 (Magic (CHRCT)))
                 then (EndLine INDENT+2))
             (if (ListGT A2 0)
                 then (PRIN1 " free ")
                      (PrintFormula A2 INDENT+(DIST-(CHRCT))
                                    12 0))
             (A1←FORMULA:1)
             (PRIN1 " : ")
             (if (ListGT A1 (Magic (CHRCT)))
                 then (EndLine INDENT+2))
             (PrintFormula A1 INDENT+(DIST-(CHRCT))
                           12 0)
             (if PAREN
                 then (PRIN1 ")")))
elseif OPF='FOR
  then (PRIN1 "for ")
       (PRIN1 (Convert A1:2))
       (PRIN1 (Convert 'ASSIGN))
       (if (ListGT A2 (Magic (CHRCT)))
           then (EndLine INDENT+2))
       (PrintFormula A2 INDENT+(DIST-(CHRCT))
                     12 0)
       FORMULA←FORMULA::4
       (PRIN1 " ")
       (PRIN1 (Convert FORMULA:1:2))
       (PRIN1 " ")
       A1←FORMULA:2
       A2←FORMULA:3
```

```
        (if (ListGT A1 (Magic (CHRCT)))
            then (EndLine INDENT+2))
        (PrintFormula A1 INDENT+(DIST-(CHRCT))
                       12 0)
        (PRIN1 " do ")
        (if (ListGT A2 (Magic (CHRCT)))
            then (EndLine INDENT+2))
        (PrintFormula A2 INDENT+(DIST-(CHRCT))
                       12 0)
elseif OPF='CASE
  then (PRIN1 "case ")
       (PrintFormula A1 INDENT+(DIST-(CHRCT))
                       12 0)
       (PRIN1 " of ")
       (for aa in FORMULA::3 do (EndLine INDENT+3)
                                (PrintFormula aa INDENT+(DIST-(CHRCT))
                                               12 0)
                                (PRIN1 ":"))
       (EndLine INDENT)
       (PRIN1 "end")
elseif OPF='REPEAT
  then (PRIN1 "repeat ")                    (* This does not work. R.B.)
       (while A1 do (if A1:1:1 gt 0
                        then (PrintFormula A1:1 INDENT+(DIST-(CHRCT))
                                             12 0)
                             (PRIN1 ":"))
                    (if A1::1
                        then (EndLine INDENT+7))
                    (A1←A1::1))
       (if (ListGT A2 (Magic (CHRCT)
                             - 7))
           then (EndLine INDENT))
       (PRIN1 "until ")
       (PrintFormula A2 INDENT+(DIST-(CHRCT))
                       12 0)
elseif OPF MEMB '(CC CO OC OO)
  then (if OPF MEMB '(CC CO)
           then (PRIN1 "[")
         else (PRIN1 "("))
       (PrintFormula A1 INDENT-(DIST-(CHRCT))
                       12 0)
       (PRIN1 ",")
       (if (ListGT A2 (Magic (CHRCT)))
           then (EndLine INDENT))
       (PrintFormula A2 INDENT+(DIST-(CHRCT))
                       12 0)
       (if OPF MEMB '(CC OC)
           then (PRIN1 "]")
         else (PRIN1 ")"))
elseif ((ILEQ LEVEL (Magic (CHRCT))) or PRINTING\SWITCH=3)
       and PRINTING\SWITCH~=6 and PRINTING\SWITCH~=5 and PRINTING\SWITCH~=8
  then                                      (* All other operators processed here if on 1 print
                                               line.)
       (PROG (HIER PAREN RFLG)
             (HIER←(GETP 'HIERARCHY OPF))
             (if PREC lt HIER
                 then PAREN←T
               elseif PREC=HIER
                      and (PRINTING\SWITCH=2 or PRINTING\SWITCH=1
                               and OPF MEMB #NO\LEFT\ASSOC\OPS
                               or PRINTING\SWITCH=7
                               and OPF MEMB #NO\RIGHT\ASSOC\OPS)
                 then PAREN←T
               else PAREN←NIL)
             (if PAREN
                 then (PRIN1 "("))
             (if FEWER\PARENS
                 then RFLG←0
               elseif OPF MEMB #NO\RIGHT\ASSOC\OPS
                 then RFLG←2
               else RFLG←1)
             (PrintFormula A1 INDENT+(DIST-(CHRCT))
                              HIER RFLG)
             (if (IGEQ LEVEL 4) and (GetOpLength OPF)=1 and OPF~='DOT
                 then (PRIN1 " ")
                      (PRIN1 (Convert OPF))
                      (PRIN1 " ")
               else (PRIN1 (Convert OPF)))
             (if OPF MEMB #NO\LEFT\ASSOC\OPS
                 then RFLG←2
               else RFLG←7)
```

