PRINCIPIA MATHEMATICA PROOFS
PART I SECTION A

Produced by a SNOBOL4 program.

University of Maine at Orono
Orono, Maine

```
*          DATE LAST CHANGED JULY 23, 1982
-LIST LEFT
*
*          THE DEFINED FUNCTIONS COME FIRST.
*          THE PROGRAM STARTS AT THE LABEL START
*
*
*          SUB_LINE IS THE TEXT INTO WHICH THE SUBSTITUTION IS MADE
*          SUBS IS THE STRING WITH THE SUBSTITUTIONS IN IT.
*          FILL UP THE SUBSTITUTION TABLE.
*          START BY SETTING IT TO THE IDENTITY SUBSTITUTION
F1         SENVAR = 'pqrstuv'
F1.1       SENVAR   LEN(1) . S1 =     :F(F1.2)
           SUB<S1> = S1     :(F1.1)
*          NOW GET THE INDICATED SUBSTITUTION INTO THE TABLE.
*          FIRST APPEND A COMMA AND A BLANK TO SHOW THE END
F1.2       SUBS = SUBS ', '
F1.3       SUBS    (BREAK('/') . S1) '/' (BREAK(',') . S2)
+                  ',' (SPAN(' ') . S3)    :F(F1.4)
           SUB<S1> = S2
           SUBS   S1 '/' S2 ',' S3 =     :(F1.3)
*
*          NOW WE ACTUALLY MAKE THE SUBSTITUTION AND PRINT
F1.4       SENVAR = 'pqrstuv'
           NUM = '1234567'
*          CHANGE THE SENTENTIAL VARIABLES TO NUMBERS
F1.5       SENVAR   LEN(1) . S1 =     :F(F1.7)
           NUM   LEN(1) . N1 =
F1.6       SUB_LINE   S1 = N1    :S(F1.6)F(F1.5)
*          NOW REPLACE THE NUMBERS BY THEIR SUBSTITUTIONS
F1.7       SENVAR = 'pqrstuv'
           NUM = '1234567'
F1.8       SENVAR   LEN(1) . S1 =     :F(F1END)
           NUM   LEN(1) . N1 =
F1.9       SUB_LINE   N1 = SUB<S1>    :S(F1.9)F(F1.8)
F1END      SUBSTITUTION = SUB_LINE    :(RETURN)
*
*
F2         GE(IN,0)    :F(FRETURN)
           N1 = 0
F2.1       N1 = N1 + 1
           OUTPUT =
F2END      GE(N1,IN)    :F(F2.1)S(RETURN)
*
*
F3         SPACES = 50
           PAGE =
           OUTPUT = CENTER(HEADING)
           OUTPUT =
           OUTPUT = DUPL(' ',SPACES) THS_NUM
           OUTPUT = DUPL(' ',SPACES) DATE
           SKIP(1)
F3END      PAGE_LINES = 10    :(RETURN)
*
*
F4         ARABIC = ROMAN_TO_ARABIC<IN>
F4END      NE(ARABIC,0)    :S(RETURN)F(FRETURN)
*
*
F5         ARG PRINT_ON    :S(F5END)
           ARG PRINT_OFF    :F(RETURN)
           HEADING = 'PROGRAM TERMINATION: FIRST NON-PRINTABLE PROOF: '
+                     PRINT_OFF
           PUT_PAGE()
           :(TABLE)
```

```
*
F6          ONECH = LEN(1) . CH
F6.1        STRING   ONECH =     :F(RETURN)
F6END       REVERSE = CH REVERSE    :(F6.1)
*
*
*           READ IN A STRING AND REVERSE IT.
*           PUSH SENTENTIAL VARIABLES DOWN ON A STACK.  WHEN A
*           CONNECTIVE IS ENCOUNTERED POP OFF THE CORRECT
*           NUMBER OF AGRUMENTS, CONSTRUCT THE WFF, AND PUSH IT
*           ONTO THE STACK.  WHEN WE ARE DONE RETURN A STRING WITH THE
*           ANTECEDANT AND THE CONSEQUENT SEPERATED BY '#'.
F7          STRING   POS(0) 'C' | POS(0) 'E'   :F(FRETURN)
            STACK_COUNT = 1
            STRING   POS(0) 'C' = '#'
            STRING   POS(0) 'E' = '#'
            STRING = REVERSE(STRING)
            ONECH = LEN(1) . CH
F7.1        STRING   ONECH =     :F(FRETURN)
*           IF IT IS A SENTENTIAL VARIABLE PUSH IT ONTO THE STACK.
            CH   NOTANY('#NCAKE')    :F(F7.2)
            STACK<STACK_COUNT> = CH
            STACK_COUNT = STACK_COUNT + 1   :(F7.1)
*           IF IT IS A UNARY CONNECTIVE POP, CONCATENATE, AND PUSH.
F7.2        CH   'N'   :F(F7.3)
            STACK<STACK_COUNT - 1> = STACK<STACK_COUNT - 1> CH :(F7.1)
*           IF IT IS A BINARY CONNECTIVE POP, POP, CONCAT, AND PUSH.
F7.3        CH   ANY('CAKE')    :F(F7.4)
            STACK<STACK_COUNT - 2> = STACK<STACK_COUNT - 2>
+                                    STACK<STACK_COUNT - 1>
+                                    CH
            STACK_COUNT = STACK_COUNT - 1   :(F7.1)
*           IF IT IS THE "C" OR "E" FROM THE BEGINING WE ARE DONE.
F7.4        CH   '#'   :F(FRETURN)
            IMPLICATION = REVERSE(STACK<2>) '#' REVERSE(STACK<1>)
F7END       :(RETURN)
*
*
*           CONTROLS THE SPACING OF LINE NUMBERS AND INTERNAL
*           LINE NUMBERS TO KEEP THE PROOF LEFT JUSTIFIED.
F8          SP = IN ' '
            SP   PARENS   :F(F8.2)
*           INTERNAL LINE NUMBERS.  THEY HAVE A CONSTANT WIDTH OF 7
*           IF THEY ARE USED. OTHERWISE THEY ARE NULL.
