```
1657                FP
1660                CEP
1661                EXPR
1662                A
1663                EXPRESSION
1664                A
1665                EXPRESSION
1666                AREXP
1667                UNSTACK
1670                STK
1671                ESTACK
1672                VAL
1673                TEMP
1674                EXITA
1675                FP
1676                CEP
1677                TP
1700                CTP
1701                EXPRESSION
1702                BOOLEXP
1703                UNSTACK
1704                STK
1705                ESTACK
1706                VAL
1707                TEMP
1710                EXITA
1711                FP
1712                CEP
1713                TP
1714                CTP
1715                GOTO
1716                DESIGEXPR
1717                DESIGEXPR
1720                NIL
1721                TECLAUSE
1722                FALSE
1723                DESIGEXPR
1724                NIL
1725                TRUE
1726                SDESIG
1727                DESIGEXPR
1730                SDESIG
1731                SDESIG
1732                DESIGEXPR
1733                SDESIG
1734                EVAL
1735                VAL
1736                TEMP
1737                TP
1740                CTP
1741                FP
1742                CEP
1743                TO
1744                LABEL
1745                GOTOLAB
1746                TP
1747                CTP
1750                FP
1751                CEP
1752                LAB
 753                TEMP
1754                LABEL
```

```
0755                    IDENT
0756                    SDESIG
0757                    SUBSEXPR
0760                    EVAL
0761                    VAL
0762                    TEMP
0763                    TP
0764                    CTP
0765                    FP
 766                    CFP
0767                    TD
0770                    SWITCHTO
0771                    GOTOSP
0772                    SW
0773                    TEMP
0774                    TP
0775                    CTP
0776                    FP
0777                    CFP
1000                    SUBS
1001                    ESTACK
1002                    AREXP
1003                    SAEXP
1004                    AREXP
1005                    NT1
1006                    TECLAUSE
1007                    TRUE
1010                    NT2
1011                    SAEXP
1012                    NT1
1013                    FALSE
1014                    AREXP
1015                    NT2
1016                    TECLAUSE
1017                    BOOLEXP
1020                    CTRRLRI
1021                    JR
1022                    BRANCH
1023                    STK
1024                    ESTACK
1025                    SAEXP
1026                    SAEXP
1027                    TERM
1030                    ADD
1031                    STK
1032                    ESTACK
1033                    SAEXP
1034                    SAEXP
1035                    TERM
1036                    SUBT
1037                    STK
1040                    ESTACK
1041                    SAEXP
1042                    TERM
1043                    SAEXP
1044                    TERM
1045                    NEGATE
1046                    STK
1047                    ESTACK
1050                    SAEXP
1051                    TERM
1052                    TERM
```

```
1053                TERM
1054                FACTOR
1055                MULTOP
1056                TERM
1057                FACTOR
1060                MULTOP
1061                MULT
1062                STK
1063                ESTACK
1064                MULT
1065                EQUALITY
1066                STK
1067                ESTACK
1070                MULTOP
1071                SATISFY
1072                STK
1073                ESTACK
1074                FACTOR
1075                FACTOR
1076                PRIMARY
1077                EXPOS
1100                STK
1101                ESTACK
1102                FACTOR
1103                PRIMARY
1104                PRIMARY
1105                AHEAD
1106                BOOLEXP
1107                ATT
1110                TRUTH
1111                TWIN
1112                ATT2
1113                SIMPBOOL
1114                ATT
1115                FALSE
1116                BOOLEXP
1117                ATT2
1120                BOOLEXP
1121                SIMPBOOL
1122                SIMPBOOL
1123                SIMPBOOL
1124                IMPLIC
1125                EQUIV
1126                STK
1127                ESTACK
1130                SIMPBOOL
1131                IMPLIC
1132                IMPLIC
1133                IMPLIC
1134                BOOLTERM
1135                IMPLIC
1136                STK
1137                ESTACK
1140                IMPLIC
1141                BOOLTERM
1142                BOOLTERM
1143                BOOLTERM
1144                BOOLFAC
1145                OR
1146                STK
1147                ESTACK
1150                BOOLTERM
```

```
1151                                    BOOLEAC
1152                                    BOOLEAC
1153                                    BOOLEAC
1154                                    BOOLSEC
1155                                    AND
1156                                    STK
1157                                    ESTACK
1160                                    BOOLEAC
1161                                    BOOLSEC
1162                                    BOOLSEC
1163                                    BOOLPRIM
1164                                    NOT
1165                                    STK
1166                                    ESTACK
1167                                    BOOLSEC
1170                                    BOOLPRIM
1171                                    BOOLPRIM
1172                                    BOOLEXPR
1173                                    BOOLPRIM
1174                                    RELATION
1175                                    RELATION
1176                                    SAEXP
1177                                    SAEXP
1200                                    RELOP
1201                                    STK
1202                                    ESTACK
1203                                    OPR
1204                                    RELOP
1205                                    BOOLPRIM
1206                                    VAR
1207                                    STACKX
1210                                    STK
1211                                    ESTACK
1212                                    VAL
1213                                    TEMP
1214                                    BOOLPRIM
1215                                    FUNCDESIG
1216                                    STACKX
1217                                    STK
1220                                    ESTACK
1221                                    VAL
1222                                    TEMP
1223                                    BOOLPRIM
1224                                    STACKX
1225                                    STK
1226                                    ESTACK
1227                                    VAL
1230                                    BOOLEAN
1231                                    LOGVAL
1232                                    PRIMARY
1233                                    VAR
1234                                    STACKX
1235                                    STK
1236                                    ESTACK
1237                                    VAL
1240                                    TEMP
1241                                    PRIMARY
1242                                    FUNCDESIG
1243                                    STACKX
1244                                    STK
1245                                    ESTACK
1246                                    VAL
```

```
1247                    TEMP
1250                    PRIMARY
1251                    STACKX
1252                    STK
1253                    FSTACK
1254                    VAL
1255                    INTEGER
1256                    UNSINTEG
1257                    PRIMARY
1260                    STACKX
1261                    STK
1262                    FSTACK
1263                    VAL
1264                    REAL
1265                    UNSREAL
1266                    VAR
1267                    FVAL
1270                    VAL
1271                    TEMP
1272                    TP
1273                    CIP
1274                    FP
1275                    CEP
1276                    TD
1277                    IDENT
1300                    VAR
1301                    SUBSVAR
1302                    SUBSVAR
1303                    STACKX
1304                    STK
1305                    FSTACK
1306                    VAL
1307                    SUBSLIST
1310                    FVAL
1311                    VAL
1312                    TEMP
1313                    TP
1314                    CIP
1315                    FP
1316                    CEP
1317                    TD
1320                    IDENT
1321                    FVSBVAR
1322                    ARRAY
1323                    TEMP
1324                    SUBS
1325                    FSTACK
1326                    SUBSLIST
1327                    SUBSLIST
1330                    SUBSEXPR
1331                    SUBSLIST
1332                    SUBSEXPR
1333                    SUBSEXPR
1334                    AREXP
1335                    TYPEDEC
1336                    SIMPNOV
1337                    OUT
1340                    TEMP
1341                    TN
1342                    TYPE
1343                    TYPELIST
1344                    TYPEDEC
```

```
1345                                    SIMPMOV
1346                                    OUT
1347                                    TEMP
1350                                    TA
1351                                    TYPE
1352                                    OWNTYPLIST
1353                                    TYPELIST
1354                                    CRVAR
1355                                    TYPE
1356                                    TEMP
1357                                    AR
1360                                    AR
1361                                    TD
1362                                    IDENT
1363                                    TYPELIST
1364                                    TYPELIST
1365                                    CRVAR
1366                                    TYPE
1367                                    TEMP
1370                                    AR
1371                                    AR
1372                                    TD
1373                                    IDENT
1374                                    OWNTYPLIST
1375                                    CR
1376                                    CRDVAR
1377                                    TYPE
1400                                    TEMP
1401                                    AR
1402                                    AR
1403                                    VAL
1404                                    CR
1405                                    TD
1406                                    IDENT
1407                                    OWNTYPLIST
1410                                    OWNTYPLIST
1411                                    CR
1412                                    CRDVAR
1413                                    TYPE
1414                                    TEMP
1415                                    AR
1416                                    AR
1417                                    VAL
1420                                    CR
1421                                    TD
1422                                    IDENT
1423                                    SWITCHDEC
1424                                    CR
1425                                    CRSW
1426                                    FP
1427                                    FP
1430                                    AR
1431                                    AR
1432                                    LIST
1433                                    SWITCHLIST
1434                                    CR
1435                                    TD
1436                                    SWITCHID
1437                                    SWITCHLIST
1440                                    SWITCHLIST
1441                                    A
1442                                    DESIGEXPR
```

```
1443                    A
1444                    SWITCHLIST
1445                    A
1446                    DESIGNDR
1447                    A
1450                    LABELDEC
1451                    LABELLIST
1452                    LABELLIST
1453                    LABELLIST
1454                    CR
1455                    CRLAH
1456                    AR
1457                    AR
1460                    FP
1461                    CFP
1462                    TP
1463                    CTP
1464                    TASTW
1465                    A1
1466                    LABEL
1467                    CR
1470                    TD
1471                    A1
1472                    LABELLIST
1473                    CR
1474                    CRLAH
1475                    AR
1476                    AR
1477                    FP
1500                    CFP
1501                    TP
1502                    CTP
1503                    TASTW
1504                    A1
1505                    LABEL
1506                    CR
1507                    TD
1510                    A1
1511                    PROCDEC
1512                    CR
1513                    CRPROC
1514                    AR
1515                    AR
1516                    FP
1517                    CFP
1520                    TYPE
1521                    TYPE
1522                    CR
1523                    TD
1524                    PROCID
1525                    CR
1526                    PROC
1527                    A
1530                    PROCHEAD
1531                    PROCBODY
1532                    EXITA
1533                    FP
1534                    CFP
1535                    TP
1536                    CTP
1537                    A
1540                    PROCDEC
```

```
1541                                    CR
1542                                    CRPROC
1543                                    AR
1544                                    AR
1545                                    FP
1546                                    CEP
1547                                    TYPE
1550                                    REAL
1551                                    CR
1552                                    TO
1553                                    PROCID
1554                                    CR
1555                                    PROC
1556                                    A
1557                                    PROCHEAD
1560                                    PROCBODY
1561                                    EXIT1
1562                                    FP
1563                                    CEP
1564                                    IP
1565                                    CIP
1566                                    A
1567                                    PROCBODY
1570                                    STMT
1571                                    PROCBODY
1572                                    CODE
1573                                    PROCHEAD
1574                                    FRMPARPART
1575                                    INSTACK
1576                                    STK
1577                                    ISTACK
1600                                    VAL
1601                                    TASVAR
1602                                    AR
1603                                    AR
1604                                    FP
1605                                    CEP
1606                                    VALUEPART
1607                                    SPECPART
1610                                    FRMPARPART
1611                                    FRMPARPART
1612                                    FRMPARLIST
1613                                    FRMPARLIST
1614                                    FRMPAR
1615                                    FRMPARLIST
1616                                    FRMPARLIST
1617                                    FRMPAR
1620                                    FRMPAR
1621                                    CRPAR
1622                                    AR
1623                                    AR
1624                                    STK
1625                                    ISTACK
1626                                    TO
1627                                    IDENT
1630                                    VALUEPART
1631                                    VALUEPART
1632                                    VALIDLIST
1633                                    VALIDLIST
1634                                    VALIDLIST
1635                                    EVAL
1636                                    VAL
```

```
1637                                    TEMP
1640                                    TP
1641                                    CIP
1642                                    FP
1643                                    CEP
1644                                    TD
1645                                    N1
1646                                    IDENT
1647                                    STVLPAR
1650                                    FP
1651                                    CEP
1652                                    VAL
1653                                    TEMP
1654                                    TD
1655                                    N1
1656                                    VALIDLIST
1657                                    FVAL
1660                                    VAL
1661                                    TEMP
1662                                    TP
1663                                    CIP
1664                                    FP
1665                                    CEP
1666                                    TD
1667                                    N1
1670                                    IDENT
1671                                    STVLPAR
1672                                    FP
1673                                    CEP
1674                                    VAL
1675                                    TEMP
1676                                    TD
1677                                    N1
1700                                    SPECPART
1701                                    SPECPART
1702                                    SIMPMOV
1703                                    OUT
1704                                    TEMP
1705                                    TA
1706                                    SPECIFIER
1707                                    SPECIDLIST
1710                                    SPECPART
1711                                    SPECPART
1712                                    SIMPMOV
1713                                    OUT
1714                                    TEMP
1715                                    TA
1716                                    SPECIFIER
1717                                    SPECIDLIST
1720                                    SPECIDLIST
1721                                    SPECIDLIST
1722                                    USESPEC
1723                                    SPEC
1724                                    TEMP
1725                                    FP
1726                                    CEP
1727                                    TD
1730                                    IDENT
1731                                    SPECIDLIST
1732                                    USESPEC
1733                                    SPEC
1734                                    TEMP
```

```
1735
1736
1737
1740
1741
1742
1743
1744
1745
1746
1747
1750
1751
1752
1753
1754
1755
1756
1757
1760
1761
1762
1763
```

```
FP
CEP
TD
TDENT
TDENT
LITERAL
PROCID
LITERAL
STRING
LITERAL
DESINTEG
LITERAL
HASHEAL
LITERAL
RELOP
LITERAL
TYPE
LITERAL
LOGVAL
LITERAL
SPECIFIER
LITERAL
```