```
                    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                                  HIER RFLG)
                    (if PAREN
                        then (PRIN1 ")")))
          elseif OPF ~MEMB '(. COMMA DOT)
            then                                    (* All other operators -- except comma -- processed here
                                                       if breaking on two or more lines)
                 (PROG (DIST2 DIST3)
                       (DIST2←(GetOpLength OPF))
                       (if PRINTING\SWITCH=6 or PRINTING\SWITCH=9
                           then DIST3←0
                          else DIST3←(IMAX 2 DIST2))
                       (SPACES DIST3)
                       (PrintFormula A1 INDENT+DIST3 12 (PrintingSwitchValue PRINTING\SWITCH
                                                                             OPF A1))
                       (EndLine INDENT-(IMAX 2 DIST2)+DIST3)
                       (PRIN1 (Convert OPF))
                       (if DIST2=1
                           then (PRIN1 " "))
                       (PrintFormula A2 INDENT+(DIST-(CHRCT))
                                     12
                                     (PrintingSwitchValue PRINTING\SWITCH OPF A2)))
          else                                      (* Comma broken on two lines processed here
               (PrintFormula A1 INDENT 12 0)
               (PRIN1 (Convert OPF))
               (EndLine INDENT)
               (PrintFormula A2 INDENT 12 0])
```

29

```
(OnTTY
  [LAMBDA NIL                                       (* Selects the tty characters
    CHARSET←'TTYCHARSET])
```

30

```
(PrintArray
  [LAMBDA (OPF A1 A2 INDENT DIST)                   (* D.Thompson "25-Mar-80 09:23")
    (PrintFormula A1 INDENT 12 0)
    (if A2
        then (printout NIL LeftSqBracket)
             (PrintFormula A2 INDENT-(DIST-(CHRCT))
                           12 0)
             (printout NIL RightSqBracket])
```

31

```
(PrintCase
  [LAMBDA (OPF A1 FORMULA INDENT DIST)              (* D.Thompson "20-Mar-80 18:30")
    (printout NIL "case" .)
    (PrintFormula A1 INDENT+(DIST-(CHRCT))
                  12 0)
    (printout NIL . "of" .)
    (for aa in FORMULA do (EndLine INDENT+3)
                          (PrintFormula aa INDENT+(DIST-(CHRCT))
                                        12 0)
                          (printout NIL SemiColon))
    (EndLine INDENT)
    (printout NIL "end"])
```

32

```
(PrintCmpd
  [LAMBDA (OPF FORMULA INDENT DIST)                 (* D.Thompson "20-Mar-80 18:41")
    (printout NIL "begin" .)
    (for X in FORMULA do (if X:2~='NOP
                             then (PrintFormula X INDENT+(DIST-(CHRCT))
                                                12 0)
                                  (printout NIL SemiColon)
                                  (EndLine INDENT)))
    (printout NIL "end"])
```

33

```
(PrintCommaDot
  [LAMBDA (OPF A1 A2 INDENT)                        (* D.Thompson "20-Mar-80 18:35")
```

```
    (PrintFormula A1 INDENT 12 0)
    (printout NIL (Convert OPF))
    (EndLine INDENT)
    (PrintFormula A2 INDENT 12 0])
```

34

```
(PrintFor
  [LAMBDA (OPF A1 A2 FORMULA INDENT DIST)                    (* D.Thompson "20-Mar-80 16:39")
    (printout NIL "for" . (Convert A1:2)
              (Convert 'ASSIGN))
    (if (ListGT A2 (Magic (CHRCT)))
        then (EndLine INDENT+2))
    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                  12 0)
    FORMULA←FORMULA::4
    (printout NIL . (Convert FORMULA:1:2)
              .)
    A1←FORMULA:2
    A2←FORMULA:3
    (if (ListGT A1 (Magic (CHRCT)))
        then (EndLine INDENT+2))
    (PrintFormula A1 INDENT+(DIST-(CHRCT))
                  12 0)
    (printout NIL . "do")
    (EndLine INDENT+2)
    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                  12 0])
```

35

```
(PrintFormula
  [LAMBDA (formula currentIndentation parentOpPrecedence printSwitch)
                                                    (* D.Thompson "25-Mar-80 09:32")
    (if PFFlag
        then (NewPrintFormula formula currentIndentation parentOpPrecedence printSwitch)
      else (OldPrintFormula formula currentIndentation parentOpPrecedence printSwitch])
```

36

```
(PrintIf
  [LAMBDA (operator formula level parentOpPrecedence currentIndentation currentLineWidth)
                                                    (* D.Thompson "15-Jan-81 15:37")

          (* * This routine oversees the printing of if-then-else expressions)