            EQ(NUM_PF_LINES,1)    :F(F8.1)
              SP =     :(RETURN)
F8.1
*           RIGHT JUSTIFY AT LINE 7
            SP   '()' =
            SP = DUPL(' ',7 - SIZE(SP)) SP   :(RETURN)
*           LINE NUMBERS
F8.2        GE(IN,10)    :S(RETURN)
F8END         SP = ' ' SP    :(RETURN)
*
*
F9          EXCESS = PAGE_WIDTH - SIZE(IN)
            GT(EXCESS,0)    :F(FRETURN)
            EQ(REMDR(EXCESS,2),0)    :S(F9.1)F(F9END)
F9.1          CENTER = DUPL(' ',EXCESS / 2) IN    :(RETURN)
F9END         CENTER = DUPL(' ',(EXCESS + 1) / 2) IN    :(RETURN)
*
*
F10         INTEGER(IN)    :F(FRETURN)
            ABS = IN
            ABS = LT(IN,0) -ABS
```

```
*
*
F11      SPACES = 50
         PAGE =
         OUTPUT = CENTER(HEADING)
         OUTPUT =
         OUTPUT = DUPL(' ',SPACES) THS_NUM   '(CONT.)'
         OUTPUT = DUPL(' ',SPACES) DATE
         SKIP(1)
F11END   PAGE_LINES = 10   :(RETURN)
```

```
-EJECT
*
*
START
                OUTPUT('OUTPUT',6,'(7X,126A1)')
                OUTPUT('PAGE',6,'(1H1,5(/),(1X,132A1))')
                &STLIMIT = 1000000
                &ERRLIMIT = 100
                &TRIM = 1
                &OUTPUT = 1
                &INPUT = 1
                &TRACE = 0
                &DUMP = 0
                &FTRACE = 0
*
*
                THEOREM = TABLE(250)
                NAME = TABLE(35)
                LINE = TABLE(35,1)
                PART = TABLE(31,1)
                SUB = TABLE(9)
                ROMAN_TO_ARABIC = TABLE(3)
                  ROMAN_TO_ARABIC<'I'> = 1
                  ROMAN_TO_ARABIC<'II'> = 2
                  ROMAN_TO_ARABIC<'III'> = 3
                STACK = TABLE(5,1)
                PF_LINE = TABLE(24,1)
*
*
                MP_MARK = POS(0) 'C'
                PM = (POS(0) '*1.01')
+                  | (POS(0) '*2.33')
+                  | (POS(0) '*3.01')
+                  | (POS(0) '*3.02')
+                  | (POS(0) '*4.01')
+                  | (POS(0) '*4.02')
+                  | (POS(0) '*4.34')
                TI = (POS(0) '*4.1')
+                  | (POS(0) '*4.11')
+                  | (POS(0) '*4.12')
+                  | (POS(0) '*4.13')
+                  | (POS(0) '*4.14')
                ASSUMP_MARK = 'ASSUMP'
                CONJ_MARK = POS(0) '*3.03'
                SYLL_MARK = (POS(0) 'Syll') | (POS(0) 'SYLL')
                SUB_MARK = '/'
                SE_MARK = POS(0) 'SE'
                DE_MARK = POS(0) 'DE' | POS(0) 'DER'
                MULTI_LINE_PF_MARK = '&'
                COMMENT_MARK = POS(0) '+'
                DIGIT = SPAN('()#*.0123456789')
                DIGIT_SP = SPAN(' ()#*.0123456789')
                PARENS = ANY('()')
*
*
                PAGE_NUM = 2
                PAGE_WIDTH = 60
                PAGE_LENGTH = 55
*
*
                DEFINE('SUBSTITUTION(SUB_LINE,SUBS)S1,S2,S3,N1','F1')
                DEFINE('SKIP(IN),N1','F2')
                DEFINE('PUT_PAGE()','F3')
                DEFINE('ARABIC(IN),N','F4')
                DEFINE('OUTPUT_CONTROL(ARG)','F5')
```

```
          DEFINE('IMPLICATION(STRING)','F7')
          DEFINE('SP(IN)','F8')
          DEFINE('CENTER(IN),EXCESS','F9')
          DEFINE('ABS(IN)','F10')
          DEFINE('CONT_PAGE()','F11')
*
*
          TRACE('MP','LABEL')
          TRACE('ASSUMP','LABEL')
          TRACE('SUB','LABEL')
          TRACE('REP','LABEL')
          TRACE('PLACE','LABEL')
          TRACE('COMP','LABEL')
          TRACE('SYLL','LABEL')
          TRACE('SE','LABEL')
*
*
*         READ IN A TITLE PAGE
          NUM_LINES = INPUT
          NUM_LINES = CONVERT(NUM_LINES,'INTEGER')
          SKIP(15)
          COUNT = 0
T1        COUNT = COUNT + 1
          TITLE_PAGE = CENTER(INPUT)
          OUTPUT = TITLE_PAGE
          GE(COUNT,NUM_LINES) :F(T1)
```

```
-EJECT
*
*
*          READ IN A HEADING FOR EACH PAGE.
*          USE IT TO DETERMINE THE SET OF DEFINITIONS TO USE.
           HEADING = CENTER(INPUT)
           HEADING   'PRINCIPIA MATHEMATICA'   :F(HEAD1)
             REP_MARK = PM    :(PRINT)
HEAD1      HEADING   'THE THEORY OF IMPLICATION'   :F(ERROR1)
             REP_MARK = TI    :(PRINT)
*
*
*         READ IN THE DATE.  THE BUILT IN DATE FUNCTION DOESN'T
*         WORK AT UMO.
PRINT     DATE = INPUT
*
*
*          READ IN PRINTER CONTROL
           TEXT = INPUT
           TEXT   BREAK(' ') . PRINT_ON SPAN(' ')
+                 REM . PRINT_OFF   :F(ERROR1)
           &OUTPUT = 0
*
*
*          READ IN THE PROOF LINE ABBREVIATION AND THE THESIS
READ
           &INPUT = 1
*          SET UP THE INITIAL CONDITIONS ON THE PROOF
*          ABBREVATION TO BE READ IN AND THE FULL PROOF
*          TO BE GENERATED.