Appendix E: SNOBOL4 Parser Listing

```
SNOBOL4 (VERSION 3.9, MAY 19, 1972)
-------

BELL TELEPHONE LABORATORIES, INCORPORATED
CDC 6000 VERSION BY PURDUE UNIVERSITY AND I. D. A.
        DEFINE(↑TRC(STR,RULE)↑)                 :(START)
TRC     SEG[N] = RULE ↑ ↑ SEG[N]
        SEG[N] BREAK(↑ ↑) = INTEGER(RULE) RULE N        :S(RETURN)
         SEG[N] BREAK(↑ ↑) = GE(RULE,NTRULE) ↑←↑ RULE ↑ ↑ $STR
+               :(RETURN)
START ATRIM = 1         ;       ATRACE = 10000
        NRULE = 210     ;   NTRULE = 200
        V = ARRAY(NRULE + 1)
        STRAT = ↑17↑ V NULL
        N = 1
TR      TRACE(.V[N],,N,↑TRC↑)
        N = LT(N,NRULE) N + 1                   :S(TR)
        TRACE(.V[NRULE + 1],,↑<STMT>↑,↑TRC↑)
*
*
*       ALGOL HALF-GRAMMAR
*
*

        PROGRAM = *UNLABBLOCK . V[1] V *UNLABCMPD . V[169]
        BLOCK = (*LABEL ↑:↑ *BLOCK) . V[2] V *UNLABBLOCK . V[3]
        UNLABBLOCK = (*BLOCKHEAD ↑;↑ *CMPDTAIL) . V[4]
        BLOCKHEAD = (↑ΞBEGINΞ↑ *DECLAR) . V[5] V (*BLOCKHEAD ↑;↑ *DECLAR)
+                   . V[6]
        DECLAR = *TYPEDEC . V[7] V *ARRAYDEC . V[8] V *PROCDEC . V[9]
+               V *SWITCHDEC . V[10] V *LABELDEC . V[11]
        CMPDSTMT = (*LABEL ↑:↑ *CMPDSTMT) . V[12] V *UNLABCMPD . V[13]
        UNLABCMPD = (↑ΞBEGINΞ↑ *CMPDTAIL) . V[14]
        CMPDTAIL = *STMT (↑ΞENDΞ↑ . V[15] V (↑;↑ *CMPDTAIL) . V[16])
        STMT = *UNCOND . V[17] V *COND . V[18] V *FORSTMT . V[19]
+               V *PSEUDO . V[17]  ;  PSEUDO = ↑<STMT>↑ . V[NRULE + 1]
        UNCOND = *BLOCK . V[20] V *CMPDSTMT . V[21] V *BASIC . V[22]
        BASIC = (*LABEL ↑:↑ *BASIC) . V[23] V *UNLBASIC . V[24]
        UNLBASIC = *PROC . V[25] V *GOTO . V[26] V *ASSIGN . V[29]
        COND = (*LABEL ↑:↑ *COND) . V[30] V *IFCLAUSE (*FORSTMT . V[31]
+              V *UNCOND ((↑ΞELSEΞ↑ *STMT) . V[33] V NULL . V[32]))
        FORSTMT = (*LABEL ↑:↑ *FORSTMT) . V[34]
+               V (*FORCLAUSE *STMT) . V[35]
        FORCLAUSE = (↑ΞFORΞ↑ *VAR ↑:=↑ *FORLIST ↑ΞDOΞ↑) . V[36]
        FORLIST = *FORLISTELM . V[38] V (*FORLIST ↑,↑ *FORLISTELM) . V[37]
        FORLISTELM = *ARFXP ((↑ΞWHILEΞ↑ *BOOLEXP) . V[40] V
+                    (↑ΞSTEPΞ↑ *ARFXP ↑ΞUNTILΞ↑ *ARFXP) . V[41] V
+                    NULL . V[39])
        ASSIGN = *LEFTPART (*ASSIGN . V[42] V *ARFXP . V[43]
+               V *BOOLEXP . V[170])
        LEFTPART = (*VAR ↑:=↑) . V[44]
        PROC = (*PROCID *ACTPARPRT) . V[45]
        FUNCDESIG = (*PROCID *ACTPARPRT) . V[46]
        ACTPARPRT = ↑(↑ (NULL . V[47] V *ACTPARLIST . V[48]) ↑)↑
        ACTPARLIST = (*ACTPARAM) . V[50] V (*ACTPARLIST ↑,↑ *ACTPARAM)
+                   . V[49]
        ACTPARAM = *STRING . V[51] V *EXPRESSION . V[52]
        EXPRESSION = *ARFXP . V[53] V *BOOLEXP . V[171]
        GOTO = (↑ΞGOTOΞ↑ *DESIGEXPR) . V[54]
        DESIGEXPR = (*IFCLAUSE *SUBSIG ↑ΞELSEΞ↑ *DESIGEXPR) . V[55]
+                  V *SUBSIG . V[56]
        SUBSIG = (↑(↑ *DESIGEXPR ↑)↑) . V[57] V *LABEL . V[58] V
+                  *SWITCH ↑[↑ *SUBSEXPR ↑]↑ . V[60]
```

```
      LABEL = *IDENT . V[89]
      ARITHEXP = *SAEXP . V[61] v (*IFCLAUSE *SAEXP ↑ΞELSEΞ↑ *ARITHEXP)
+               . V[62]
      IFCLAUSE = (↑ΞIFΞ↑ *BOOLEXP ↑ΞTHENΞ↑) . V[63]
      SAEXP = (↑+↑ *TERM) . V[66] v (↑-↑ *TERM) . V[67] v *TERM . V[68]
+              v *SAEXP ((↑+↑ *TERM) . V[64] v (↑-↑ *TERM) . V[65])
      TERM = *FACTOR . V[70] v (*TERM *MULTOP *FACTOR) . V[69]
      MULTOP = ↑*↑ . V[71] v ↑//↑ . V[73] v ↑/↑ . V[72]
      FACTOR = *PRIMARY . V[75] v (*FACTOR ≠↑≠ *PRIMARY) . V[74]
      PRIMARY = (↑(↑ *ARITHEXP ↑)↑) . V[76] v *VAR . V[94] v *FUNCDESIG
+               . V[172] v *UNSINTEG . V[95] v *UNSREAL . V[96]
      BOOLEXP = (*IFCLAUSE *SIMPBOOL ↑ΞELSEΞ↑ *BOOLEXP) . V[77]
+              v *SIMPBOOL . V[78]
      SIMPBOOL = *IMPLIC . V[80] v (*SIMPBOOL ↑ΞEQUIVΞ↑ *IMPLIC)
+               . V[79]
      IMPLIC = *BOOLTERM . V[82] v (*IMPLIC ↑ΞIMPLIESΞ↑ *BOOLTERM)
+               . V[81]
      BOOLTERM = *BOOLFAC . V[84] v (*BOOLTERM ↑∨↑ *BOOLFAC) . V[83]
      BOOLFAC = *BOOLSEC . V[86] v (*BOOLFAC ↑∧↑ *BOOLSEC) . V[85]
      BOOLSEC = (↑¬↑ *BOOLPRIM) . V[87] v *BOOLPRIM . V[88]
      BOOLPRIM = (↑(↑ *BOOLEXP ↑)↑) . V[89] v *RELATION . V[90]
+               v *VAR . V[92] v *FUNCDESIG . V[173] v *LOGVAL . V[93]
      RELATION = (*SAEXP *RELOP *SAEXP) . V[91]
      VAR = *IDENT . V[97] v *SUBSVAR . V[98]
      SUBSVAR = (*IDENT ↑[↑ *SUBSLIST ↑]↑) . V[99]
      SUBSLIST = *SUBSEXPR . V[101] v (*SUBSLIST ↑,↑ *SUBSEXPR) . V[100]
      SUBSEXPR = *ARITHEXP . V[102]
*
*     DECLARATIONS
*
      TYPEDEC = (*TYPE *TYPELIST) . V[103] v (↑ΞOWNΞ↑ *TYPE *OWNTYPLIST)
+               . V[104]
      TYPELIST = *IDENT (NULL . V[106] v (↑,↑ *TYPELIST) . V[105])
      OWNTYPLIST = *IDENT (NULL . V[108] v (↑,↑ *OWNTYPLIST) . V[107])
      SWITCHDEC = (↑ΞSWITCHΞ↑ *SWITCHID ↑:=↑ *SWITCHLIST) . V[109]
      SWITCHLIST = *DESIGEXPR . V[111] v (*SWITCHLIST ↑,↑ *DESIGEXPR)
+               . V[110]
      LABELDEC = (↑ΞLABELΞ↑ *LABELLIST) . V[112]
      LABELLIST = *LABEL . V[114] v (*LABELLIST ↑,↑ *LABEL) . V[113]
      PROCDEC = (*TYPE ↑ΞPROCEDUREΞ↑ *PROCID *PROCHEAD *PROCBODY)
+               . V[115] v (↑ΞPROCEDUREΞ↑ *PROCID *PROCHEAD *PROCBODY)
+               . V[174]
      PROCBODY = *STMT . V[116] v *CODE . V[117]
      PROCHEAD = (*FRMPARPART ↑;↑ *VALUEPART *SPECPART) . V[118]
      FRMPARPART = (↑(↑ *FRMPARLIST ↑)↑) . V[120] v NULL . V[119]
      FRMPARLIST = *FRMPAR . V[122] v (*FRMPARLIST ↑,↑ *FRMPAR)
+               . V[121]
      FRMPAR = *IDENT . V[123]
      VALUEPART = (↑ΞVALUEΞ↑ *VALIDLIST ↑;↑) . V[125] v NULL . V[124]
      VALIDLIST = *IDENT . V[127] v (*VALIDLIST ↑,↑ *IDENT) . V[126]
      SPECPART = (*SPECIFIER *SPECIDLIST ↑;↑) . V[129] v NULL . V[128]
+              v (*SPECPART *SPECIFIER *SPECIDLIST ↑;↑) . V[130]
      SPECIDLIST = *IDENT . V[132] v (*SPECIDLIST ↑,↑ *IDENT) . V[131]
*
*     IN THIS SECTION, THE ARRAY CODING SHOULD BE INSERTED.
*
      ARRAYDEC = ↑ΞARRAYΞ↑
*
*     TERMINAL RULES
*
      LETTER = ↑ABCDEFGHIJKLMNOPQRSTUVWXYZ↑
      NUMBER = ↑0123456789↑
```

```
        IDENT = (ANY(LETTER) (SPAN(LETTER NUMBER) v NULL)) . V[200]
        PROCID = IDENT . V[201]
        STRING = ↑Ξ↑ SPAN(LETTER NUMBER) . V[202] ↑Ξ↑
        UNSINTEG = SPAN(NUMBER) . V[203]
        UNSREAL = (SPAN(NUMBER) ↑.↑ SPAN(NUMBER)) . V[204]
        RELOP = (↑<↑ v ↑>↑ v ↑≠↑ v ↑=↑ v ↑≤↑ v ↑≥↑) . V[205]
        TYPE = ↑Ξ↑ (↑INTEGER↑ v ↑REAL↑ v ↑BOOLEAN↑ v ↑STRING↑) . V[206]
+               ↑Ξ↑
        LOGVAL = ↑Ξ↑ (↑TRUE↑ v ↑FALSE↑) . V[207] ↑Ξ↑
        SPECIFIER = ↑Ξ↑ (↑INTEGER↑ v ↑REAL↑ v ↑BOOLEAN↑ v ↑STRING↑ v
+                   ↑ARRAY↑) . V[208] ↑Ξ↑
*
*
*
*
*
*
INI     OUTPUT = ↑TIME:  ↑ TIME()
IN      LINE = INPUT                    :F(END)
        IDENT(LINE)                     :S(MATCH)
        OUTPUT = LINE
L       LINE SPAN(↑ ↑) =                :S(L)
        TEXT = TEXT LINE                  :(IN)
MATC-   OUTPUT =      ;   OUTPUT =
        I = 1;     N = 0;     S = ARRAY(20)  ;  SEQ = ARRAY(20)
        BK = ≠↓≠ v ≠#≠ v NULL
BEG     TEXT ↑ΞBEGINΞ↑ = ≠↓≠            :S(BEG)
EN      TEXT ↑ΞENDΞ↑ = ≠#≠              :S(EN)
*
T       TEXT (BK BREAK(≠↓#≠)) . TEMP =          :F(RECOV)
        TEXT POS(0) ≠#≠ =               :F(BG)
        S[N] = S[N] TEMP ↑#↑
        N = N - 1                       :(T)
BG      S[N] = S[N] TEMP ↑<STMT>↑
        N = N + 1           :(T)
*
RECOV   I = DIFFER(S[N + 1]) N + 1      :F(MAT)
REC1    S[N] ≠↓≠ = ↑ΞBEGINΞ↑            :S(REC1)
REC2    S[N] ≠#≠ = ↑ΞENDΞ↑                      :S(REC2)F(RECOV)
MAT     N = 1
OUT     OUTPUT = ↑S[↑ N ↑] = ↑ S[N]
        I = LT(N,M) N + 1               :S(OUT)
        N = 1
        OUTPUT =     ;  OUTPUT =
        SEQ[1] PROGRAM
        OUTPUT = SEQ[1]
INCR    N = LT(N,M) N + 1               :F(ZERO)
RMAT    S[N] LPCONC =                    :F(INCR)
        SEQ[1] STRAT ↑ <STMT>↑ N - 1 ↑ ↑ = SEQ[1]
        OUTPUT =  ;  OUTPUT = ↑****LEVEL ↑ N ↑: ↑ SEQ[N] ; OUTPUT =
        SEQ[N] =          :(RMAT)
ZERO    TEXT =  ;   OUTPUT = SEQ[1]  ; OUTPUT =   ;   OUTPUT =   :(INI)
END
```