    (PROG (booleanTest elseBranch offset parenthesize thenBranch)
          (booleanTest←formula:3)
          (thenBranch←formula:4)
          (elseBranch←formula:5)
          (if (ILEQ level (Magic (CHRCT))) and (ILEQ parentOpPrecedence 10)
              then currentIndentation←currentIndentation+1
                   parenthesize←T
                   (printout NIL LeftParenthesis)
            else parenthesize←NIL)
          (if operator='CONDEXP
              then (printout NIL . # (printName 'if 'IfWord))
                   offset←4
            else (printName 'if 'IfWord)
                 offset←3)
          (PrintFormula booleanTest currentIndentation+(currentLineWidth-(CHRCT))
                        12 0)
          (printName currentIndentation+offset 'ThenWord)
          (PrintFormula thenBranch currentIndentation+(currentLineWidth-(CHRCT))
                        12 0)
          (if elseBranch
              then (printName currentIndentation+offset 'ElseWord)
                   (PrintFormula elseBranch currentIndentation+(currentLineWidth-(CHRCT))
                                 12 0))
          (if parenthesize
              then (printout NIL RightParenthesis])
```

37

```
(PrintInterval
  [LAMBDA (OPF A1 A2 INDENT DIST)                    (* D.Thompson "20-Mar-80 16:36")
```

```
    (if OPF MEMB '(CC CO)
        then (printout NIL LeftSqBracket)
      else (printout NIL LeftParenthesis))
    (PrintFormula A1 INDENT+(DIST-(CHRCT))
                  12 0)
    (printout NIL Period Period)
    (if (ListGT A2 (Magic (CHRCT)))
        then (EndLine INDENT))
    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                  12 0)
    (if OPF MEMB '(CC OC)
        then (printout NIL RightSqBracket)
      else (printout NIL RightParenthesis])
```

38

```
(PrintLabel
  [LAMBDA (OPF A1 A2 INDENT DIST)                    (* D.Thompson "20-Mar-80 18:21")
    (PrintFormula A1 INDENT+(DIST-(CHRCT))
                  12 0)
    (if OPF='LABEL
        then (printout NIL Colon .)
      else (printout NIL .))
    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                  12 0])
```

39

```
(PrintMisc1Line
  [LAMBDA (operator leftSubExp rightSubExp printSwitch level parentOpPrecedence currentIndentation
                  currentLineWidth)                  (* D.Thompson "20-Jun-80 11:51")

          (* * This routine oversees the printing of expressions fitting into the remaining space of the current line.
          Precedence is worried about, so that operands that are themselves expressions will be parenthesized if necessary.)


    (PROG (operatorPrecedence parenthesize subExpressionPrintSwitch)
          (operator←(Shorten operator))
          (operatorPrecedence←(OperatorPrecedence operator))
          (if parentOpPrecedence lt operatorPrecedence or (UserProfile 'FullyParenthesize T
              then parenthesize←T
            elseif parentOpPrecedence=operatorPrecedence
                   and (printSwitch=2 or printSwitch=1 and operator MEMB #NO\LEFT\ASSOC\OPS
                            or printSwitch=7 and operator MEMB #NO\RIGHT\ASSOC\OPS)
              then parenthesize←T
            else parenthesize←NIL)
          (if parenthesize
              then (printout NIL LeftParenthesis))
          (if operator MEMB #NO\RIGHT\ASSOC\OPS
              then subExpressionPrintSwitch←2
            else subExpressionPrintSwitch←1)
          (PrintFormula leftSubExp currentIndentation+(currentLineWidth-(CHRCT))
                        operatorPrecedence subExpressionPrintSwitch)
          (if (IGEQ level 4) and (GetOpLength operator)=1 and operator~='DOT
              then (printout NIL , (Convert operator)
                             ,)
            else (printout NIL (Convert operator)))
          (if operator MEMB #NO\LEFT\ASSOC\OPS
              then subExpressionPrintSwitch←2
            else subExpressionPrintSwitch←7)
          (PrintFormula rightSubExp currentIndentation+(currentLineWidth-(CHRCT))
                        operatorPrecedence subExpressionPrintSwitch)
          (if parenthesize
              then (printout NIL RightParenthesis])
```

40

```
(PrintMisc2Lines
  [LAMBDA (operator leftSubExp rightSubExp printSwitch currentIndentation currentLineWidth)
                                                     (* D.Thompson "16-Aug-80 14:02")

          (* * This routine oversees the printing of an expression that's spread over 2 (or more) lines.)


    (PROG (operatorLength localIndentation)
          (operatorLength←(GetOpLength operator))
          (if printSwitch=6 or printSwitch=9
              then localIndentation←0
```

```
        else localIndentation←(IMAX 2 operatorLength)
             (printout NIL .SP localIndentation))
       (PrintFormula leftSubExp currentIndentation+localIndentation 12
                     (PrintingSwitchValue printSwitch operator leftSubExp))
       (EndLine currentIndentation←(IMAX 2 operatorLength)+localIndentation)
       (printout NIL (Convert operator))
       (if operatorLength=1
           then (printout NIL .))
       (PrintFormula rightSubExp currentIndentation+(currentLineWidth-(CHRCT))
                     12
                     (PrintingSwitchValue printSwitch operator rightSubExp])
```

41

(PrintPrefixFunction

```
  [LAMBDA (operator parameters currentIndentation currentLineWidth)
                                                    (* D.Thompson "28-Jul-80 17:59")

        (* * This routine oversees the printing of prefix functions and their parameter lists.)