           LINE_NUM = 0
           NUM_PF_LINES = 1
           JUST = 0
READ4      PROOF = INPUT    :F(TABLE)
           PROOF   COMMENT_MARK    :F(READ1)
           PROOF   COMMENT_MARK =
           OUTPUT = PROOF    :(READ4)
READ1      PF_LINE<1> = PROOF
           PROOF   MULTI_LINE_PF_MARK   :F(READ2)
             JUST = 7
             PROOF   BREAK(MULTI_LINE_PF_MARK) . NUM_PF_LINES
             PROOF   NUM_PF_LINES MULTI_LINE_PF_MARK =    :F(ERROR1)
             PF_LINE<1> = PROOF
             PF_LINE_NUM = 1
READ3          PF_LINE_NUM = PF_LINE_NUM + 1
               PF_LINE<PF_LINE_NUM> = INPUT
               GE(PF_LINE_NUM,NUM_PF_LINES)   :F(READ3)
READ2      THESIS = INPUT    :F(ERROR1)
*          CHECK FOR PRINTER CONTROL
           THESIS   DIGIT . THS_NUM
           OUTPUT_CONTROL(THS_NUM)
           PUT_PAGE()
*          IF &OUTPUT IS NOT POSITIVE DON'T CONSTRUCT THE PROOF
           GE(&OUTPUT,1)    :S(THM)
*          PUT THE THESIS TO BE PROVED IN THE THEOREM TABLE.
           THESIS   DIGIT_SP REM . THEOREM<THS_NUM>
*          IF THE THEOREM HAS A BLANK THERE IS A NAME
           THEOREM<THS_NUM>   BREAK(' ') . THEOREM<THS_NUM>
+                             SPAN(' ') REM . NAME<THS_NUM>
+                             :F(READ)
           NAME<THS_NUM> = '(' NAME<THS_NUM> ')'
           :(READ)
```

```
-EJECT
*
*
*          NOW ANALYZE THE PROOF LINE
*          IF THE PROOF DOESN'T CONTAIN 'AX', 'DEF', OR 'INF' IT IS A THEOREM
THM        PROOF 'AX' | 'DEF' | 'INF'    :S(AX)
           DELIM = ANY('+-;')
*          PUT THE THESIS TO BE PROVED IN THE THEOREM TABLE.
           THESIS    DIGIT . THM_NUM SPAN(' ') . B
           THESIS    DIGIT_SP REM . THEOREM<THM_NUM>
*          IF THE THEOREM HAS A BLANK THERE IS A NAME
           THEOREM<THM_NUM>   BREAK(' ') . THEOREM<THM_NUM>
+                             SPAN(' ') REM . NAME<THM_NUM>
+                             :F(THM.1)
             NAME<THM_NUM> = '(' NAME<THM_NUM> ')'
THM.1
           OUTPUT =  THM_NUM ' ' THEOREM<THM_NUM> ' ' NAME<THM_NUM>
           SKIP(1)
*
*          PRINT THE PROOF ABBREVIATION.
           OUTPUT = 'PROOF ABBREVIATION'
           PF_LINE_NUM = 0
PF1        PF_LINE_NUM = PF_LINE_NUM + 1
           OUTPUT = PF_LINE<PF_LINE_NUM>
           GE(PF_LINE_NUM,NUM_PF_LINES)    :F(PF1)
*
*          PRINT THE FULL PROOF.
           SKIP(1)
           OUTPUT = 'PROOF'
           PAGE_LINES = PAGE_LINES + NUM_PF_LINES + 2
*          GET THE PARTS OUT OF THE PROOF LINE
           PF_LINE_NUM = 0
PF2        PF_LINE_NUM = PF_LINE_NUM + 1
           PROOF = PF_LINE<PF_LINE_NUM>
           GT(PF_LINE_NUM,NUM_PF_LINES)    :S(ERROR6)
           PROOF = PROOF '+'
           PART_NUM = 0
LOOP1      PART_NUM = PART_NUM + 1
           PROOF   BREAK('+-;') . PART<PART_NUM>    :F(THS)
           PROOF   PART<PART_NUM> DELIM =     :(LOOP1)
*
*
*          IF THE PROOF CONTAINS 'AX' IT IS AN AXIOM
AX         PROOF   'AX'   :F(DEF)
           THESIS    DIGIT . AX_NUM
           THESIS    DIGIT_SP REM . THEOREM<AX_NUM>
*          IF THE AXIOM HAS A BLANK THERE IS A NAME
           THEOREM<AX_NUM>   BREAK(' ') . THEOREM<AX_NUM>
+                               SPAN(' ') REM . NAME<AX_NUM>
+                               :F(AX.1)
             NAME<AX_NUM> = '(' NAME<AX_NUM> ')'
AX.1       SKIP(2)
           OUTPUT = AX_NUM ' ' THEOREM<AX_NUM> ' Pp. ' NAME<AX_NUM>
           :(READ)
*
*
*          IF THE PROOF CONTAINS 'DEF' IT IS A DEFINITION
DEF        PROOF   'DEF'   :F(INF)
           THESIS    DIGIT . DEF_NUM
           THESIS    DIGIT_SP REM . THEOREM<DEF_NUM>
*          IF THE DEFINITION HAS A BLANK THERE IS A NAME
           THEOREM<DEF_NUM>   BREAK(' ') . THEOREM<DEF_NUM>
+                               SPAN(' ') REM . NAME<DEF_NUM>
+                               :F(DEF.1)
             NAME<DEF_NUM> = '(' NAME<DEF_NUM> ')'
```

```
                OUTPUT = DEF_NUM ' ' THEOREM<DEF_NUM> ' Df. ' NAME<DEF_NUM>
                :(READ)
        *
        *
        *       IF THE PROOF CONTAINS 'INF' JUST READ IN THE WORDS.
        *       NOTE THAT '#' IS AN END OF TEXT MARKER.
        INF     PROOF   'INF'   :F(ERROR1)
                SKIP(2)
        INF.1   THESIS   '#'    :S(INF.2)
                  OUTPUT = THESIS
                  THESIS = INPUT    :(INF.1)
        INF.2   THESIS   '#' = ' Pp.'