0 ERRORS DETECTED IN SOURCE PROGRAM

Appendix F: Specifier and Generator Listings

```
1                PROGRAM TEST(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
          C      DEBUG PARAMETER IS READ IN AS FIRST CARD.  BITS ARE SET AS FOLLOWS:
          C         1 = FEP TABLE
          C         2 = RESERVED WORD TABLE
          C         4 = TRACE SYNTACTIC STACK (INITIALLY)
          C        10 = TRACE SEMANTIC STACK (INITIALLY)
          C        20 = TRACE SEMANTIC STACK DURING INTERPRETATION
          C        40 = TRACE OF PACKED TABLE OF EXEC CALLS AND LITERAL TABLE
          C       100 = TRACE EXEC-SEM-S CALLED DURING INTERPRETATION
          C       200 = TRACE OF GRAPH AS RULES ARE EXECUTED
          C       400 = TRACE OF FULL GRAPH AT END OF GENERATION
          C      1000 = ROUTINE TIMINGS
          C      2000 = LIST RIGHT-SIDE GRAMMAR
          C      4000 = TRACE SEQUENCE OF RIGHT-SIDE RULES USED
          C
000002                 COMMON/OSFLAG/IOFLAG
000002                 COMMON IBUF(120)
000002                 COMMON/UNI/UNIV,NONTERM,IG
000002                 COMMON/MATCH/LITRUL(1200),NEXEC(500),NSTART(250),NLPTR,NRPTR,NFULL
000002                 COMMON/NTERM/NT(11)
000002                 COMMON/RDRS/RDR,RDR2
000002                 DATA NT/7HNONTERM,3HNT1,3HNT2,3HNT3,3HNT4,3HNT5,3HNT6,3HNT7,3HNT8,
000002                * 3HNT9,4HNT10/
000002                 DIMENSION ARRAY(6000)
          C
000002                 CALL SECOND(T0)
          C                    READ I/O FLAG FIRST
000004                 READ 1020,IOFLAG
000012           1020  FORMAT(O4)
          C                    SET UP RESERVED WORD TABLE
000012                 X=SETRW(X)
          C                    CONVERT FEP-S TO USABLE ARRAYS
000015                 CALL FEPLOAD
          C                    CREATE UNIVERSE GRAPH: PLACE <PROGRAM> ON IT.
000016               1 X=SETUP(ARRAY,6000,0.15,400)  $  NONTERM=INTGER(ATOM(7HNONTERM,1))
000027                 CALL SAVAR(RDR,2)
000031                 RDR=CREEDR(GRSET(X))
000035                 RDR2=CREEDR(GRSET(X))
          C                    SET UP NONTERMINAL LABELS
000042                 DO 90 I=1,11
000043              90 NT(I)=INTGER(ATOM(NT(I),1))
000053                 UNIV=CREGR(ATOM(4HUNIV,1))
000057                 PROG=CRNODE(NONTERM,UNIV)
000062                 X=HANG(UNIV,PROG)  $  X=HANG(PROG,ATOM(7HPROGRAM,1))
000073                 IG=0
000074                 IF((IOFLAG.AND.512).EQ.0) GO TO 10  $  CALL SECOND(T1)
000100                 T=T1-T0  $  PRINT 2000,T
000110           2000  FORMAT(10X,*TIME FOR INITIAL SET-UP =*,F10.5)
          C                    READ IN AND SET-UP PARSE OF GRAPH (RIGHT-SIDE) RULES
000110              10 CALL PARSER
          C                    PRINT PACKED TABLE OF EXEC-SEM DATA IF DESIRED
000111                 IF(IOFLAG.AND.32) 40,50
000113              40 I=0  $  PRINT 2020
000120           2020  FORMAT(1H1,*PACKED SEQUENCES OF EXEC-SEM CALLS, LITERAL RULES, AND
                      * RIGHT-SIDE POINTER TABLES*,//,*    N *,20X,*POINTERS*,20X,*EXEC CA
                      *LLS*,15X,*LITERAL RULES*,/)
000120              60 I=I+1  $  PRINT 2030,I
000127           2030  FORMAT(1X,O4)
000127                 IF(NSTART(I).NE.0.AND.I.LE.250) PRINT 2040,NSTART(I)
000145           2040  FORMAT(1H+,20X,O20)
000145                 IF(NEXEC(I).NE.0.AND.I.LE.400) PRINT 2050,NEXEC(I)
```

```
000163     2050 FORMAT(1H+,43X,O20)
000163          IF(LITRUL(I).EQ.0) GO TO 70
000165          IPT=IMAGE(LITRUL(I))  $  IF(IPT.GT.1000) GO TO 80
000172          ENCODE(10,2070,IPT) IPT
000201     2070 FORMAT(I10)
000201          IPT=LSHIFT(IPT,42)
000204       80 PRINT 2060,IPT
000212     2060 FORMAT(1H+,73X,A10)
000212       70 IF(I.LT.NLPTR.OR.I.LT.NRPTR) GO TO 60
000222       50 IF((IOFLAG.AND.512).EQ.0) GO TO 20  $  CALL SECOND(T2)
000226          T=T2-T1  $  PRINT 2010,T
000236     2010 FORMAT(10X,*TIME TO INITIALLY PARSE GRAPH RULES =*,F10.5)
           C         CREATE GRAPH USING SNOBOL LEFT-SIDE PARSE RULES AND THE EXEC-SEM-S
000236       20 CALL CRGRAPH
000237          IF((IOFLAG.AND.512).EQ.0) GO TO 30
000241          CALL SECOND(T1)  $  T=T1-T2  $  PRINT 2080,T
000253     2080 FORMAT(10X,*TIME TO GENERATE GRAPH =*,F10.5)
           C         TRACE FINAL GRAPH IF DESIRED
000253       30 IF(IOFLAG.AND.256) CALL TRGRAPH(UNIV,1000,1)
000260          CALL SECOND(T1)
000262          T=T1-T0
000264          PRINT 1000,T
000272     1000 FORMAT(* EXECUTION TIME =*,F10.5)
000272          END)
006100
1                LOGICAL FUNCTION SETRW(IW)
               C
               C     THIS FUNCTION SETS UP THE RESERVED WORD TABLE.
               C     ENTRIES IN THIS TABLE ARE THE FUNCTION NAMES USED IN THE INSTRUCTION
               C     GRAPHS :  THEY WILL BE THE FUNCTION-ATOMS IN THE SYSTEM.
               C
               C     ENTRY RESWORD(W) RETURNS T OR F DEPENDING ON WHETHER OR NOT W IS A
               C     RESERVED WORD
               C
000002          COMMON/OSFLAG/IOFLAG
000002          COMMON IBUF(120)
000002          COMMON/RWTABL/IHTAB(54),ISTAB(2,54)
000002          DATA IHTAB,ISTAB/162*0/
               C         READ NWORDS AND INITIALIZE
000002          READ 1000,NWORDS
000010     1000 FORMAT(I2)
000011          IFREE=1  $  I=0  $  J=0  $  IR=1
000014          IP=LSHIFT(IOFLAG,-1).AND.1
000020          SETRW=.TRUE.
               C         READ 8 RESERVED WORDS INTO BUFFER
000023        5 READ 1010,(IBUF(I1),I1=1,8)
000031     1010 FORMAT(8A10)
000032          J=0
000033       10 I=I+1  $  J=J+1
000036          IWORD=IBUF(J)  $  GO TO 25
               C         ENTRY FOR LOOKUP ONLY
000040          ENTRY RESWORD
000046          IWORD=IW
               C         HASH THE WORD
000047       25 IHAD=MOD(LSHIFT(IWORD,-30),53)+1
               C         CHECK TO SEE IF HASH TABLE IS SET
000056          IF(IHTAB(IHAD).NE.0) GO TO 30
000061          SETRW=.FALSE.
000062          IF(IR.EQ.0) RETURN
               C         PLACE WORD IN S-TABLE AND SET H-TABLE POINTER TO WORD
000064       15 IHTAB(IHAD)=IFREE
000070       20 ISTAB(1,IFREE)=IWORD
```

```
000071          IFREE=IFREE+1
000074       22 IF(I.LT.NWORDS) GO TO 40  $  IR=0  $  IF(IP.EQ.0) RETURN
000102          PRINT 2005  $  PRINT 2010,(I1,IHTAB(I1),ISTAB(1,I1),ISTAB(2,I1),
                1 I1=1,54)   $   PRINT 2015
000137       2005 FORMAT(///,90X,*RESERVED WORD TABLE*,//,84X,*INDEX*,4X,*H-TAB*,7:,
                * *S-TAB(VALUE)*,2X,*CHAIN POINTER*,/)
000140       2010 FORMAT(85X,I2,*:*,2X,I5,10X,A10,2X,I5)
000140       2015 FORMAT(//)
000140          RETURN
             C          CHECK TO SEE IF ALL OF BUFFER HASH BEEN PROCESSED
000142       40 IF(J.LT.8) 10,5
             C
             C          HASH TABLE ADDRESS WAS NON-EMPTY--CHECK THE CHAIN FOR WORD
             C
000147       30 ISAD=IHTAB(IHAD)
000151       50 IF(ISTAB(1,ISAD).NE.IWORD) GO TO 60
000154          IF(IR.NE.0) GO TO 22
000155          SETRW=.TRUE.
000156          RETURN
             C          NO MATCH--CHECK TO SEE IF THERE IS MORE CHAIN
000157       60 IF(ISTAB(2,ISAD).NE.0) GO TO 70
1000161         SETRW=.FALSE.
000162          IF(IR.EQ.0) RETURN
000164          ISTAB(2,ISAD)=IFREE
000167          GO TO 20
             C          CHAIN CONTINUES--FOLLOW IT SEARCHING FOR MATCH OR END
000167       70 ISAD=ISTAB(2,ISAD)
000171          GO TO 50
000172          END
006200
1                SUBROUTINE FEPLOAD
             C
             C     THIS SUBROUTINE READS IN THE FEP-S AND TABLES THEM INTO THE
             C     COMMON BLOCK /FEPS/.  IT ASSUMES THE PROGRAM NAME CARD
             C     ASSIGNMENT TAPE5=INPUT WAS MADE.  A CARD WITH * IN COL. 1
             C     TERMINATES THE FEP INPUT DECK.
             C
000001          COMMON/FEPS/LSIDE(60,5),NARROW(60),NRSIDE(60,3),NACTION(60),
                * NACTNO(60),NSTAR(60),JUMP(60),NUML(60),NUMR(60)
000001          COMMON LABEL(60),NJMP(60)
000001          COMMON/OSFLAG/IOSEL
000001          DATA JUMP/60*0/,KBLANK/1H /
             **IF NO. OF FEPS EXCEEDS 60, VALUE OF *LIMIT* AND ARRAY DIMENSIONS MUST CHANGE
000001          LIMIT=61
             **K INDEXES FEP TABLES DURING READ, THEN HOLDS COUNT
000002          K=0
000003       1  K=K+1
000005          IF(K.EQ.LIMIT) GO TO 900
             * READ AND TABLE NEXT FEP
000006          READ 100,LABEL(K),(LSIDE(K,6-I),I=1,5),NARROW(K),(NRSIDE(K,4-J),
                * J=1,3),NACTION(K),NACTNO(K),NSTAR(K),NJMP(K)
000050       100  FORMAT(A3,5(1X,A7),A1,3(A7,1X),A5,I2,1X,A1,A3)
000050          IF(EOF,5) 902,2
             * IF ENDFILE, ERROR EXIT
000053       2  IF(LABEL(K).NE.1H*) GO TO 1
             **READING COMPLETE; ASSIGN NUMERIC JUMP VALUES CORRESPONDING TO NUMBER OF FEP
             **CONTAINING LABEL AND COUNT LEFT AND RIGHT SIDE NONBLANK ENTRIES
000056       3  K=K-1
000060          NUML(1)=NUMR(1)=1
000062          DO 10 I=2,K
000063          IF(LABEL(I).EQ.KBLANK) GO TO 5
             **IF FEP LABELED, FIND AND SET ALL JUMPS TO LABEL
```

```
000065          DO 4 J=1,K
000067          IF(LABEL(I).EQ.NJMP(J)) JUMP(J)=I
000075      4   CONTINUE
           **COUNT ENTRIES
000100      5   N=0
000101      6   N=N+1
000103          IF(N.EQ.6) GO TO 7
000104          IF(LSIDE(I,N).NE.KBLANK) GO TO 6
000110      7   NUML(I)=N-1
000113          N=0
000113      8   N=N+1
000115          IF(N.EQ.4) GO TO 9
000116          IF(NRSIDE(I,N).NE.KBLANK) GO TO 8
000122      9   NUMR(I)=N-1
000125      10  CONTINUE
           **FEPS ASSEMBLED; LIST IF I/O SELECTION FLAG BIT 0 SET, ELSE RETURN
000127          IF(IOSEL.AND.1) 11,13
000131      11  PRINT 101
000135      101 FORMAT(12X,*N LAB*,3X,*L5*,6X,*L4*,6X,*L3*,6X,*L2*,6X,*L1*,4X,*R*,
           *  3X,*R3*,6X,*R2*,6X,*R1*,4X,*ACTION*,3X,*S*,2X,*JUMP*,7X,*N*,5X,
           *  *L  R*,/)
000135          DO 12 I=1,K
000137          PRINT 102,I,LABEL(I),(LSIDE(I,6-N),N=1,5),NARROW(I),(NRSIDE(I,
           *  4-N),N=1,3),NACTION(I),NACTNO(I),NSTAR(I),NJMP(I),JUMP(I),I,
           *  NUML(I),NUMR(I)
000212      102 FORMAT(10X,I3,1X,A3,5(1X,A7),1X,A1,1X,3(A7,1X),A5,I3,1X,A1,A3,
     1     *  I5,I6,5X,I1,2X,I1)
000212      12  CONTINUE
           **NORMAL RETURN
000215      13  RETURN
           **IF TABLE LIMIT REACHED, CHECK IF NEXT CARD HAS TERMINATOR
000216      900 READ 100,N
000224          IF(EOF,5) 902,901
000227      901 IF(N.EQ.1H*) GO TO 3
           **IF TERMINATOR, GO ASSEMBLE ELSE ABORT
000231          PRINT 103
000235      103 FORMAT(15X,*NO. OF FEPS EXCEEDS LIMIT OF 175*)
000235          GO TO 903
           **IF ENDFILE, ABORT
000236      902 PRINT 104
000242      104 FORMAT(15X,*IMPROPER TERMINATION OF FEP DECK*)
           **ABNORMAL EXIT
000242      903 PRINT 105
000246      105 FORMAT(15X,*EXECUTION TERMINATED*)
000246          END
006300
     1          SUBROUTINE PARSER
               C
               C     THIS SUBROUTINE INTERPRETS THE FEP-S PREVIOUSLY READ INTO THE
               C     COMMON BLOCK FEPS.
               C     INITIALLY, IT SETS UP THE TABLE(S) OF FEP MATCHES USED BY THE
               C     EXEC-SEM-S
               C
000001          COMMON/OSFLAG/IOFLAG
000001          COMMON/FEPS/LSIDE(60,5),NARROW(60),NRSIDE(60,3),NACTION(60),
               *  NACTNO(60),NSTAR(60),JUMP(60),NUML(60),NUMR(60)
000001          COMMON ISYN(50),ISEM(50),ISTK,IGRBG(19)
000001          DATA NPCUR,NPJMP,NBLANK/0,1,10H          /
000001          ISTK=1
               C          NPCUR=CURRENT PRODUCTION, NPJMP=CURRENT SEQUENCE
000002      5 NPCUR=NPCUR+1
               C          TEST FOR MATCH WITH CURRENT PRODUCTION
```

```
000004       15 I=1
000005          IF(LSIDE(NPCUR,I).NE.10H<SG>        ) GO TO 10
000011          ISG=ISYN(ISTK)    $    ISMSG=ISEM(ISTK)
000015          GO TO 20
000015       10 IF(LSIDE(NPCUR,I).NE.ISYN(ISTK-I+1)) GO TO 5
000023       20 I=I+1    $    IF(I.LE.NUML(NPCUR)) GO TO 10
                C          PERFORM DESIRED ACTION, IF INDICATED
000027          IF(NACTION(NPCUR).EQ.NBLANK) GO TO 30
000031          IF(NACTION(NPCUR).EQ.10HHALT        ) RETURN
000034          IF(NACTION(NPCUR).EQ.10HEXEC        ) CALL EXECSYN(NACTNO(NPCUR))
000040          IF(NACTION(NPCUR).EQ.10HERROR       ) CALL ERROR(NACTNO(NPCUR))
                C          PERFORM REPLACEMENT, IF INDICATED
000044       30 IF(NARROW(NPCUR).EQ.NBLANK) GO TO 40
000047          ISTK=ISTK-NUML(NPCUR)+1      $  I=NUMR(NPCUR)
000053       32 IF(NRSIDE(NPCUR,I).NE.10H<SG>        ) GO TO 35
000060          ISYN(ISTK)=ISG  $  ISEM(ISTK)=ISMSG   $   GO TO 40
000064       35 ISYN(ISTK)=NRSIDE(NPCUR,I)
000071          IF(I.EQ.1) GO TO 40
000072          I=I-1  $  ISTK=ISTK+1  $  GO TO 32
                C          SCAN NEXT CHARACTER, IF INDICATED
000074       40 IF(NSTAR(NPCUR).EQ.NBLANK) GO TO 50
000077          ISTK=ISTK+1
000100          IF(ISTK.GT.50) GO TO 70
000103          CALL SCANNER(ISYN(ISTK),ISEM(ISTK))