    (if operator=IH2OP
        then (printout NIL (Convert operator)
                       LeftParenthesis # (PrintFormula parameters:3 currentIndentation+(
                                                       currentLineWidth-(CHRCT))
                                                  12 0)
                       Comma . (Convert parameters:4))
             (if (FIXP parameters:4) and parameters:4~=(NodeId parameters:4)
                 then (printout NIL . LeftCurlyBracket (NodeId parameters:4)
                                RightCurlyBracket RightParenthesis)
             else (printout NIL RightParenthesis))
      else (printout NIL (Convert operator))
           (if parameters
               then (printout NIL LeftParenthesis # (PrintFormula parameters currentIndentation←(
                                                                  currentLineWidth←(CHRCT));
                                                             12 0)
                            RightParenthesis])
```

42

(PrintProgramPiece

```
  [LAMBDA (operator A1 A2 formula level parentOpPrecedence currentIndentation currentLineWidth)
                                                    (* D.Thompson "25-Mar-80 11:31")

        (* * This routine oversees the printing of pieces of a program.)


    (PROG (foundIt)
          (foundIt←T)
          (SELECTQ operator
                   (ARRAY (PrintArray operator A1 A2 currentIndentation currentLineWidth))
                   (CASE (PrintCase operator A1 formula::3 currentIndentation currentLineWidth))
                   ((CC CO OC OO)
                     (PrintInterval operator A1 A2 currentIndentation currentLineWidth))
                   (CMPD (PrintCmpd operator formula::2 currentIndentation currentLineWidth))
                   ((CONDEXP IF IFELSE)
                     (PrintIf operator formula level parentOpPrecedence currentIndentation
                              currentLineWidth))
                   (FOR (PrintFor operator A1 A2 formula currentIndentation currentLineWidth))
                   ((LABEL OTHERWISE)
                     (PrintLabel operator A1 A2 currentIndentation currentLineWidth))
                   (WHILE (PrintWhile operator A1 A2 currentIndentation currentLineWidth))
                   (REPEAT (PrintRepeat operator A1 A2 currentIndentation currentLineWidth))
                   foundIt←NIL)
          (RETURN foundIt])
```

43

(PrintQexpression

```
  [LAMBDA (operator formula parentOpPrecedence currentIndentation currentLineWidth)
                                                    (* D.Thompson "20-Jun-80 11:55")

        (* * This routine oversees the printing of Q-expressions.)


    (PROG (expression findList freeList givenList leftParens operatorPrecedence)
          (operatorPrecedence←(OperatorPrecedence operator))
          (if parentOpPrecedence lt operatorPrecedence
```

```
        then (printout NIL LeftParenthesis)
             leftParens←1
      else leftParens←0)
     (givenList←formula:3)
     (findList←formula:4)
     (freeList←formula:5)
     (expression←formula:6)
     (leftParens←leftParens+(PrintQuantifiers givenList findList (ListGT expression
                                                                        (Magic (CHRCT)))
                                                currentIndentation currentLineWidth))
     (PrintFormula expression currentIndentation+(currentLineWidth-(CHRCT))
                   12 0)
     (for i from 1 to leftParens do (printout NIL RightParenthesis])
```

44

(PrintQuantifierSequences

```
[LAMBDA (outputSequence expressionIsBig currentIndentation currentLineWidth)
                                                        (* D Thompson " 1-Apr-80 14:35")

    (* * This routine actually prints the quantifiers of a Q-expression. Order has already been determined.)


  (PROG (leftParens name segment)
        (leftParens←0)
        (if outputSequence:1≠NIL
            then outputSequence←outputSequence::1
                 name←'some
          else name←'all)
        [outputSequence←(for x in outputSequence collect (if (EQLENGTH x 1)
                                                             then (MakeLevels x:1)
                                                           else (MakeBinaryLevels x 'COMMA]
        (for s on outputSequence bind segment eachtime segment←s:1
           do (printout NIL name .)
              (PrintSequence segment currentIndentation←(currentLineWidth-(CHRCT)))
              (if s::1 and (ListGT s:2 (Magic (CHRCT)))
                  then (EndLine currentIndentation)
                       (printout NIL LeftParenthesis)
                elseif s::1≠NIL and expressionIsBig
                  then (EndLine currentIndentation←2)
                       (printout NIL LeftParenthesis)
                else (printout NIL . LeftParenthesis))
              (leftParens←leftParens-1)
              (SELECTQ name
                       (all name←'some)
                       (some name←'all)
                       (Unexpected 'BinaryValue T)))
        (RETURN leftParens])
```

45

(PrintQuantifiers

```
[LAMBDA (originalGivenList originalFindList expressionIsBig currentIndentation currentLineWidth)
                                                        (* D Thompson "20-Jun-80 15:27")

    (* * This routine prints the quantifiers of a Q-expression. It is smart about dependency lists in given lists. etc.)