                OUTPUT = THESIS
                :(READ)
```

```
-EJECT
*
*
*           PARSE EACH PART AND CONSTRUCT THE LINE
*             FIRST TURN OFF INPUT FOR EFFICIENCY.  SEE GRISWOLD,
*             POAGE, AND POLONSKY SECTION 11.5.
THS         &INPUT = 0
            BIGGEST_PART_NUM = PART_NUM - 1
*           THE LINE NUMBER IS SET TO ZERO UP AT READ, SO THAT
*           IT WILL BE INCREMENTED THROUGH THE WHOLE PROOF AS
*           EACH PROOF LINE IS ANALYZED.
*           I WISH TO PRINT THE LINE NUMBERS USED IN THE PROOF
*           ABBREVIATIONS.  THEY ARE AT THE END OF EACH PROOF
*           ABBREVIATION AND THEY CONTAIN LEFT AND RIGHT
*           PARENTHESES.  THEY WILL BE CALLED INTERNAL LINE NUMBERS.
            PART_NUM = 0
            INT_LINE_NUM = '()'
PARSE       PART_NUM = PART_NUM + 1
            EQ(PART_NUM, BIGGEST_PART_NUM - 1)    :F(PARSE1)
            PART<BIGGEST_PART_NUM>   PARENS   :F(PARSE1)
                INT_LINE_NUM = PART<BIGGEST_PART_NUM>
PARSE1
*           CHECK FOR SKIP TO A NEW PAGE BECAUSE THE PROOF IS TOO LONG.
            GE(PAGE_LINES, PAGE_LENGTH)   CONT_PAGE()
            LINE_NUM = LINE_NUM + 1
            GT(PART_NUM,BIGGEST_PART_NUM)    :S(ERROR6)
*           THE PART MUST CALL FOR MODUS PONENS, SUBSTITUTION,
*           REPLACEMENT, SYLL, SUB. OF EQ,
*           OR PLACING A PREVIOUSLY PROVED THEOREM
*           ON A LINE.  THE LINE TO BE PROVED MUST BE IN THE
*           LAST PART; ALL OTHER SINGLE NUMBERS ARE PLACEMENTS
*           ON A LINE.
            PART<PART_NUM>   MP_MARK   :S(MP)
            PART<PART_NUM>   CONJ_MARK   :S(CONJ)
            PART<PART_NUM>   ASSUMP_MARK   :S(ASSUMP)
            PART<PART_NUM>   REP_MARK   :S(REP)
            PART<PART_NUM>   SYLL_MARK   :S(SYLL)
            PART<PART_NUM>   SE_MARK   :S(SE)
            PART<PART_NUM>   SUB_MARK   :S(SUB)
            PART<PART_NUM>   DE_MARK   :S(DE)
            EQ(PART_NUM,BIGGEST_PART_NUM)    :F(PLACE)S(COMP)
*
*
*           SUBSTITUTION RULE
*           GET THE THEOREM NUMBER
SUB         PART<PART_NUM>   DIGIT . THM_NUM SPAN(' ') . B   :F(ERROR2)
*           PRINT THE THEOREM AND, IF ANY, THE SUBSTITUTIONS
            PART<PART_NUM>   PARENS   :F(SUB.1)
            LINE_NUM = LINE_NUM - 1   :(SUB.2)
SUB.1       LINE<LINE_NUM> = THEOREM<THM_NUM>
            OUTPUT = DUPL(' ',JUST) SP(LINE_NUM) LINE<LINE_NUM>
+                  '   ' THM_NUM '   ' NAME<THM_NUM>
SUB.2       PART<PART_NUM>   THM_NUM B =
            OUTPUT = DUPL(' ',JUST + 6) PART<PART_NUM>
*           NOW PERFORM THE SUBSTITUTION AND PRINT THE RESULT
            LINE<LINE_NUM + 1> = SUBSTITUTION(LINE<LINE_NUM>,
+                                            PART<PART_NUM>)
            LINE_NUM = LINE_NUM + 1
            OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM> '   SUB'
            PAGE_LINES = PAGE_LINES + 3
            :(PARSE)
*
*
MP
*           MODUS PONENS
*       CPQ, THE ANTECEDANT P, AND THE CONSEQUENT Q.  THE PROOF
*       PART GIVES US THE ANTECEDANT ONLY.  THE ANTECEDANT IS
*       CONSTRUCTED FIRST, THEN WE BACK UP THROUGH THE PROOF TO
*       THE IMPLICATION, AND FINALLY PULL OUT THE CONSEQUENT.
*       FIRST CONSTRUCT THE ANTECEDANT.
        PART<PART_NUM>   'C' DIGIT . THM_NUM
*       THE IMPLICATION COULD ALREADY BE IN THE PROOF.  THIS IS
*       TRUE IFF THE THEOREM NUMBER HAS LEFT AND RIGHT PARENS.
        THM_NUM   PARENS    :F(LMP4)
*       BACK UP THROUGH THE PROOF UNTIL WE GET THE ANTECEDANT.
*       DECREMENT THE LINE NUMBER BECAUSE WE DO NOT GENERATE
*       A NEW LINE.
        LINE_NUM = LINE_NUM - 1
        ANT_NUM = LINE_NUM
LMP5    LINE<ANT_NUM> POS(0) THEOREM<THM_NUM> RPOS(0)    :S(LMP1)
        ANT_NUM = ANT_NUM - 1
        EQ(ANT_NUM,0)    :S(ERROR5) F(LMP5)
*       THE IMPLICATON IS NOT ALREADY IN THE PROOF
LMP4    LINE<LINE_NUM> = THEOREM<THM_NUM>
        ANT_NUM = LINE_NUM
        OUTPUT = DUPL(' ',JUST) SP(LINE_NUM) LINE<LINE_NUM>
+                 '    ' THM_NUM '    ' NAME<THM_NUM>
        PAGE_LINES = PAGE_LINES + 1
*       TEST FOR A SUBSTITUTION
        PART<PART_NUM> '/'    :F(LMP1)
        PART<PART_NUM>   'C' THM_NUM SPAN(' ') REM . TEMP
        OUTPUT = DUPL(' ',JUST + 6) TEMP
        LINE<LINE_NUM + 1> = SUBSTITUTION(LINE<LINE_NUM>,TEMP)
        LINE_NUM = LINE_NUM + 1
        ANT_NUM = LINE_NUM
        OUTPUT = DUPL(' ',JUST) SP(LINE_NUM) LINE<LINE_NUM>
+                 '    SUB'
        PAGE_LINES = PAGE_LINES + 2
*       NOW FIND THE IMPLICATION
LMP1    IMP_NUM = LINE_NUM
*       THE IMPLICATION CAN BE IN ONE OF THREE FORMS: CPQ, EPQ, OR EQP.