                C          IF TRACE IS DESIRED, OUTPUT STACKS AND PROD. NOS.
000105       50 IF(IOFLAG.AND.4) 52,60
000107       52 IF(LINEF.EQ.0) GO TO 54
000110          IF(LINEC.LT.LINEF.OR.LINEC.GT.LINEL) GO TO 60
000117       54 PRINT 1000,NPJMP,NPCUR,(ISYN(ISTK-5+I),I=1,5)
000136     1000 FORMAT(50X,*SEQUENCE*,I4,*, LINE*,I4,* MATCHED  *,5A10)
000136          IF(IOFLAG.AND.8) PRINT 1010,(ISEM(ISTK-5+I),I=1,5)
000153     1010 FORMAT(82X,5A10)
                C          JUMP TO NEXT PRODUCTION
000153       60 NPCUR=NPJMP=JUMP(NPCUR)
000156          GO TO 15
000156       70 PRINT 1020
000162     1020 FORMAT(6H *****,*ERROR--PARSE STACK TOO LARGE*)
000162          RETURN
000163          END
006500
1               SUBROUTINE SCANNER(ICLASS,IFORM)
                C
                C     THIS SCANNER RECOGNIZES 4 CLASSES OF SYMBOLS:
                C          1 -- <IDENTIFIER>
                C          2 -- <NUMBER>
                C          3 -- RESERVED WORD
                C          4 -- SPECIAL SYMBOL
                C          5 -- EOF
                C
                C     IT REQUIRES A SUBROUTINE GETCHAR WHICH RETRIEVES THE NEXT CHARACTER FROM
                C     THE INPUT STRING, DETERMINES ITS SUBCLASS, AND TAKES CARE OF READING THE
                C     SOURCE PROGRAM DECK.
                C
000004          DIMENSION ICHAR(12)
000004          DATA ICH/80/
                C          SKIP BLANKS IF NECESSARY
000004       10 CALL GETCHAR(ICL,ICHAR(1),ICH)    $    IF(ICL.EQ.4) GO TO 10
                C
                C     PROCESS INTEGER
                C
000013       15 IF(ICL.NE.2) GO TO 30
000015          I=2    $    ICLASS=10H<NUMBR>
```

```
000017          IF(ICHAR(1).NE.1R0) GO TO 20
                C       SKIP LEADING 0S
000021       45 CALL GETCHAR(ICL,ICHAR(1),ICH)
000024          IF(ICHAR(1).EQ.1R0) GO TO 45
000030          IF(ICL.EQ.2) GO TO 20
000032          ICHAR(1)=1R0    $    I=1    $    GO TO 60
000034       20 CALL GETCHAR(ICL,ICHAR(I),ICH)
000040          IF(ICL.NE.2) GO TO 25
000044          I=I+1    $    IF(I.EQ.13) I=12    $    GO TO 20
000051       25 I=I-1    $    IF(I.LE.10) GO TO 60
                C       INTEGER TOO LONG--PRINT ERROR MSG BUT CONTINUE
000055          ICH1=ICH-1
000056          PRINT 9999,ICH1
000064    9999 FORMAT(5X,5H*****,*ERROR--ABOVE LINE CONTAINS ILLEGAL INTEGER ENDI
                *NG AT COL.*,I3)
000066          I=10
000067       60 ENCODE(10,2000,IFORM) (ICHAR(I1),I1=1,I)  $  ICH=ICH-1  $ RETURN
000103    2000 FORMAT(10R1)
                C
                C       PROCESS IDENTIFIERS AND RESERVED WORDS
                C
000103       30 IF(ICL.NE.1) GO TO 50
000105          I=2    $    ICLASS=10H<IDENT>
000107       40 CALL GETCHAR(ICL,ICHAR(I),ICH)
000113          IF(ICL.GT.2) GO TO 35
000120          I=I+1    $    IF(I.EQ.12) I=11  $  GO TO 40
000125       35 I=I-1    $    IF(I.GT.10) I=10
000132          ENCODE(10,2000,IFORM) (ICHAR(I1),I1=1,I)
000144          ICH=ICH-1  $  RETURN
                C
                C       PROCESS SPECIAL CHARACTER--STORE CHAR. AND RETURN
                C
000146       50 IF(ICL.NE.3) GO TO 70
000150          ENCODE(10,2000,ICLASS) ICHAR(1)
000161          IFORM=ICLASS    $    RETURN
                C
1               C       PROCESS END-OF-FILE -- STORE E-O-F AND RETURN
                C
000164       70 ICLASS=IFORM=10HE-O-F
000166          RETURN    $    END
006700
1               SUBROUTINE GETCHAR(ICLASS,ICHAR,ICH)
                C
                C     THIS SUBROUTINE IS USED BY THE SCANNER TO GET THE NEXT CHARACTER.
                C     IT RETURNS THE CHARACTER (OR EOF) AND ITS CLASS, DEFINED AS FOLLOWS:
                C     1-LETTER  2-DIGIT  3-SP. SYMBOL  4-BLANK  5-EOF
                C     IT ASSUMES THE PROGRAM NAME CARD ASSIGNMENT TAPES=INPUT WAS MADE.
                C
000005          DATA IEOF/0/
000005          COMMON/OSFLAG/IOFLAG
000005          DIMENSION ICARD(80)
                C       INCREMENT POINTER TO GET NEXT CHARACTER
000005          ICH=ICH+1
000006          IF(IEOF.EQ.0) GO TO 5
000007          ICLASS=5    $    RETURN
                C       CHECK TO SEE IF NEED NEW CARD
000011       5  IF(ICH.LE.80) GO TO 10
000013          READ 1000,ICARD  $  LINEC=LINEC+1
000023          IF((LINEC.EQ.1).AND.(IOFLAG.AND.1024)) PRINT 2010
000037    1000 FORMAT(80R1)
000037    2010 FORMAT(1H1)
000037          ICH=1
```

```
                C           CHECK TO SEE WHETHER END-OF-FILE WAS ENCOUNTERED
000040                  IF(EOF.5) 99,50
                C           END-OF-FILE  --  PRINT ERROR MESSAGE AND RETURN
000043             99 ICLASS=5    $     IEOF=1     $     RETURN
000046             50 IF(IOFLAG.AND.1024) PRINT 2000,ICARD
000060           2000 FORMAT(1X,80R1)
                C           SUCCESSFUL READ PERFORMED--ASSIGN CLASS
000060             10 ICHAR=ICARD(ICH)
                C           IF BLANK, CLASS=4
000061                  IF(ICHAR.NE.1H ) GO TO 20
000063                  ICLASS=4    $    RETURN
                C           IF LETTER, CLASS=2
000065             20 IF(ICHAR.GT.1HZ) GO TO 30
000070                  ICLASS=1    $    RETURN
                C           IF DIGIT, CLASS=2
000071             30 IF(ICHAR.GT.1H9) GO TO 40
000074                  ICLASS=2    $    RETURN
                C           MUST BE SPECIAL CHARACTER--CLASS=3
000075             40 ICLASS=3    $    RETURN    $    END
007200
1                       SUBROUTINE EXECSYN(N)
                C
                C       SYNTAX-MATCHING EXECS -- THESE SAVE A TRACE OF THE MATCHING IN
                C       COMMON BLOCK /MATCH/ SO THAT DURING INTERPRETATION THE ACTUAL
                C       SEMANTIC EXECS CAN BE EXECUTED PROPERLY.
                C
000002                  COMMON/MATCH/LITRUL(1200),NEXEC(500),NSTART(250),NLPTR,NRPTR,NFULL
000002                  DATA NLPTR,NRPTR/1,0/
000002                  COMMON ISYN(50),ISEM(50),ISTK,IGRPG(19)
                C
000002                  IF(N.EQ.6) GO TO 6
000004                  IF(N.EQ.15) GO TO 15
000005                  IF(N.EQ.1) GO TO 1
                C           ALL EXECS PACK THEIR NUMBER INTO NEXEC--THIS IS THE ↑TRACE↑
000007             10 CALL PACK(N)  $  RETURN
                C           EXEC 6 MATCHED AN IDENTIFIER--SAVE IT FOR THE SEMANTIC EXEC.
000011              6 LITRUL(NLPTR)=INTGER(ATOM(ISEM(ISTK),1))
000020                NLPTR=NLPTR+1    $    GO TO 20
                C           EXEC 15 MATCHED A NUMBER -- SAME AS 6, ONLY INTEGER ATOM.
000023             15 LITRUL(NLPTR)=INTGER(ATOM(NDECOD(ISEM(ISTK)),0)) $ NLPTR=NLPTR+1
000037             20 IF(NLPTR.LE.1200) GO TO 10
000042                PRINT 1000  $  STOP
000050           1000 FORMAT(* ERROR IN RIGHT-SIDE INPUT--ARRAY LITRUL OVERFLOW*)
                C           EXEC-SYN 1 ESTABLISHES LOCATIONS IN LITRUL AND NEXEC THAT
                C           BEGIN CURRENT RIGHT-SIDE RULE.
000050              1 NRPTR=NRPTR+1  $  NUM=NDECOD(ISEM(ISTK))
000054                NSTART(NUM)=LSHIFT(NLPTR,30).OR.NRPTR  $  NFULL=0  $  GO TO 10
000064                END
007200
1                       SUBROUTINE PACK(N)
                C
                C       PACKS (UNPACKS) INTEGER 0 ≤ N < 32 INTO LOC; IF LOC IS FULL (EMPTY),
                C       INCREMENTS IT AND CONTINUES
                C
000002                  COMMON/MATCH/LITRUL(1200),NEXEC(500),NSTART(250),NLPTR,NRPTR,NFULL
000002                  DATA NEXEC/400*0/,NBYTE/10/,NBIT/6/
                C
000002                  NEXEC(NRPTR)=NEXEC(NRPTR).OR.LSHIFT(N,NFULL*NBIT)
000011             10 NFULL=NFULL+1
000013                IF(NFULL.NE.NBYTE) RETURN
000015                NRPTR=NRPTR+1  $  NFULL=0  $  IF(NRPTR.LE.500) RETURN
000023                PRINT 1000  $  STOP
```

```
000032     1000 FORMAT(* ERROR IN RIGHT-SIDE RULE INPUT--ARRAY NEXEC OVERFLOW*)
           C
000032           ENTRY UNPACK
000040           N=LSHIFT(NEXEC(NRPTR),-NFULL*NBIT).AND.077B
000050           GO TO 10
000051           END)
005100
1                SUBROUTINE CRGRAPH
           C
           C     THIS SUBROUTINE TRACES THROUGH THE PARSE TREE, CALLING =RSIDE=
           C     TO BUILD THE ACTUAL PROGRAM GRAPH.
           C
000001           N=NRULE(-1)
000004         5 LIT=1CH
000005           NTRANS=NRULE(1)  $  IF(NTRANS.EQ.1H+) RETURN
000012           IF(NTRANS.GT.0) GO TO 10
000015           NTRANS=-NTRANS  $  LIT=NRULE(2)
000020        10 CALL RSIDE(NTRANS,LIT)  $  GO TO 5
000023           END
007400
1                FUNCTION NRULE(ITYPE)
000002           COMMON/OSFLAG/IOFLAG
000002           DIMENSION ICARD(80)
000002           IF(ITYPE.LT.0) GO TO 10
000003           LIT=1OH              $  IC=10
000005           IF(ICARD(ICOL).NE.1R ) GO TO 5  $  NRULE = 1H+  $  RETURN
000012         5 IF(ICOL.LE.80) GO TO 20
000015        10 READ 1000,ICARD      $   ICOL=1
000024      1000 FORMAT(80R1)
000025           IF(IOFLAG.AND.2048) PRINT 1010,ICARD
000036      1010 FORMAT(1X,80R1)
000036           IF(ITYPE.LT.0) RETURN
000041        20 IF(ICARD(ICOL).EQ.1R ) GO TO 30
000044           LIT = LSHIFT(LIT,6) .AND. 7777777777777777700B
000047           LIT = LIT .OR. ICARD(ICOL)
000051           ICOL=ICOL+1  $  IC=IC-1
000054           GO TO 5
000055        30 IF(ITYPE.EQ.2) GO TO 40
000057           DECODE(10,2000,LIT) NRULE
000066      2000 FORMAT(I10)
000067           GO TO 50
000070        40 NRULE=LSHIFT(LIT,6*IC)
000076        50 ICOL=ICOL+1
000100           RETURN
000101           END
007200
1                SUBROUTINE RSIDE(NRULE,LITERAL)
           C
           C     THIS IS THE BASIC INTERPRETER OF THE SYSTEM -- IT CALLS THE GRAPH-
           C     BUILDING SEMANTIC EXECS IN SEQUENCE SPECIFIED IN COMMON /MATCH/
           C
000004           COMMON/MATCH/LITRUL(1200),NEXEC(500),NSTART(250),NLPTR,NRPTR,NFULL
000004           COMMON/UNI/UNIV,NONTERM,IG
000004           COMMON/OSFLAG/IOFLAG
           C          GET STARTING LOCN (INDEX) OF FIRST LITERAL
000004           NLPTR=LSHIFT(NSTART(NRULE),-30).AND.077777B
           C          GET STARTING LOCN (INDEX) OF EXEC SEQUENCE
000010           NRPTR=NSTART(NRULE).AND.077777B  $  NFULL=0
           CF            GET EXEC NUMBER TO CALL IF ANY
000015        10 CALL UNPACK(N)  $  IF(N.GT.0) GO TO 15
000023           IF((IOFLAG.AND.2048).EQ.0) GO TO 25
000025           PRINT 1010,NRULE,LITERAL
```

```
000034     1010 FORMAT(* RIGHT-SIDE RULE*,I4,* LITERAL *,A10)
000036          RETURN
000036       25 IF((IOFLAG.AND.128).EQ.0) RETURN
000041          PRINT 1000,NRULE,LITERAL
000051     1000 FORMAT(///,* PARTIAL GRAPH AFTER RULE *,I4,* WITH LITERAL *,A10,/)
000053          CALL TRGRAPH(UNIV,1000,1)
000055          RETURN
                C          CALL SEMANTIC EXEC TO BUILD GRAPH
000056       15 CALL EXECSEM(N,LITERAL)  $  IF(IOFLAG.AND.64) 20,10,20
                C          TRACE SEMANTIC STACK IF DESIRED
000064       20 CALL PRTSEM(N)  $  GO TO 10
000070          END
007200
1               SUBROUTINE PRTSEM(N)
                C
                C     PRINTS TOP 5 WORDS OF SEMANTIC STACK--IF GROPE VALUES,
                C     PRINTS AS SUCH; ELSE AS LITERALS
                C
000002          COMMON ISYN(50),ISEM(50),ISTK,IGRPG(19)
000002          DIMENSION IPRLIST(5)
000002          DO 10 I=1,5
000004          J=ISEM(ISTK-5+I)
000007           5 IF(ISVAL(J)) 15,10,15
000013          15 IF(ISATOM(J)) 20,25,20
000017          25 IF(EQUAL(OBJECT(J),ATOM(7HNONTERM,1))) 30,35,30
000030          30 J=INTGER(VALUE(J))  $  GO TO 20
000036          35 J=INTGER(OBJECT(J))  $  GO TO 5
000044          20 J=IMAGE(J)
000046             IF(J.GE.1000) GO TO 10
000052             ENCODE(10,2000,J) J
000061             J=LSHIFT(J,42)
000064        2000 FORMAT(I10)
000065          10 IPRLIST(I)=J
000071             PRINT 1000,N,IPRLIST
000101        1000 FORMAT(10X,*AFTER EXEC *,I2,*, SEMANTIC STACK:*,5A10)
000102             RETURN  $  END
007200
1               FUNCTION NDECOD(LIT)
                C
                C     FUNCTION DECODES LEFT-JUSTIFIED DISPLAY CODED *LIT* INTO AN
                C     INTEGER, RETURNED AS THE FUNCTION VALUE.
                C
000002          LIT1=LIT
000003           5 IF((LIT1.AND.77B).NE.1B ) GO TO 10
000006             LIT1=LSHIFT(LIT1,54)  $  GO TO 5
000012          10 DECODE(10,1000,LIT1) LIT1
000022        1000 FORMAT(I10)
000023             NDECOD=LIT1
000025             RETURN  $  END
007500
1               FUNCTION RDECOD(LIT)
                C
                C     FUNCTION DECODES LEFT-JUSTIFIED REAL
                C
000002          LIT1=LIT
000003          DECODE(10,1000,LIT1) RLIT
000013        1000 FORMAT(F10.0)
000014          RDECOD=RLIT  $  RETURN  $  END
007500
1               SUBROUTINE EXECSEM(N,LITERAL)
                C
                C     SEMANTIC EXECUTIVE ROUTINE -- THESE APPROX. 15 ROUTINES ARE
```