  (PROG (dependencies findList givenList leftParens outputSequence)
                                                        (* normalize find list: all atoms
                                                        (ignore "dependencies"))
        (findList←(for x in originalFindList collect (firstElement x)))
                                                        (* normalize given list: each element either an atom. or
                                                        a list: (given Var. dependencies))
        [givenList←(for x in originalGivenList
                      collect (if (NLISTP x)
                                  then x
                                elseif (EQLENGTH x 1)
                                  then (firstElement x)
                                else dependencies←(for d in x::1 bind dep
                                                     eachtime dep←(firstElement d)
                                                     when dep MEMB findList collect dep)
                                     (if dependencies
                                         then <(firstElement x:1) ! dependencies>
                                       else (firstElement x:1]
                                                        (* outputSequence is list of lists.
                                                        (giveVars findVars) (giveVars findVars)
        (outputSequence←(DetermineQuantifierOrder givenList findList))
                                                        (* Print the quantifier sets and count the number of
```

```
                                                        open parens)
        (leftParens←(PrintQuantifierSequences outputSequence expressionIsBig currentIndentation
                                              currentLineWidth))
        (RETURN leftParens])
```

46

```
(PrintRepeat
  [LAMBDA (OPF A1 A2 INDENT DIST)                    (* D.Thompson "12-Mar-80 20:56")
    (PRIN1 "repeat ")                                (* This does not work. R.B.)
    (while A1 do (if A1:1:1 gt 0
                     then (PrintFormula A1:1 INDENT+(DIST-(CHRCT))
                                        12 0)
                          (PRIN1 ";"))
                 (if A1::1
                     then (EndLine INDENT+7))
                 (A1←A1::1))
    (if (ListGT A2 (Magic (CHRCT)
                          - 7))
        then (EndLine INDENT))
    (PRIN1 "until ")
    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                  12 0])
```

47

```
(PrintSequence
  [LAMBDA (formula currentIndentation)               (* D.Thompson "25-Mar-80 08:54")

        (* * This routine will print sequences of identifiers. Right now it lets the normal mechanism
        (PrintFormula) handle it.)


    (PrintFormula formula currentIndentation 12 0])
```

48

```
(PrintSimple
  [LAMBDA (formula)                                  (* D.Thompson "25-Mar-80 08:56")

        (* * This routine prints all the trivial cases of formulas -- atoms, lists of length 1, etc.)


    (if formula=NIL
        then T
      elseif (NLISTP formula)
        then (printout NIL (Convert formula))
             T
      elseif formula:1=1
        then (printout NIL (Convert formula:2))
             T
      elseif formula:2='NOP
        then T
      elseif DOTS\FLAG and (CHRCT) lt 10
        then (printout NIL Ellipses)
             T
      else NIL])
```

49

```
(PrintSomeAll
  [LAMBDA (OPF A1 A2 INDENT DIST)                    (* D.Thompson "20-Mar-80 18:10")
    (printout NIL (Convert OPF))
    (PrintFormula A1 INDENT+(DIST-(CHRCT))
                  12 0)
    (printout NIL LeftParenthesis)
    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                  12 0)
    (printout NIL RightParenthesis])
```

50

```
(PrintWhile
  [LAMBDA (OPF A1 A2 INDENT DIST)                    (* D.Thompson "20-Mar-80 18:11")
    (printout NIL "while" .)
    (PrintFormula A1 INDENT+(DIST-(CHRCT))
```

```
                12 0)
    (printout NIL , "do")
    (EndLine INDENT+2)
    (PrintFormula A2 INDENT+(DIST-(CHRCT))
                12 0])
```

51

```
(PrintingSwitchValue
  [LAMBDA (printSwitch presentOp subExp)                    (* D.Thompson "16-Aug-80 14:07")

          (* * Sets the printSwitch for recursive PrintFormula calls on two or more lines)


    (PROG (nextOp)
          (RETURN (if (LISTP subExp)
                      then presentOp←(Shorten presentOp)
                           nextOp←(Shorten subExp:2)
                           (if printSwitch=5 and nextOp=ANDIO
                              then 8
                            elseif (printSwitch=8 or printSwitch=6) and nextOp=ANDIO
                              then 6
                            elseif printSwitch=9 and presentOp=nextOp
                              then 9
                            elseif printSwitch=0 and presentOp MEMB <ANDIO ORIO ADDIO MULTIO>
                                       and presentOp=nextOp
                              then 9
                            else 0)
                  else 0])
```

52

```
(SeparateWithBlanks?
  [LAMEDA (previous current)                                (* D.Thompson "31-Mar-81 13:49")

          (* * This routine returns a Boolean value: -
          T if the two parameters previous and current are both atoms, but neither is a special character
          (and thus the two should be separated with a blank if they're printed in order) -
          NIL otherwise (and it should be OK to run them together on output, and still be able to read them in with our parser
          and PascalReadTable).)