*       IN EACH CASE WE KNOW ONLY P.  WHEN IT MAY BE EQP THEN THE
*       LENGTH OF Q MUST BE COMPUTED SO THAT IT MAY BE SKIPPED.
*       Q ITSELF IS NOT KNOWN.
LMP3    LINE<IMP_NUM>    (POS(0)  ('C' | 'E')  LINE<ANT_NUM>)
+        | ('E' LEN(ABS(SIZE(LINE<IMP_NUM>) - SIZE(LINE<ANT_NUM>) - 1))
+        LINE<ANT_NUM> RPOS(0))    :S(LMP2)
        IMP_NUM = IMP_NUM - 1
        EQ(IMP_NUM,0)    :S(ERROR5) F(LMP3)
*       NOW PULL OUT THE CONSEQUENT
LMP2    LINE_NUM = LINE_NUM + 1
        LINE<LINE_NUM> = LINE<IMP_NUM>
*       CHECK THE RESULT.
*       ALLOW EITHER 'C' OR 'E' ON THE FRONT.
        LINE<LINE_NUM>   POS(0)  ('C' | 'E') LINE<ANT_NUM> =    :S(LMP6)
*       THIS WILL ALLOW P TO BE DERIVED FROM EPQ AND Q.
          LINE<LINE_NUM> = IMPLICATION(LINE<LINE_NUM>)
          LINE<LINE_NUM>   '#' LINE<ANT_NUM> =     :F(ERROR5)
LMP6    OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM>
+                 '    MP: ' IMP_NUM ', ' ANT_NUM
        PAGE_LINES = PAGE_LINES + 1
        :(PARSE)
*
*
*       REPLACEMENT RULE
*       GET THE DEFINITION NUMBER
REP     PART<PART_NUM>   DIGIT . DEF_NUM
        TEMP = THEOREM<DEF_NUM>
        OUTPUT = DUPL(' ',JUST + 6) TEMP ' Df. ' DEF_NUM
+                 '    ' NAME<DEF_NUM>
```

```
         REPLACEMENT_SPOT = 1
         SPOT =
*        TEST FOR A ROMAN NUMERAL
         PART<PART_NUM> 'I'   :F(LREP1)
           PART<PART_NUM>   BREAK('I') SPAN('I') . SPOT   :F(ERROR3)
           PART<PART_NUM>   SPOT =     :F(ERROR3)
           REPLACEMENT_SPOT = ARABIC(SPOT)    :F(ERROR3)
           PART<PART_NUM> = TRIM(PART<PART_NUM>)
*        TEST FOR A SUBSTITUTION.
LREP1    PART<PART_NUM>   SUB_MARK   :F(LREP2)
           PART<PART_NUM>   DEF_NUM SPAN(' ') =    :F(ERROR3)
           TEMP = SUBSTITUTION(TEMP,PART<PART_NUM>)
           OUTPUT = DUPL(' ',JUST + 9) PART<PART_NUM>
           OUTPUT = DUPL(' ',JUST + 6) TEMP ' Df.'
         PAGE_LINES = PAGE_LINES + 2
LREP2    TEMP    (BREAK('=') . LHS) '=' (REM . RHS)    :F(ERROR3)
         LINE<LINE_NUM> = LINE<LINE_NUM - 1>
*        ASSUME THE RIGHT SIDE OF THE DEFINITION IS TO BE REPLACED
*        BY THE LEFT SIDE.  IF NOT WE REVERSE THE RIGHT AND LEFT
*        SIDES OF THE DEFINITION AND PROCEED.
         LINE<LINE_NUM>   RHS   :S(LREP6)
         S = RHS
         RHS = LHS
         LHS = S
*        DETERMINE THE PLACE FOR THE REPLACEMENT.
*        IF REPLACEMENT_SPOT IS ONE, SNOBOL4
*        WILL DO THE JOB AUTOMATICALLY.
*        USE '#' TO MARK THE PLACES WE DON'T WANT TO TOUCH
LREP6    EQ(REPLACEMENT_SPOT,1)    :S(LREP4)
         N = 0
LREP3    N = N + 1
         LINE<LINE_NUM>   RHS = '#'    :F(ERROR3)
         EQ(N,REPLACEMENT_SPOT - 1)    :F(LREP3)
LREP4    LINE<LINE_NUM>   RHS = LHS    :F(ERROR3)
LREP5    LINE<LINE_NUM>   '#' = RHS    :S(LREP5)
         OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM>
+                 '   REP ' SPOT
         PAGE_LINES = PAGE_LINES + 1
         :(PARSE)
*
*
*        PLACE A PREVIOUSLY PROVED THEOREM ON A LINE
*        UNLESS IT IS THERE ALREADY.  IT IS IFF THE THEOREM
*        NUMBER HAS LEFT AND RIGHT PARENS.  IF THIS IS TRUE
*        WE ALSO GIVE BACK THE NEW LINE NUMBER WE GOT.
PLACE    PART<PART_NUM>   PARENS    :F(PLACE1)
         LINE_NUM = LINE_NUM - 1    :(PARSE)
PLACE1   LINE<LINE_NUM> = THEOREM<PART<PART_NUM>>    :F(ERROR4)
         OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM>
+                 '    ' PART<PART_NUM> '    '
+                 NAME<PART<PART_NUM>>
         PAGE_LINES = PAGE_LINES + 1
         :(PARSE)
*
*
*        This places an assumption on a proof line.