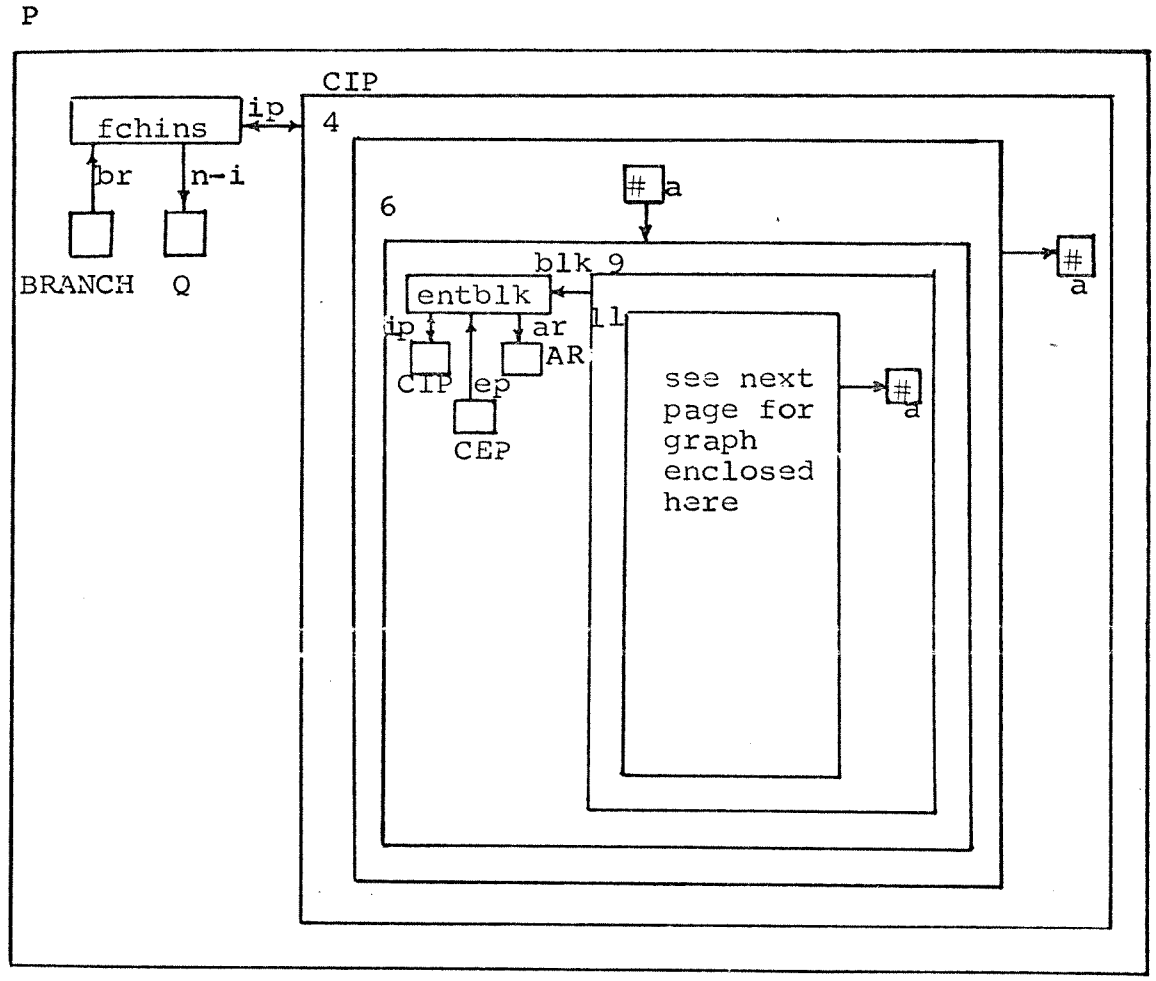
```
                C     USED TO BUILD THE ACTUAL PROGRAM GRAPH STRUCTURE AT RUN-TIME.
                C     THE COMMON BLOCK /MATCH/ FROM *PARSER* CONTAINS ALL THE INFORMATION
                C     NECESSARY FOR THEM TO FOLLOW THE PARSE OF THE GRAPH LANGUAGE AND
                C     CONSTRUCT THE CORRECT STRUCTURES.
                C
000004                COMMON/UNI/UNIV,NONTERM,IG
000004                COMMON/OSFLAG/IOFLAG
000004                COMMON/MATCH/LITRUL(1200),NEXEC(500),NSTART(250),NLPTR,NRPTR,NFULL
000004                COMMON/NTERM/NT(11)
000004                COMMON SYN(50),SEM(50),ISTK,IGRAG(19)
000004                EXTERNAL CHATON,CHAFRN
000004                GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15),N
                C     <NUMBR>
000026             1  IENTRY=0  $  XND=0.  $  RETURN
                C     ↑   <N-LAB> <A-LAB> <N-LAB> ↑
000030             2  IARC=1  $  GO TO 950
                C     [      <N-LAB>    <SG>
000032             3  IF(SEM(ISTK)) 31,30,31
000034            30  IG=IG+1  $  SEM(ISTK)=CHOBJ(CURND,QUOTE(IG))  $  RETURN
000045            31  SEM(ISTK)=CHOBJ(CURND,SEM(ISTK))  $  RETURN
                C     /      <NTERM> =       <S-O-N> /      ↱      /
000053             4  X=RSETI(SEM(ISTK-2))  $  IF(LENGTH(X)) 41,41,40
000065            40  CALL MAPFT(X,CHATON,SEM(ISTK-1))
000073            41  X=RSETO(SEM(ISTK-2))  $  IF(LENGTH(X)) 43,43,42
000105            42  CALL MAPFT(X,CHAFRN,SEM(ISTK))
000113            43  CALL DETND(SEM(ISTK-2))  $  CALL UNREL(SEM(ISTK-2))
000121                DO 44 I=2,11
000124                L=LENGTH(NSET(NT(I)))
000130                IF(L) 44,44,45
000133            45  RDR=CREEDR(NSET(NT(1)))
000141                DO 46 J=1,L
000144            46  X=CHOBJ(TO(RDR),NONTERM)
000155            44  CONTINUE
000157                ISTK=ISTK-3  $  RETURN
                C     <SG>   ↱      <------>  <SG>
000161             5  ISTK=ISTK+1  $  SEM(ISTK)=0.  $  RETURN
                C     <IDENT> ↱      <------>
000165             6  ISTK=ISTK+1  $  SEM(ISTK)=REALE(LITRUL(NLPTR))  $  NLPTR=NLPTR+1
000175                RETURN
                C     ↓   <N-LAB> <A-LAB> <N-LAB> ↓
000175             7  IARC=-1  $  GO TO 950
                C     [
000177             8  XCURND=CURND  $  IF(XND) 80,81,80
000202            80  CURND=XND  $  XND=0.  $  GO TO 82
000205            81  CURND=CRNODE(QUOTE(0),CURGR)
000214            82  CURGR=CRETGR(X)  $  X=HANG(CURND,CURGR)
000222                ISTK=ISTK+1
000223                SEM(ISTK)=CURND
000225                IF(IENTRY.EQ.1) X=HANG(GRAPH(XCURND),CURND)
000235                IENTRY=1
000236                IF(INTGER(SEM(ISTK-1)).EQ.10H=          ) SEM(ISTK-1)=CURND
000247                RETURN
                C     =
000250             9  SEM(ISTK-1)=GETNT(SEM(ISTK))  $  CURND=SEM(ISTK-1)
000256                CURGR=GRAPH(CURND)  $  SEM(ISTK)=10H=          $  RETURN
1               C     <NTERM> <SG>   ↱      <NODE>  <SG>
000263            10  IF(SEM(ISTK)) 100,101,100
000265           101  PRINT 1000. $  SEM(ISTK)=ATOM(3HNIL,1)  $  GO TO 103
000277          1000  FORMAT(6H *****,*ERROR--NON-TERMINAL WITH NULL VALUE ENCOUNTERED--
                     *WILL USE ATOM ↑NIL↑ AS VALUE*)
000277           100  IF(SEM(ISTK).NE.ATOM(7HLITERAL,1)) GO TO 103
000305           102  LT=LSHIFT(LITERAL,6).AND.077B
```