    (if (ATOM previous) and (ATOM current) and ((1=(NCHARS current)) and (CHCON1 current) MEMB
                                                                        CurrentBreakCharacters
                                            or (1=(NCHARS previous)) and (CHCON1 previous)
                                                                        MEMB
                                                                        CurrentBreakCharacters)
      then NIL
      else T])
```

53

```
(SetHierarchy
  [LAMBDA (OPERATORS HIER)                                  (* SETS the hierarchy of a list of OPERATORS)
    (for X in OPERATORS do (PUT 'HIERARCHY X HIER])
)

(RPAQQ DeCanonicalizeFNS (DeCanonicalize LtReplace MixedToNary RemoveMinus ReorderPlus SigmaEnd
                                         SigmaInit))
(DEFINEQ
```

54

```
(DeCanonicalize
  [LAMBDA (U)                                               (* R.Bates "30-Jul-79 14:32")

          (* * Pretties up certain EQ, PLUS, NE, LE forms for printing. DeCanonicalize takes a canonicalized prefix expression
          and finds all parts having these forms -
          (equality-relation --- (PLUS --- (MINUS A) ---)) -
          (equality-relation --- (MINUS A)) -
          and moves all the MINUS A's to the left side. Also takes form -
          (PLUS (MINUS A) B (MINUS C) --- D) -
          to -
          (PLUS --- (DIFFERENCE (PLUS (MINUS A) B) C) --- D) -
          if FEWER.PARENS is T. But if FEWER.PARENS is NIL then to -
          (DIFFERENCE (PLUS B --- D) (PLUS A C ---)). In addition, if LESS.THAN.FLAG flag is T, then LtReplace will try to
          replace certain LE's with equivalent LT's!
```

```
(if CANON or (NLISTP U)
    then U
  elseif U:1 MEMB <LEOP EQOP NEOP>
    then (PROG (LEFT RIGHT (ORIGRHS (MixedToNary U:3)))
               (if (NLISTP ORIGRHS)
                   then (RETURN U)
                 elseif ORIGRHS:1=ADDOP
                   then LEFT←(SigmaInit (MixedToNary U:2))
                        (for X in ORIGRHS::1 do (if (NEG\TERM X)
                                                    then LEFT← < !! LEFT ! <(SIMP\NEG X)
                                                                >>
                                                  else RIGHT← < !! RIGHT ! <X>>))
                        (RETURN (LtReplace <U:1 (DeCanonicalize (SigmaEnd LEFT))
                                                (DeCanonicalize (SigmaEnd RIGHT))
                                                >))
                 elseif (NEG\TERM ORIGRHS)
                   then (RETURN <U:1 (DeCanonicalize (SigmaEnd < !!(SigmaInit U:2) ! <(SIMP\NEG
                                                                                       ORIGRHS)
                                                                                    >>))
                                     0>)
                 else (RETURN <U:1 !(DeCanonicalize U::1)
                                   >)))
  elseif U:1=ADDOP
    then (if FEWER\PARENS
             then (RemoveMinus U)
           else (ReorderPlus U))
  elseif U:1=MULTOP
    then (PROG (NUMDEN)
               (NUMDEN←(GETNUMDEN U))
               (if NUMDEN:EL2=1
                   then (RETURN <(DeCanonicalize U:1) !(DeCanonicalize U::1)
                                 >)
                 else (RETURN <QUOTIENTOP (DeCanonicalize NUMDEN:EL1)
                                          (DeCanonicalize NUMDEN:EL2)
                                          >)))
  else <(DeCanonicalize U:1) !(DeCanonicalize U::1)
        >])
```

55

(LtReplace

```
[LAMBDA (U)                                                    (* R.Bates "31-Oct-79 13:33")
```

(* Changes appropriate LE forms to LT for printing. LtReplace leaves a canonical prefix expression unchanged unless it is of form (LE (PLUS --- 1 ---) ---) where the 1 occurs exactly once. This form is changed to the equivalent LT form)

```
(if LESS\THAN\FLAG=NIL or (NLISTP U) or U:1~=LEOP or (NLISTP U:2) or U:2:1~=ADDOP
      or ~(ISINT U)
    then U
  else (PROG ((OLDARGS (U:2::1))
              (NEWARGS NIL)
              (FOUND 0))
             (for X in OLDARGS do (if X=1
                                      then FOUND←FOUND+1
                                    else NEWARGS← < !! NEWARGS ! <X>>))
             (RETURN (if FOUND=1
                         then <LTOP (SigmaEnd NEWARGS)
                                    U:3>
                       else U])
```

56

(MixedToNary

```
[LAMBDA (U)                                                    (* R.Bates "30-Jul-79 14:34")
```

(* * Converts mixed binary and n-ary to n-ary. Converts top level operator and its n-ary arguments, but not lower levels than that.)