ASSUMP
         PART<PART_NUM> 'ASSUMP' SPAN(' ') =     :F(ERROR4)
         LINE<LINE_NUM> = PART<PART_NUM>
         OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM>
+          '   ASSUMP'
         PAGE_LINES = PAGE_LINES + 1
         :(PARSE)
*
*
```

```
*             THIS IS A BINARY RULE WHICH TAKES IN PREVIOUSLY PROVED
*             FORMULAS OF THE FORM CPQ, CQR AND GIVES AS RESULT CPR.
SYLL
*             GET OUT THE THEOREM NUMBERS
              PART<PART_NUM>    'SYLL' SPAN(' ') DIGIT . THM1
+                                SPAN(', ') DIGIT . THM2    :F(ERROR7)
*             BACK UP THROUGH THE PROOF UNTIL THE THEOREMS ARE FOUND.
              BACK_UP = LINE_NUM - 1
              MATCH = 0
SYLL1         LINE<BACK_UP>    POS(0) THEOREM<THM1> RPOS(0)    :S(SYLL2)
              LINE<BACK_UP>    POS(0) THEOREM<THM2> RPOS(0)
+                                  :S(SYLL3)F(SYLL4)
SYLL2         NUM_THM1 = BACK_UP
              MATCH = MATCH + 1    :(SYLL4)
SYLL3         NUM_THM2 = BACK_UP
              MATCH = MATCH + 1
SYLL4         BACK_UP = BACK_UP - 1
              EQ(MATCH,2)    :S(SYLL5)
              EQ(BACK_UP,0)    :F(SYLL1)
SYLL5         THM1 = IMPLICATION(LINE<NUM_THM1>)    :F(ERROR7)
              THM2 = IMPLICATION(LINE<NUM_THM2>)    :F(ERROR7)
              THM1   BREAK('#') . ANT1 '#' REM . CONS1
              THM2   BREAK('#') . ANT2 '#' REM . CONS2
              CONS1   ANT2    :F(SYLL6)
              LINE<LINE_NUM> = 'C' ANT1 CONS2    :(SYLL7)
SYLL6         ANT1   CONS2    :F(ERROR7)
              LINE<LINE_NUM> = 'C' ANT2 CONS1
SYLL7         OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM>
+                    '    SYLL: ' NUM_THM1 ', ' NUM_THM2
              PAGE_LINES = PAGE_LINES + 1
              :(PARSE)
*
*
*             SUBSTIVITY OF EQUIVALENCE
SE
              PART<PART_NUM> ':' :F(SEE)
              PART<PART_NUM>    SE_MARK SPAN(' (') BREAK(':') . S1
+                                ':' BREAK(')') . S2    :F(ERROR9)
*             GET THE THEOREM NUMBERS OUT AND PRINT THE THEOREMS.
              S1   DIGIT . THM_NUM_LHS
              S2   DIGIT . THM_NUM_RHS
              LHS = THEOREM<THM_NUM_LHS>
              RHS = THEOREM<THM_NUM_RHS>
              OUTPUT = DUPL(' ',JUST + 6) LHS ' ' THM_NUM_LHS ' : '
+                    RHS ' ' THM_NUM_RHS
              PAGE_LINES = PAGE_LINES + 1
              SUB_LHS =
              SUB_RHS =
*             TEST FOR SUBSTITUTIONS ON THE LHS.
              S1   SUB_MARK    :F(SE1)
                S1 THM_NUM_LHS SPAN(' ') =     :F(ERROR9)
                SUB_LHS = S1
                LHS = SUBSTITUTION(LHS,SUB_LHS)
*             TEST FOR SUBSTITUTIONS ON THE RHS.
SE1           S2   SUB_MARK    :F(SE2)
                S2 THM_NUM_RHS SPAN(' ') =     :F(ERROR9)
                SUB_RHS = S2
                RHS = SUBSTITUTION(RHS,SUB_RHS)
*             IF THERE ARE SUBSTITUTIONS, THEN PRINT THEM AND THEIR RESULT.
SE2           EQ(SIZE(SUB_LHS) + SIZE(SUB_RHS),0)    :S(SE3)
                OUTPUT = DUPL(' ',JUST + 9) SUB_LHS ' : ' SUB_RHS
                OUTPUT = DUPL(' ',JUST + 6) LHS ' : ' RHS
              PAGE_LINES = PAGE_LINES + 2
*             VERIFY THAT THE ANTECEDANTS AND CONSEQUENTS MATCH PROPERLY.
SE3           LHS = IMPLICATION(LHS)    :F(ERROR9)
```

```
         LHS   BREAK('#') . ANT_LHS '#' REM . CONS_LHS    :F(ERROR9)
         RHS   BREAK('#') . ANT_RHS '#' REM . CONS_RHS    :F(ERROR9)
         ANT_LHS   CONS_RHS    :F(ERROR9)
         ANT_RHS   CONS_LHS    :F(ERROR9)
*        NOW MAKE THE REPLACEMENT.
         LINE<LINE_NUM> = LINE<LINE_NUM - 1>
         LINE<LINE_NUM>   ANT_LHS = CONS_LHS    :S(SE4)
           LINE<LINE_NUM>   CONS_LHS = ANT_LHS   :F(ERROR9)
SE4      OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM> '    SE'
         PAGE_LINES = PAGE_LINES + 1
         :(PARSE)
*
*
SEE      PART<PART_NUM> SE_MARK SPAN(' (') BREAK(')') . S1    :F(ERROR11)
*        GET THE THEOREM NUMBER
         S1   DIGIT . THM_NUM
         THM = THEOREM<THM_NUM>
         OUTPUT = DUPL(' ',JUST + 6) THM '     ' THM_NUM
         PAGE_LINES = PAGE_LINES + 1
*        TEST FOR IMPLICATION
         THM POS(0) 'E'    :F(ERROR11)
*        TEST FOR SUBSTITUTIONS
         S1   SUB_MARK    :F(SEE1)
           S1   THM_NUM SPAN(' ') =      :F(ERROR11)
           OUTPUT = DUPL(' ',JUST + 9) S1
           THM = SUBSTITUTION(THM,S1)
           OUTPUT = DUPL(' ',JUST + 6) THM
         PAGE_LINES = PAGE_LINES + 2
SEE1     THM = IMPLICATION(THM)    :F(ERROR11)
         THM   BREAK('#') . S1 '#' REM . S2    :F(ERROR11)
*        MAKE THE REPLACEMENT
*        IT IS MADE IN THIS ORDER SINCE THE MOST COMMON USE IS
*        TO CHANGE NNP TO P USING EPNNP.