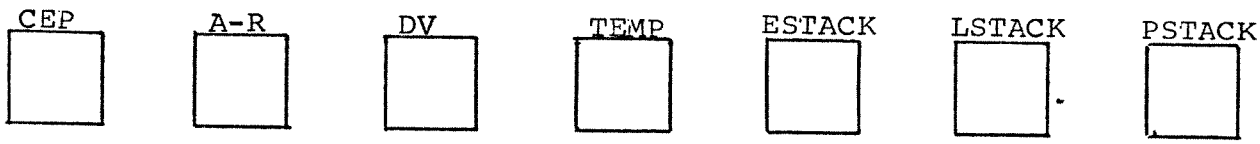
```
000310          IF(LT.GT.1RZ.AND.(LT.LE.1R9.OR.LT.EQ.1R.)) GO TO 104
000326          SEM(ISTK)=ATOM(LITERAL,1)  $  GO TO 105
                C          CONVERT TO INTEGER OR REAL ATOM...
000334        104 LIT=LITERAL
000335          DO 918 I=1,10
000337          LIT=LSHIFT(LITERAL,6)
000343        918 IF((LIT.AND.077B).EQ.1R.) GO TO 919
000347          SEM(ISTK)=ATOM(NDECOD(LITERAL),0)  $  GO TO 105
000360        919 SEM(ISTK)=ATOM(RDECOD(LITERAL),-1)
000371        105 IF(SEM(ISTK-1)) 106,107,106
000373        106 NTLAB=INTGER(SEM(ISTK-1))  $  GO TO 917
000401        107 IG=IG+1  $  NTLAB=INTGER(QUOTE(IG))  $  GO TO 917
000411        103 IF(SEM(ISTK-1)) 915,916,915
000413        915 NTLAB=INTGER(SEM(ISTK-1))  $  GO TO 917
000421        916 NTLAB=NONTERM
000423        917 IF(XND) 109,905,109
000424        109 CURND=CHOBJ(XND,NTLAB)  $  XND=0.  $  GO TO 108
000432        905 CURND=CRNODE(NTLAB,CURGR)
000436        108 SEM(ISTK-1)=HANG(CURND,SEM(ISTK))
000443          IF(INTGER(SEM(ISTK-2)).EQ.10H=          ) SEM(ISTK-2)=CURND
000453          ISTK=ISTK-1
000455          IF(IENTRY.EQ.0) RETURN
000456          X=HANG(CURGR,SEM(ISTK))  $  IENTRY=0  $  RETURN
                C     [     <N-LAB> <S-0-N> ]     ↱     <NODE>
000463         11 CURND=SEM(ISTK-2)  $  CURGR=GRAPH(CURND)  $  XND=0.
000470          ISTK=ISTK-2  $  RETURN
                C     <N>,<A>  <S-0-N>,<S-0-A>  ↱  <S-0-N>,<S-0-A>
000473         12 SEM(ISTK-1)=SEM(ISTK)  $  ISTK=ISTK-1  $  RETURN
                C     (     <N-LAB> <VALUE> )     ↱     <NODE>
000477         13 IF(XND) 130,131,130
000500        130 IF(SEM(ISTK-1)) 907,908,907
000502        907 X=CHOBJ(XND,SEM(ISTK-1))  $  XND=0.  $  GO TO 132
000511        908 IG=IG+1  $  X=CHOBJ(XND,QUOTE(IG))  $  XND=0.  $  GO TO 132
000523        131 IF(SEM(ISTK-1)) 133,134,133
000525        133 X=CRNODE(SEM(ISTK-1),CURGR)  $  GO TO 132
000533        134 X=CRTNODE(CURGR)
000537        132 CURND=X  $  SEM(ISTK-1)=X
000542          IF(SEM(ISTK)) 138,139,138
000543        138 X=HANG(X,SEM(ISTK))
000550        139 IF(INTGER(SEM(ISTK-2)).NE.10H=          ) GO TO 135
000556          SEM(ISTK-2)=X  $  GO TO 914
000561        135 IF(SEM(ISTK).EQ.0) GO TO 137
000563          IF(RESWORD(IMAGE(SEM(ISTK)))) 136,137
000572        136 SEM(ISTK-1)=HANG(SEM(ISTK-1),ATOM(IMAGE(SEM(ISTK)),-3))
000607        137 IF(IENTRY) 913,914,913
000610        913 X=HANG(CURGR,SEM(ISTK-1))  $  IENTRY=0
000616        914 ISTK=ISTK-1  $  RETURN
                C     Ξ↑↓   <N-LAB> <A-LAB> <N-LAB> ↓↑Ξ   ↱   <ARC>
000620         14 IARC=0
000621        950 IF(SEM(ISTK-2)) 140,141,140
                C          HERE CHECK FOR AMBIGUITY ↓ *<N-LAB>↓
1000623       140 IF(SEM(ISTK-1).NE.0..OR.SEM(ISTK).NE.0.) GO TO 906
000632          SEM(ISTK)=SEM(ISTK-2)  $  GO TO 141
000634        906 IF(LENGTH(NSET(SEM(ISTK-2)))) 142,142,143
000644        142 I=IMAGE(SEM(ISTK-2))  $  PRINT 1400,I  $  GO TO 141
000661       1400 FORMAT(6H *****,*ERROR--NODE*,A10,* REFERENCED IN ARC DEFN BEFORE
                 *CREATION:  CURRENT NODE WILL BE USED*)
000661        143 FN=FIRST(NSET(SEM(ISTK-2)))  $  GO TO 144
000671        141 FN=CURND
000673        144 IF(SEM(ISTK-1)) 145,146,145
000675        145 AL=SEM(ISTK-1)  $  GO TO 147
000700        146 AL=QUOTE(0)
```

```
000704      147 IF(SEM(ISTK)) 148,149,148
000706      148 IF(LENGTH(NSET(SEM(ISTK)))) 901,901,900
000716      901 I=IMAGE(SEM(ISTK))  $  PRINT 1400,I  $  GO TO 149
000733      900 TN=FIRST(NSET(SEM(ISTK)))  $  GO TO 902
000743      149 IF(XND) 903,904,903
000744      903 TN=XND  $  GO TO 902
000746      904 TN=CRNODE(QUOTE(0),CURGR)  $  XND=TN
000756      902 IF(IARC) 909,909,910
000760      909 CALL CRARC(FN,AL,TN)
000765      910 IF(IARC) 911,912,912
000767      912 CALL CRARC(TN,AL,FN)
000774      911 ISTK=ISTK-3  $  RETURN
                C     <NUMBER>  →      <------>
000776       15 GO TO 6
000777          ENTRY ERROR
001006          PRINT 1010,N  $  RETURN
001015     1010 FORMAT(80X,5H*****,*ERROR*,I2,* CALLED*)
001015          END
004100
                FUNCTION GETNT(VAL)
1               C     THIS FUNCTION RETURNS AS ITS VALUE THE NODE WITH OBJECT
                C     =NONTERM= AND VALUE =VAL=
000002          COMMON/UNI/UNIV,NONTERM,IG
000002          COMMON/RDRS/RDR,RDR2
000002          L=LENGTH(NSET(NONTERM))
000006          RDR=CHORIG(RDR,NSET(NONTERM))
000014          DO 10 I=1,L
000016          IF(EQUAL(VAL,VALUE(TO(RDR)))) 2,10,2
000027       10 CONTINUE
000032          GETNT=0  $  PRINT 1000
000036     1000 FORMAT(* ERROR--NO NON-TERMINAL FOUND WITH VALUE:*)
000037          X=PRXPFT(VAL,OBJECT)  $  STOP
000044        2 GETNT=REED(RDR)  $  RETURN  $  END
007300
                FUNCTION CRTNODE(GRAPH)
1               C     THIS FUNCTION CREATES (USING CRNODE) A TERMINAL NODE WITH
                C     OBJECT DETERMINED BY A RUNNING COUNTER
000002          COMMON/UNI/UNIV,NONTERM,IG
000002          IG=IG+1
000004          CRTNODE=CRNODE(QUOTE(IG),GRAPH)
000012          RETURN
000012          ENTRY CRETGR
                C     AS ABOVE FOR GRAPHS
000020          IG=IG+1
000022          CRETGR=RELATE(CREGR(QUOTE(IG)))  $  RETURN
000031          END
007400
                SUBROUTINE TRGRAPH(GRAPH,MAXLEV,NUM1)
1               COMMON/RDRS/RDR,RDR2
000005          NUM=MARGIN(NUM1)
000005          LEVR=0
000006          RDR=CHORIG(RDR,RNDSET(GRAPH))
000007       20 XN=TO(RDR)
000021          X=TRNODE(XN)
000023          IF(VALUE(XN)) 15,40,15
000025       15 IF(ISGRAF(VALUE(XN))) 10,40,10
000034       10 IF(LEVR.GE.MAXLEV) GO TO 40
000043          NUM=MARGIN(NUM+2)  $  LEVR=LEVR+1
000045          RDR=DSNDTO(RDR,NDSET(VALUE(XN)))  $  GO TO 20
000053       40 IF(ISATND(RDR)) 30,20,30
000065       30 IF(ISDEEP(RDR)) 50,60,50
000072       50 NUM=MARGIN(NUM-2)
000077
```

```
000103          LEVR=LEVR-1
000105          RDR=ASEND(RDR)  $  GO TO 40
000112       60 RETURN  $  END
007200
1
000002          FUNCTION RNDSET(GR)
000002          EXTERNAL STACK
000007          RNDSET=CREL(QUOTE(0))
000014          CALL MAPFT(NDSET(GR),STACK,RNDSET)
007500          RETURN  $  END
1
000002          FUNCTION TRNODE(NODE)
000002          COMMON/RDRS/RDR,RDR2
000003          CALL TERPRI
                X=PRXPFT(LIST(ATOM(5HNODE:,1),OBJECT(NODE),ATOM(6HVALUE:,1),
               * VALUE(NODE)),OBJECT)
000030       70 L1=LENGTH(RSETI(NODE))  $  IF(L1.LE.0) GO TO 30
000036          RDR2=CHORIG(RDR2,RSETI(NODE))  $  LRSI=LIST(ATOM(6HRSETI:,1))
000050          DO 40 I=1,L1
000052          RC=TO(RDR2)
000054       40 X=CONCAT(LRSI,LIST(LIST(OBJECT(RC),OBJECT(FRNODE(RC)))))
000076          X=PRXPFT(LRSI,OBJECT)
000102       30 L1=LENGTH(RSETO(NODE))  $  IF(L1.LE.0) GO TO 50
000110          RDR2=CHORIG(RDR2,RSETO(NODE))  $  LRSO=LIST(ATOM(6HRSETO:,1))
000122          DO 60 I=1,L1
000124          RC=TO(RDR2)
000126       60 X=CONCAT(LRSO,LIST(LIST(OBJECT(RC),OBJECT(TONODE(RC)))))
000150          X=PRXPFT(LRSO,OBJECT)
000154       50 TRNODE=QUOTE(1)  $  RETURN  $  END
007000
```