```
(if (NLISTP U)
    then U
  else (PROG ((OPRU (U:1))
              ARGLIST)
             (if OPRU ~MEMB <ANDOP OROP ADDOP MULTOP MAXOP MINOP>
                 then (RETURN U))
```

```
        (U←U::1)
        (for X in U do (if (NLISTP X) or X:1~=OPRU
                           then ARGLIST← <X ! ARGLIST>
                         else ARGLIST← <OPRU !(REVERSE (MixedToNary X)::1) ! ARGLIST>)
        (RETURN <OPRU !(REVERSE ARGLIST)
                >])
```

57

```
(RemoveMinus
  [LAMBDA (U)
    (PROG (LEFT)

        (* CHANGES add of minus term into subtract for printing. RemoveMinus takes form (PLUS (MINUS A B
        (MINUS C) --- D) to (PLUS --- (DIFFERENCE (PLUS (MINUS A B) C) --- D))

        (U←(MixedToNary U))
        (LEFT←(DeCanonicalize U:2))
        (for X in U::2 do (if (NEG\TERM X)
                              then LEFT← <DIFFOP LEFT (DeCanonicalize (SIMP\NEG X))>
                            else LEFT← <ADDOP LEFT (DeCanonicalize X)
                                      >))
        (RETURN LEFT])
```

58

```
(ReorderPlus
  [LAMBDA (U)
    (PROG (LEFT RIGHT)

        (* Groups and subtracts minus terms of a sum for printing. ReorderPlus takes form (PLUS (MINUS A B
        (MINUS C) --- D) to (DIFFERENCE (PLUS B --- D) (PLUS A C ---))

        (for X in (MixedToNary U)::1 do (if (NEG\TERM X)
                                            then RIGHT← <ADDOP RIGHT (DeCanonicalize
                                                                      (SIMP\NEG X))>

                                          else LEFT← <ADDOP LEFT (DeCanonicalize X)>

        (RETURN (if LEFT=NIL
                    then <NEGOP (SigmaEnd RIGHT)>
                  elseif RIGHT=NIL
                    then (SigmaEnd LEFT)
                  else <DIFFOP (SigmaEnd LEFT)
                               (SigmaEnd RIGHT)
                       >])
```

59

```
(SigmaEnd
  [LAMBDA (U)                                    (* R Bates "14-Dec-79 14:43"
                                                 (* Ends list of items to be summed)
    (if U=NIL
        then 0
      elseif (FLENGTH U)=1
        then U:1
      else <ADDOP ! U>])
```

60

```
(SigmaInit
  [LAMBDA (U)                                    (* Starts list of items to be summed)
    (if U=0
        then NIL
      else <U>])
)
(#INIT\INFIX\PRINT)