         LINE<LINE_NUM> = LINE<LINE_NUM - 1>
         LINE<LINE_NUM>   S2 = S1    :S(SEE2)
*        IF IT DIDN'T WORK, TRY THE OTHER WAY.
           LINE<LINE_NUM>   S1 = S2    :F(ERROR11)
SEE2     OUTPUT = SP(INT_LINE_NUM)  SP(LINE_NUM)
+                   LINE<LINE_NUM> '    SE'
         PAGE_LINES = PAGE_LINES + 1
         PAGE_LINES = PAGE_LINES + 1
         :(PARSE)
*
*
*        PM RULE CALLED "CONJ".
*        THIS IS A BINARY RULE WHICH TAKES IN PREVIOUSLY PROVED
*        FORMULAS AND RETURNS THEIR CONJUNCTION.
CONJ
*        GET OUT THE THEOREM NUMBERS
         PART<PART_NUM>    '*3.03' SPAN(' ') DIGIT . THM1
+                          SPAN(', ') DIGIT . THM2    :F(ERROR10)
*        BACK UP THROUGH THE PROOF UNTIL THE THEOREMS ARE FOUND.
         BACK_UP = LINE_NUM - 1
         MATCH = 0
CONJ1    LINE<BACK_UP>    POS(0) THEOREM<THM1> RPOS(0)    :S(CONJ2)
         LINE<BACK_UP>    POS(0) THEOREM<THM2> RPOS(0)
+                             :S(CONJ3)F(CONJ4)
CONJ2    NUM_THM1 = BACK_UP
         MATCH = MATCH + 1    :(CONJ4)
CONJ3    NUM_THM2 = BACK_UP
         MATCH = MATCH + 1
CONJ4    BACK_UP = BACK_UP - 1
         EQ(MATCH,2)    :S(CONJ5)
         EQ(BACK_UP,0)    :F(CONJ1)
CONJ5    LINE<LINE_NUM> = 'K' LINE<NUM_THM1> LINE<NUM_THM2>
```

```
+                   '    *3.03: ' NUM_THM1 ', ' NUM_THM2
                PAGE_LINES = PAGE_LINES + 1
                :(PARSE)
*
*
*               THIS RULE TAKES A THEOREM OF THE FORM EPQ AND RETURNS
*               EITHER CPQ OR CQP (FOR DER).
DE
                PART<PART_NUM>   DE_MARK   :F(ERROR12)
                PART<PART_NUM>   POS(0) 'DER'   :S(DE1)
                  LINE<LINE_NUM> = LINE<LINE_NUM - 1>
                  LINE<LINE_NUM>   POS(0)  'E' = 'C'   :F(ERROR12)
                OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM> '   DE'
                PAGE_LINES = PAGE_LINES + 1
                :(PARSE)
DE1             TEMP = IMPLICATION(LINE<LINE_NUM - 1>)
                TEMP   BREAK('#') . S1   '#'   REM . S2   :F(ERROR12)
                LINE<LINE_NUM> = 'C' S2 S1
                OUTPUT = SP(INT_LINE_NUM) SP(LINE_NUM) LINE<LINE_NUM> '   DER'
                PAGE_LINES = PAGE_LINES + 1
                :(PARSE)
*
*
*               COMPARE THE THEOREM PROVED WITH THAT WHICH WAS READ IN
COMP            THESIS   DIGIT . THM_NUM
*               THE LINE NUMBER HAD BEEN INCREMENTED IN ANTICIPATION
*               OF ANOTHER INFERENCE RULE.
                LINE_NUM = LINE_NUM - 1
*               FIRST SEE IF THIS IS THE LAST LINE OF THE PROOF
*               OR JUST OF THE PROOF ABBREVIATION.
                PART<PART_NUM>   (THM_NUM | 'PROP' | 'Prop')   :S(COMP1)
*               SAVE THE PROVED LINE FOR LATER USE.
                THEOREM<PART<PART_NUM>> = LINE<LINE_NUM>   :(PF2)
COMP1           LINE<LINE_NUM>   POS(0)  THEOREM<THM_NUM> RPOS(0)   :F(PROB)
                :(READ)
PROB            OUTPUT = DUPL('*',60)
                OUTPUT = 'THE LAST LINE OF THE PROOF AND THE THESIS'
                OUTPUT = 'WHICH WAS READ IN DO NOT AGREE!'
                OUTPUT = 'THE PROGRAM TAKES THE THESIS WHICH WAS'
                OUTPUT = 'READ IN TO BE CORRECT;'
                OUTPUT = THM_NUM ' ' THEOREM<THM_NUM>
                OUTPUT = DUPL('*',60)
                :(READ)
*
*
*               THIS SECTION PRINTS OUT THEOREMS AND DEFINITIONS READ IN .
TABLE
                THS_NUM = ' '
                DATE = ' '
                HEADING = 'SUMMARY OF THEOREMS AND DEFINITIONS'
                PUT_PAGE()
                THEOREMS = CONVERT(THEOREM,'ARRAY')
                J = 0
                I = 1
TABLE1
*               IN THE NEXT STEP FAILURE COULD MEAN THE END OF THE ARRAY.
TABLE2          THEOREMS<I,1>   POS(0)  '*'   :F(TABLE3)
                  OUTPUT = THEOREMS<I,1> DUPL(' ',8 - SIZE(THEOREMS<I,1>))
+                          THEOREMS<I,2> '   ' NAME<THEOREMS<I,1>>   :F(END)
                  J = J + 1
*                 SEE IF THE ARRAY HAS ENDED.  IF YES, FINISH.