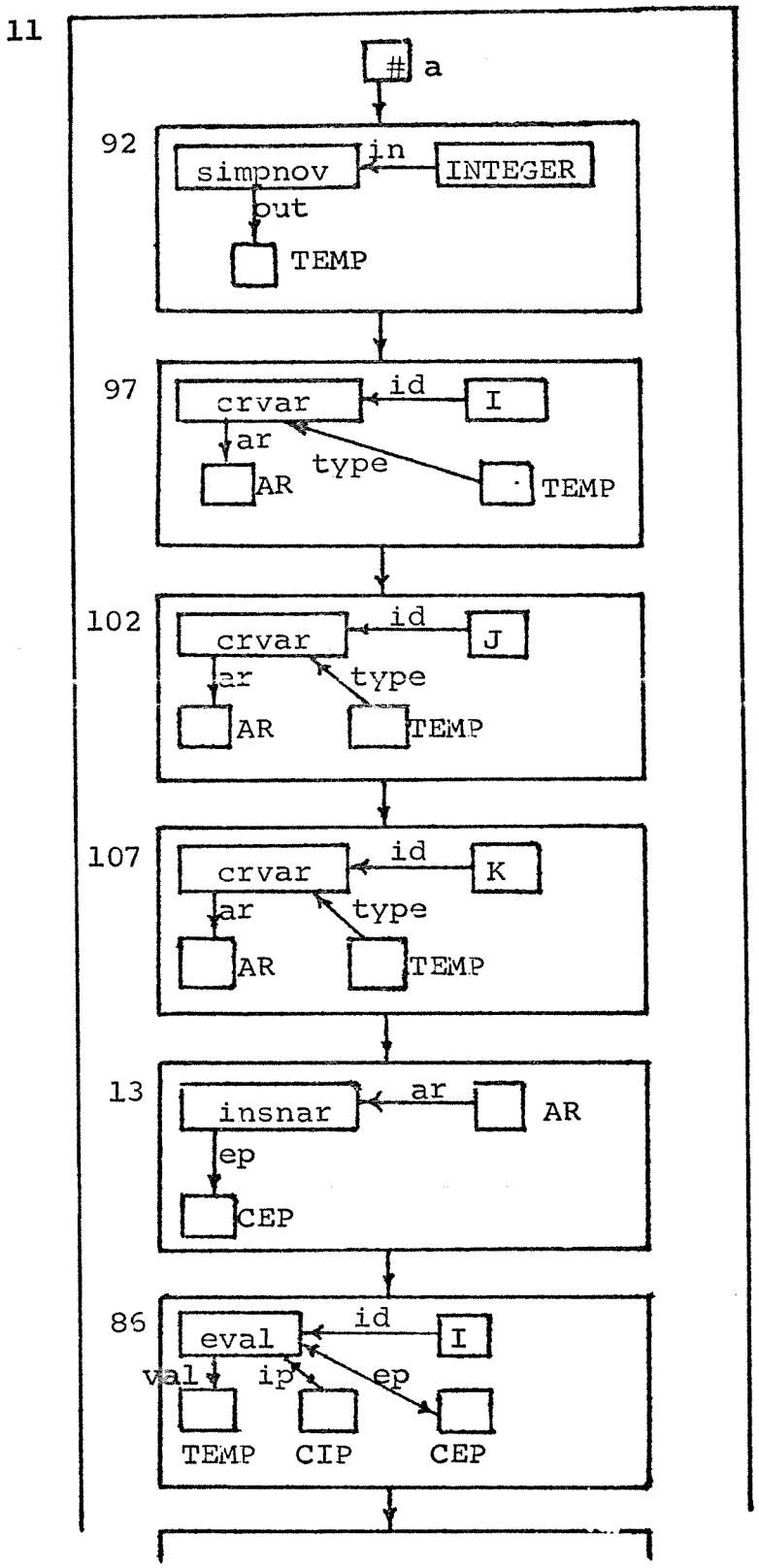
Appendix G: Sample ALGOL Program Graphs

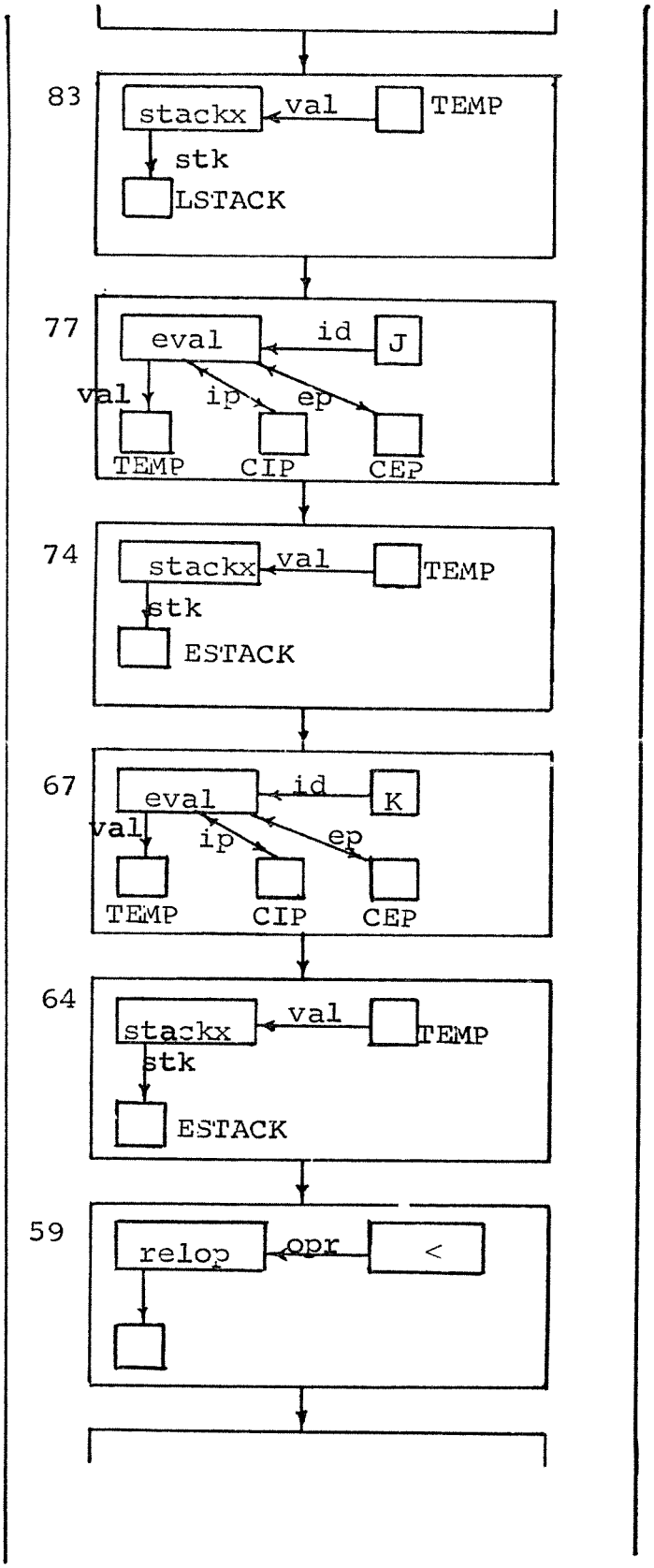


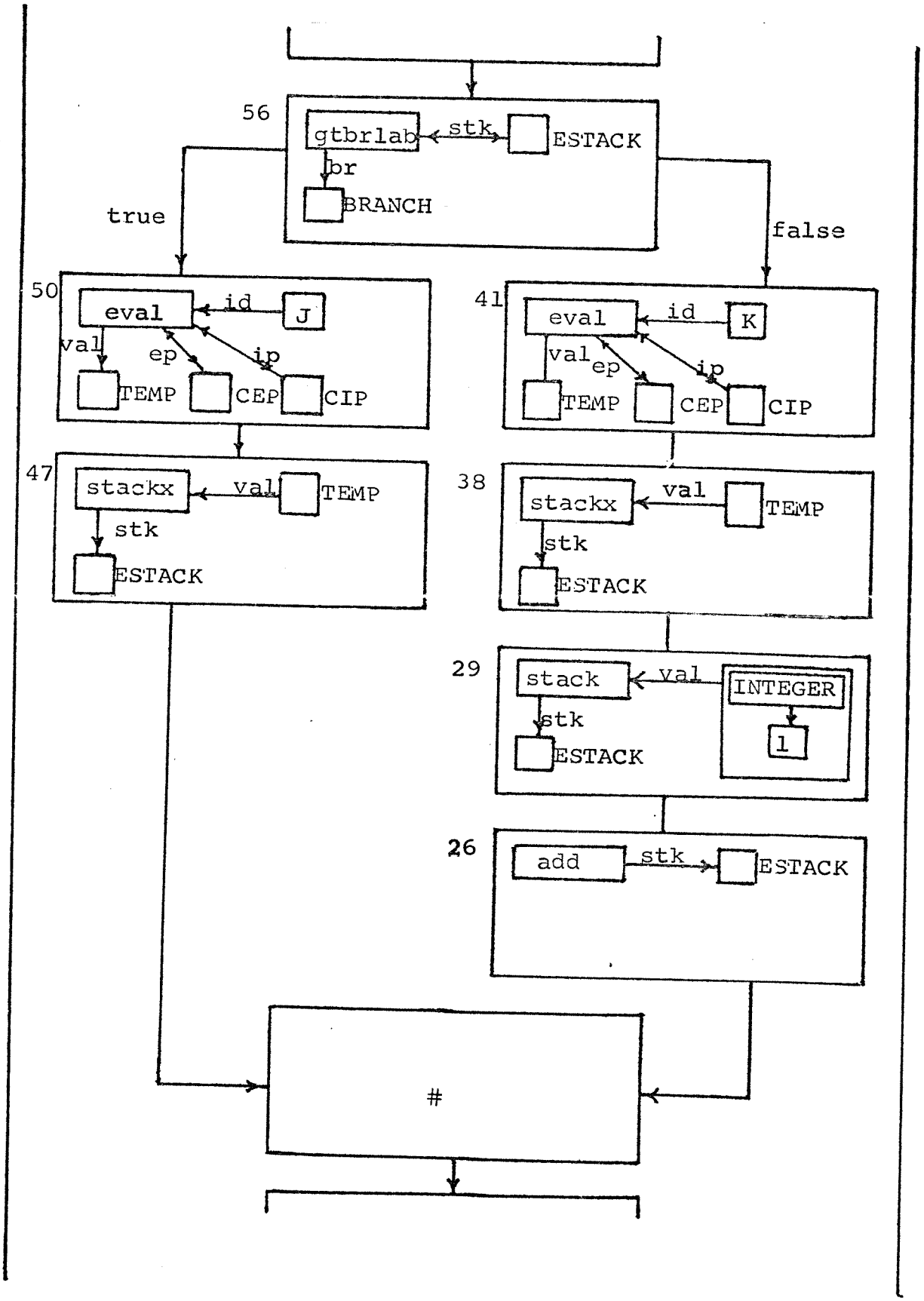


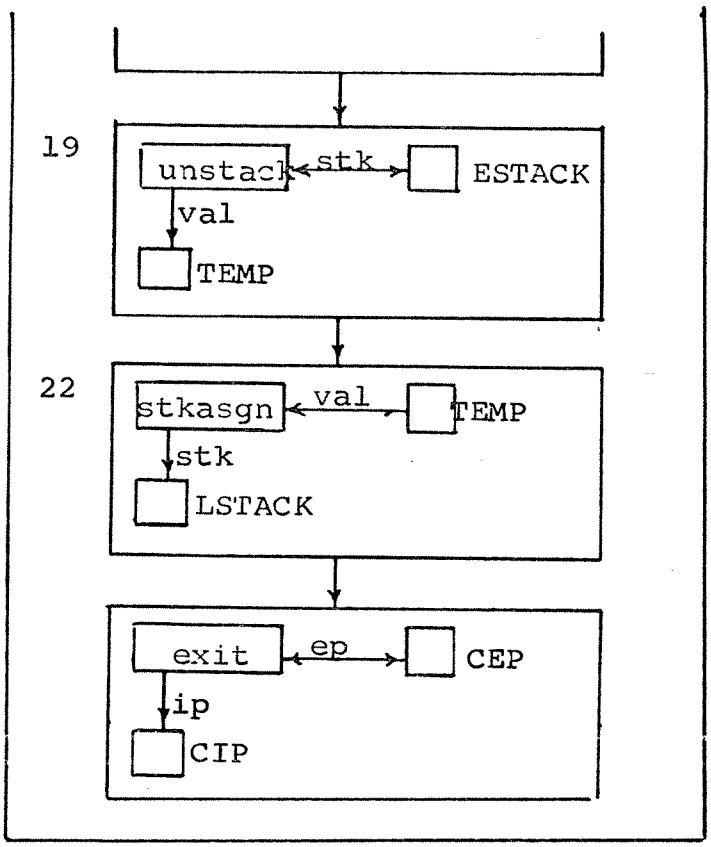
Sample Program:

```
begin
        integer i, j, k;
        i:= if j<k then j else k + 1
end
```


11
#
a
92
simpnov
in
INTEGER
out
TEMP
97
crvar
id
I
ar
type
AR
TEMP
102
crvar
id
J
ar
type
AR
TEMP
107
crvar
id
K
ar
type
AR
TEMP
13
insnar
ar
AR
ep
CEP
86
eval
id
I
val
ip
ep
TEMP
CIP
CEP


83
stackx
val
TEMP
stk
LSTACK
77
eval
id
J
val
ip
ep
TEMP
CIP
CEP
74
stackx
val
TEMP
stk
ESTACK
67
eval
id
K
val
ip
ep
TEMP
CIP
CEP
64
stackx
val
TEMP
stk
ESTACK
59
relop
opr
<


56
gtbrlab
stk
ESTACK
br
BRANCH
true
false
50
eval
id
J
val
ep
ip
TEMP
CEP
CIP
41
eval
id
K
val
ep
ip
TEMP
CEP
CIP
47
stackx
val
TEMP
stk
ESTACK
38
stackx
val
TEMP
stk
ESTACK
29
stack
val
INTEGER
stk
ESTACK
1
26
add
stk
ESTACK
#