(RPAQ MagicCount 5)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* ↑ The information for compiling and block compiling.)]

```
(RPAQQ INFIXPRINTMACROS (ListAdd1 ListGT ListPlus))
(DECLARE: EVAL@COMPILE

(PUTPROPS ListAdd1 MACRO [(X)
                         (ADD1 (COND
                                  ((LISTP X)
                                    (CAR X))
                                  (T 1])

(PUTPROPS ListGT MACRO ((X Y)
                        (IGREATERP (COND
                                     ((LISTP X)
                                       (CAR X))
                                     (T 1))
                                   Y)))

(PUTPROPS ListPlus MACRO [(X Y)
                          (IPLUS (COND
                                   ((LISTP X)
                                     (CAR X))
                                   (T 1))
                                 (COND
                                   ((LISTP Y)
                                     (CAR Y))
                                   (T 1])
)
(DECLARE: DOEVAL@COMPILE

(PUTPROPS CHRCT MACRO [X (LIST (QUOTE IDIFFERENCE)
                               (QUOTE #CURRENT\LINELENGTH)
                               (QUOTE (POSITION])
)

(RPAQQ INFIXPRINTBLOCKS ((INFIX\PRINTBLOCK ArgsMakeLevels Convert EmbeddedQuantifier EndLine
                                           FuncMakeLevels GetOpLength INFIX\PRINT INFIX\PRINT3
                                           INFIX\PPINT4 Magic MakeBinaryLevels MakeProgramLevels
                                           MakeLevels Max OperatorPrecedence PrintingSwitchValue
                                           PrintFormula PRINT\PPOGRAM NewPrintFormula PrintArray
                                           PrintCase PrintCmpd PrintCommaDot PrintFor PrintIf
                                           PrintInterval PrintLabel PrintMisc1Line PrintMisc2Lines
                                           PrintPrefixFunction PrintProgramPiece PrintQexpression
                                           PrintQuantifiers DetermineQuantifierOrder
                                           PrintQuantifierSequences PrintRepeat PrintSequence
                                           PrintSimple PrintSomeAll PrintWhile NewMakeLevels
                                           OldMakeLevels OldPrintFormula (NOLINKFNS . T)
                                           (GLOBALVARS #CURRENT\LINELENGTH #LIST\OF\SYSTEM\OPS
                                                       #NO\LEFT\ASSOC\OPS #NO\RIGHT\ASSOC\OPS
                                                       #NUMBERLINES #TERMINALPAGELENGTH CHARSET
                                                       DOTS\FLAG FEWER\PARENS DTVSCANON CONTROLKFLG
                                                       MagicCount)
                                           (ENTRIES INFIX\PRINT INFIX\PRINT3 INFIX\PRINT4
                                                    PRINT\PROGRAM Convert))
        (DeCanonicalizeBLOCK DeCanonicalize LtReplace MixedToNary RemoveMinus ReorderPlus SigmaEnd
                             SigmaInit (NOLINKFNS . T)
                             (GLOBALVARS CANON LESS\THAN\FLAG)
                             (ENTRIES DeCanonicalize))))
[DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY
(BLOCK: INFIX\PRINTBLOCK ArgsMakeLevels Convert EmbeddedQuantifier EndLine FuncMakeLevels GetOpLength
        INFIX\PRINT INFIX\PRINT3 INFIX\PRINT4 Magic MakeBinaryLevels MakeProgramLevels MakeLevels Max
        OperatorPrecedence PrintingSwitchValue PrintFormula PRINT\PROGRAM NewPrintFormula PrintArray
        PrintCase PrintCmpd PrintCommaDot PrintFor PrintIf PrintInterval PrintLabel PrintMisc1Line
        PrintMisc2Lines PrintPrefixFunction PrintProgramPiece PrintQexpression PrintQuantifiers
        DetermineQuantifierOrder PrintQuantifierSequences PrintRepeat PrintSequence PrintSimple
        PrintSomeAll PrintWhile NewMakeLevels OldMakeLevels OldPrintFormula (NOLINKFNS . T)
        (GLOBALVARS #CURRENT\LINELENGTH #LIST\OF\SYSTEM\OPS #NO\LEFT\ASSOC\OPS #NO\RIGHT\ASSOC\OPS
                    #NUMBERLINES #TERMINALPAGELENGTH CHARSET DOTS\FLAG FEWER\PARENS DTVSCANON
                    CONTROLKFLG MagicCount)
        (ENTRIES INFIX\PRINT INFIX\PRINT3 INFIX\PRINT4 PRINT\PROGRAM Convert))
(BLOCK: DeCanonicalizeBLOCK DeCanonicalize LtReplace MixedToNary RemoveMinus ReorderPlus SigmaEnd
        SigmaInit (NOLINKFNS . T)
        (GLOBALVARS CANON LESS\THAN\FLAG)
        (ENTRIES DeCanonicalize))
]
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )
```

```
(ADDTOVAR LAMA Max)
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1153 8387 (#INIT\INFIX\PRINT 1165 . 4212) (EmbeddedQuantifier 4216 . 4394) (
MakeProgramLevels 4398 . 8014) (OperatorPrecedence 8018 . 8384)) (8482 13104 (INFIX\PRINT 8494 . 9577)
 (INFIX\PRINT3 9581 . 10707) (INFIX\PRINT4 10711 . 11806) (PRINT\PROGRAM 11810 . 13101)) (13949 64491
(AbortInfixPrint 13961 . 14159) (ArgsMakeLevels 14163 . 14396) (Convert 14400 . 15243) (
DetermineQuantifierOrder 15247 . 16752) (EndLine 16756 . 17029) (FuncMakeLevels 17033 . 17815) (
GetOpLength 17819 . 18020) (InitHierarchy 18024 . 18682) (InitTTYCharSet 18686 . 20072) (ListAdd1
20076 . 20246) (ListGT 20250 . 20433) (ListPlus 20437 . 20654) (Magic 20658 . 20907) (MakeBinaryLevels
 20911 . 21705) (MakeLevels 21709 . 21926) (Max 21930 . 22270) (NewMakeLevels 22274 . 24991) (
NewPrintFormula 24995 . 28452) (OldMakeLevels 28456 . 34743) (OldPrintFormula 34747 . 45427) (OnTTY
45431 . 45564) (PrintArray 45568 . 45885) (PrintCase 45889 . 46342) (PrintCmpd 46346 . 46718) (
PrintCommaDot 46722 . 46985) (PrintFor 46989 . 47708) (PrintFormula 47712 . 48106) (PrintIf 48110 .
49509) (PrintInterval 49513 . 50131) (PrintLabel 50135 . 50467) (PrintMisc1Line 50471 . 52486) (
PrintMisc2Lines 52490 . 53622) (PrintPrefixFunction 53626 . 54705) (PrintProgramPiece 54709 . 55982) (
PrintQexpression 55986 . 57087) (PrintQuantifierSequences 57091 . 58516) (PrintQuantifiers 58520 .
60380) (PrintRepeat 60384 . 60984) (PrintSequence 60988 . 61312) (PrintSimple 61316 . 61942) (
PrintSomeAll 61946 . 62303) (PrintWhile 62307 . 62632) (PrintingSwitchValue 62636 . 63438) (
SeparateWithBlanks? 63442 . 64291) (SetHierarchy 64295 . 64488)) (64616 70324 (DeCanonicalize 64628 .
66933) (LtReplace 66937 . 67790) (MixedToNary 67794 . 68517) (RemoveMinus 68521 . 69078) (ReorderPlus
69082 . 69624) (SigmaEnd 69828 . 70146) (SigmaInit 70150 . 70321)))))
STOP
```