TABLE3            THEOREMS<I,1>   :F(END)
                  I = I + 1
                  EQ(0, REMDR(J, 50))   :F(TABLE2)
                  CONT_PAGE()   :(TABLE1)
```

```
-EJECT
*          ERROR OUTPUT SECTION
ERROR1     ST_NUM = &LASTNO
           SKIP(2)
           OUTPUT = DUPL('*',60)
           OUTPUT = 'READ ERROR'
           OUTPUT = 'PROOF IS ' PROOF
           OUTPUT = 'THESIS IS ' THESIS
           OUTPUT = 'HEADING IS ' HEADING
           OUTPUT = 'ST. NUM IS ' ST_NUM
           :(READ)
ERROR2     ST_NUM = &LASTNO
           SKIP(2)
           OUTPUT = DUPL('*',60)
           OUTPUT = 'SUBSTITUTION ERROR'
           OUTPUT = 'THESIS ' THESIS
           OUTPUT = 'P/N ' PART_NUM ' '  PART<PART_NUM>
           OUTPUT = 'L/N ' LINE_NUM ' ' LINE<LINE_NUM>
           OUTPUT = 'ST. NUM IS ' ST_NUM
           :(READ)
ERROR3     ST_NUM = &LASTNO
           SKIP(2)
           OUTPUT = DUPL('*',60)
           OUTPUT = 'REPLACEMENT ERROR'
           OUTPUT = 'P/N ' PART_NUM ' ' PART<PART_NUM>
           OUTPUT = 'DEF ' DEF_NUM ' ' THEOREM<DEF_NUM>
           OUTPUT = 'TEMP ' TEMP
           OUTPUT = 'RHS ' RHS
           OUTPUT = 'LHS ' LHS
           OUTPUT = 'REPLACEMENT_SPOT ' REPLACEMENT_SPOT
           OUTPUT = 'SPOT ' SPOT
           OUTPUT = 'ST. NUM IS ' ST_NUM
           :(READ)
ERROR4     ST_NUM = &LASTNO
           SKIP(2)
           OUTPUT = DUPL('*',60)
           OUTPUT = 'PLACE ERROR'
           OUTPUT = 'L/N ' LINE_NUM
           OUTPUT = 'P/N ' PART_NUM
           OUTPUT = 'PART ' PART<PART_NUM>
           OUTPUT = 'ST. NUM IS ' ST_NUM
           :(READ)
ERROR5     ST_NUM = &LASTNO
           SKIP(2)
           OUTPUT = DUPL('*',60)
           OUTPUT = 'MODUS PONENS ERROR'
           OUTPUT = 'ST. NUM IS ' ST_NUM
           OUTPUT = 'IMP_NUM IS ' IMP_NUM
           OUTPUT = 'LINE<IMP_NUM> IS ' LINE<IMP_NUM>
           OUTPUT = 'ANT_NUM IS ' ANT_NUM
           OUTPUT = 'LINE<ANT_NUM> IS ' LINE<ANT_NUM>
           :(READ)
ERROR6     ST_NUM = &LASTNO
           SKIP(2)
           OUTPUT = DUPL('*',60)
           OUTPUT = 'PF LINE OR PART ERROR'
           OUTPUT = 'ST. NUM IS ' ST_NUM
           :(READ)
ERROR7     ST_NUM = &LASTNO
           SKIP(2)
           OUTPUT = DUPL('*',60)
           OUTPUT = 'SYLL ERROR'
           OUTPUT = 'THM1 IS ' THM1
           OUTPUT = 'THM2 IS ' THM2
           OUTPUT = 'ST. NUM IS ' ST_NUM
```

```
          OUTPUT = 'CONS1 IS ' CONS1
          OUTPUT = 'ANT2 IS ' ANT2
          OUTPUT = 'CONS2 IS ' CONS2
          :(READ)
ERROR8    ST_NUM = &LASTNO
          SKIP(2)
          OUTPUT = DUPL('*',60)
          OUTPUT = 'SE ERROR'
          OUTPUT = 'ST. NUM IS ' ST_NUM
          OUTPUT = 'LINE IS ' LINE<LINE_NUM>
          OUTPUT = 'LINE NUMBER IS ' LINE_NUM
          OUTPUT = 'LHS IS ' LHS
          OUTPUT = 'RHS IS ' RHS
          :(READ)
ERROR9    ST_NUM = &LASTNO
          SKIP(2)
          OUTPUT = DUPL('*',60)
          OUTPUT = 'SE ERROR'
          OUTPUT = 'PART IS ' PART<PART_NUM>
          OUTPUT = 'S1 IS ' S1
          OUTPUT = 'S2 IS ' S2
          OUTPUT = 'LHS IS ' LHS
          OUTPUT = 'RHS IS ' RHS
          OUTPUT = 'SUB_LHS IS ' SUB_LHS
          OUTPUT = 'SUB_RHS IS ' SUB_RHS
          OUTPUT = 'ST. NUM. IS ' ST_NUM
          :(READ)
ERROR10   ST_NUM = &LASTNO
          SKIP(2)
          OUTPUT = DUPL('*',60)
          OUTPUT = 'CONJ ERROR'
          OUTPUT = 'THM1 IS ' THM1
          OUTPUT = 'THM2 IS ' THM2
          OUTPUT = 'ST. NUM IS ' ST_NUM
          :(READ)
ERROR11   ST_NUM = &LASTNO
          SKIP(2)
          OUTPUT = DUPL('*',60)
          OUTPUT = 'SE (EQUIV.) ERROR'
          OUTPUT = 'PART IS ' PART<PART_NUM>
          OUTPUT = 'S1 IS ' S1
          OUTPUT = 'S2 IS ' S2
          OUTPUT = 'THM IS ' THM
          OUTPUT = 'ST. NUM. IS ' ST_NUM
          :(READ)
ERROR12   ST_NUM = &LASTNO
          SKIP(2)
          OUTPUT = DUPL('*',60)
          OUTPUT = 'DE OR DER ERROR'
          OUTPUT = 'PART IS ' PART<PART_NUM>
          OUTPUT = 'S1 IS ' S1
          OUTPUT = 'S2 IS ' S2
          OUTPUT = 'ST. NUM. IS ' ST_NUM
          :(READ)
END       START

NO ERRORS DETECTED IN SOURCE PROGRAM
```