19
unstack
stk
ESTACK
val
TEMP
22
stkasgn
val
TEMP
stk
LSTACK
exit
ep
CEP
ip
CIP

```
(NODE: P VALUE: 1)

  (NODE: CIP VALUE: 2)
  (RSETI: (IP 87) (IP 78) (IP 68) (IP 51) (IP 42) (IP 17) (IP 7) (IP FCH))
  (RSETO: (IP 87) (IP 78) (IP 68) (IP 51) (IP 42) (IP 7) (IP FCH))

    (NODE: 4 VALUE: 3)
    (RSETO: (0 A))

      (NODE: 6 VALUE: 5)
      (RSETI: (0 A))

        (NODE: 9 VALUE: 8)
        (RSETO: (BLK 7))

          (NODE: 11 VALUE: 10)
          (RSETO: (0 A))

            (NODE: 107 VALUE: 106)
            (RSETI: (0 102))
            (RSETO: (0 13))

              (NODE: 110 VALUE: 109)
              (RSETO: (ID 108))

                (NODE: 111 VALUE: K)

              (NODE: 108 VALUE: CRVAR)
              (RSETI: (ID 110) (TYPE TEMP))
              (RSETO: (AR AR))

            (NODE: 102 VALUE: 101)
            (RSETI: (0 97))
            (RSETO: (0 107))

              (NODE: 105 VALUE: 104)
              (RSETO: (ID 103))

                (NODE: 112 VALUE: J)

              (NODE: 103 VALUE: CRVAR)
              (RSETI: (ID 105) (TYPE TEMP))
              (RSETO: (AR AR))

            (NODE: 97 VALUE: 96)
            (RSETI: (0 92))
            (RSETO: (0 102))

              (NODE: 100 VALUE: 99)
              (RSETO: (ID 98))

                (NODE: 113 VALUE: I)

              (NODE: 98 VALUE: CRVAR)
```

```
  (RSETI: (ID 100) (TYPE TEMP))
  (RSETO: (AR AR))

(NODE: 92 VALUE: 91)
(RSETI: (0 A))
(RSETO: (0 97))

  (NODE: 95 VALUE: 94)
  (RSETO: (IN 93))

    (NODE: 114 VALUE: INTEGER)

  (NODE: 93 VALUE: SIMPMOV)
  (RSETI: (IN 95))
  (RSETO: (OUT TEMP))

(NODE: 86 VALUE: 85)
(RSETI: (0 13))
(RSETO: (0 83))

  (NODE: 89 VALUE: 88)
  (RSETO: (ID 87))

    (NODE: 90 VALUE: I)

  (NODE: 87 VALUE: EVAL)
  (RSETI: (ID 89) (EP CEP) (IP CIP))
  (RSETO: (EP CEP) (IP CIP) (VAL TEMP))

(NODE: 83 VALUE: 82)
(RSETI: (0 86))
(RSETO: (0 77))

  (NODE: 84 VALUE: STACKX)
  (RSETI: (VAL TEMP))
  (RSETO: (STK LSTACK))

(NODE: 77 VALUE: 76)
(RSETI: (0 83))
(RSETO: (0 74))

  (NODE: 80 VALUE: 79)
  (RSETO: (ID 78))

    (NODE: 81 VALUE: J)

  (NODE: 78 VALUE: EVAL)
  (RSETI: (ID 80) (EP CEP) (IP CIP))
  (RSETO: (EP CEP) (IP CIP) (VAL TEMP))

(NODE: 74 VALUE: 73)
(RSETI: (0 77))
(RSETO: (0 67))

  (NODE: 75 VALUE: STACKX)
  (RSETI: (VAL TEMP))
  (RSETO: (STK ESTACK))

(NODE: 67 VALUE: 66)
(RSETI: (0 74))
(RSETO: (0 64))
```

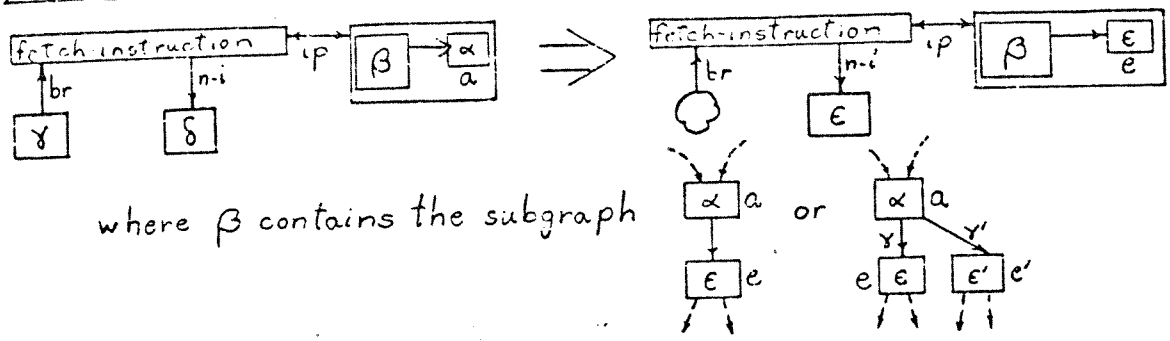
```
  (NODE: 70 VALUE: 69)
  (RSETO: (ID 68))

    (NODE: 71 VALUE: K)

  (NODE: 68 VALUE: EVAL)
  (RSETI: (ID 70) (EP CEP) (IP CIP))
  (RSETO: (EP CEP) (IP CIP) (VAL TEMP))

(NODE: 64 VALUE: 63)
(RSETI: (0 67))
(RSETO: (0 59))

  (NODE: 65 VALUE: STACKX)
  (RSETI: (VAL TEMP))
  (RSETO: (STK ESTACK))

(NODE: 59 VALUE: 58)
(RSETI: (0 64))
(RSETO: (0 56))

  (NODE: 62 VALUE: 61)
  (RSETO: (OPR 60))

    (NODE: 72 VALUE: <)

  (NODE: 60 VALUE: RELOP)
  (RSETI: (OPR 62))
  (RSETO: (STK ESTACK))

(NODE: 56 VALUE: 55)
(RSETI: (0 59))
(RSETO: (TRUE 50) (FALSE 41))

  (NODE: 57 VALUE: GTBRLBL)
  (RSETI: (STK ESTACK))
  (RSETO: (STK ESTACK) (BR BRANCH))

(NODE: 50 VALUE: 49)
(RSETI: (TRUE 56))
(RSETO: (0 47))

  (NODE: 53 VALUE: 52)
  (RSETO: (ID 51))

    (NODE: 54 VALUE: J)

  (NODE: 51 VALUE: EVAL)
  (RSETI: (ID 53) (EP CEP) (IP CIP))
  (RSETO: (EP CEP) (IP CIP) (VAL TEMP))

(NODE: 47 VALUE: 46)
(RSETI: (0 50))
(RSETO: (0 24))

  (NODE: 48 VALUE: STACKX)
  (RSETI: (VAL TEMP))
  (RSETO: (STK ESTACK))

(NODE: 41 VALUE: 40)
(RSETI: (FALSE 56))
(RSETO: (0 38))
```

```
  (NODE: 44 VALUE: 43)
  (RSETO: (IC 42))

    (NODE: 45 VALUE: K)

  (NODE: 42 VALUE: EVAL)
  (RSETI: (IC 44) (EP CEP) (IP CIP))
  (RSETO: (EP CEP) (IP CIP) (VAL TEMP))

(NODE: 38 VALUE: 37)
(RSETI: (0 41))
(RSETO: (0 29))

  (NODE: 39 VALUE: STACKX)
  (RSETI: (VAL TEMP))
  (RSETO: (STK ESTACK))

(NODE: 29 VALUE: 28)
(RSETI: (0 38))
(RSETO: (0 26))

  (NODE: 32 VALUE: 31)
  (RSETO: (VAL 30))

    (NODE: 35 VALUE: 34)
    (RSETI: (0 33))

      (NODE: 36 VALUE: 1)

    (NODE: 33 VALUE: INTEGER)
    (RSETO: (0 35))

  (NODE: 30 VALUE: STACKX)
  (RSETI: (VAL 32))
  (RSETO: (STK ESTACK))

(NODE: 26 VALUE: 25)
(RSETI: (0 29))
(RSETO: (0 24))

  (NODE: 27 VALUE: ADD)
  (RSETO: (STK ESTACK))

(NODE: 24 VALUE:)
(RSETI: (0 47) (0 26))
(RSETO: (0 19))

(NODE: 22 VALUE: 21)
(RSETI: (0 19))
(RSETO: (0 16))

  (NODE: 23 VALUE: STKASGN)
  (RSETI: (VAL TEMP))
  (RSETO: (STK LSTACK))

(NODE: 19 VALUE: 18)
(RSETI: (0 24))
(RSETO: (0 22))

  (NODE: 20 VALUE: UNSTACK)
  (RSETI: (STK ESTACK))
```

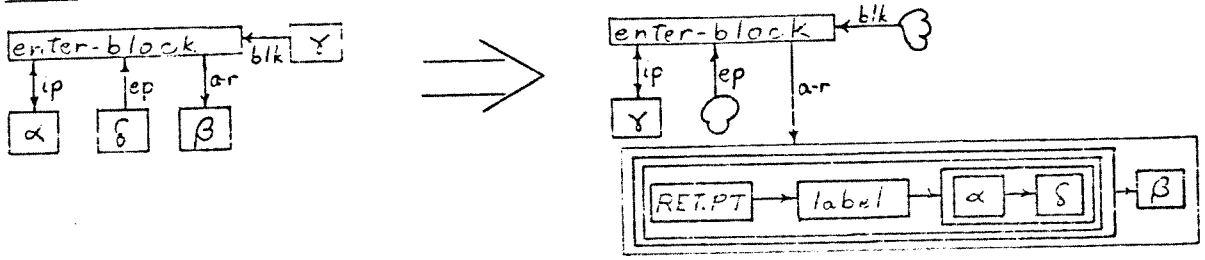
```
                (RSETO: (STK ESTACK) (VAL TEMP))

              (NODE: 16 VALUE: 15)
              (RSETI: (0 22))

                (NODE: 17 VALUE: EXITX)
                (RSETI: (EP CEP))
                (RSETO: (EP CEP) (IP CIP))

              (NODE: 13 VALUE: 12)
              (RSETI: (0 107))
              (RSETO: (0 86))

                (NODE: 14 VALUE: INSNAR)
                (RSETI: (AR AR))
                (RSETO: (EP CEP))

              (NODE: A VALUE:)
              (RSETI: (0 11))
              (RSETO: (0 92))

            (NODE: 7 VALUE: ENTBLK)
            (RSETI: (BLK 9) (EP CEP) (IP CIP))
            (RSETO: (AR AR) (IP CIP))

          (NODE: A VALUE:)
          (RSETI: (0 4))
          (RSETO: (0 6))

        (NODE: Q VALUE:)
        (RSETI: (NI FCH))

        (NODE: BRANCH VALUE:)
        (RSETI: (BR 57))
        (RSETO: (BR FCH))

        (NODE: FCH VALUE: FCHINS)
        (RSETI: (IP CIP) (BR BRANCH))
        (RSETO: (IP CIP) (NI Q))

      (NODE: CEP VALUE:)
      (RSETI: (EP 87) (EP 78) (EP 68) (EP 51) (EP 42) (EP 17) (EP 14))
      (RSETO: (EP 87) (EP 78) (EP 68) (EP 51) (EP 42) (EP 17) (EP 7))

      (NODE: AR VALUE:)
      (RSETI: (AR 108) (AR 103) (AR 98) (AR 7))
      (RSETO: (AR 14))

      (NODE: UV VALUE:)

      (NODE: TEMP VALUE:)
      (RSETI: (OUT 93) (VAL 87) (VAL 78) (VAL 68) (VAL 51) (VAL 42) (VAL 20))
      (RSETO: (TYPE 108) (TYPE 103) (TYPE 98) (VAL 84) (VAL 75) (VAL 65) (VAL 48) (VAL 39) (VAL 23))

      (NODE: ESTACK VALUE:)
      (RSETI: (STK 75) (STK 65) (STK 60) (STK 57) (STK 48) (STK 39) (STK 30) (STK 27) (STK 20))
      (RSETO: (STK 57) (STK 20))

      (NODE: LSTACK VALUE:)
      (RSETI: (STK 84) (STK 23))

      (NODE: RSTACK VALUE:)
```

Appendix H: Function Transformations

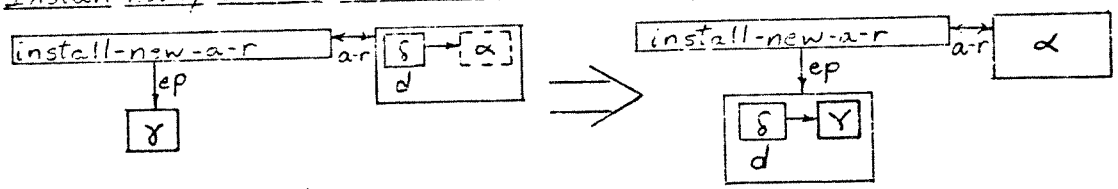
1. Fetch next instruction and update current instruction pointer



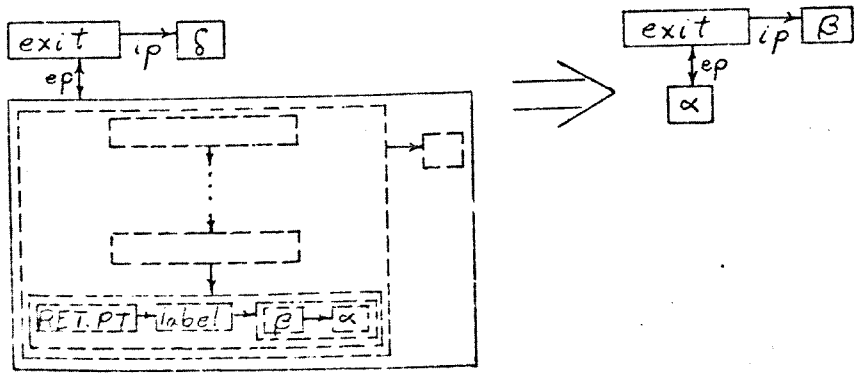
2. Enter block and set return point in new activation record.



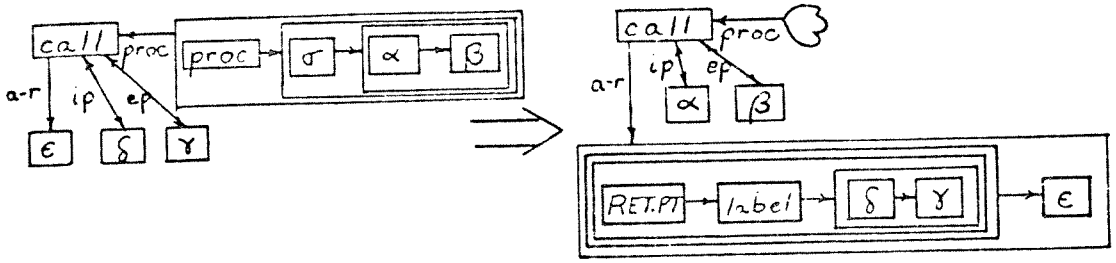
3. Install newly created activation record as top of current environment stack



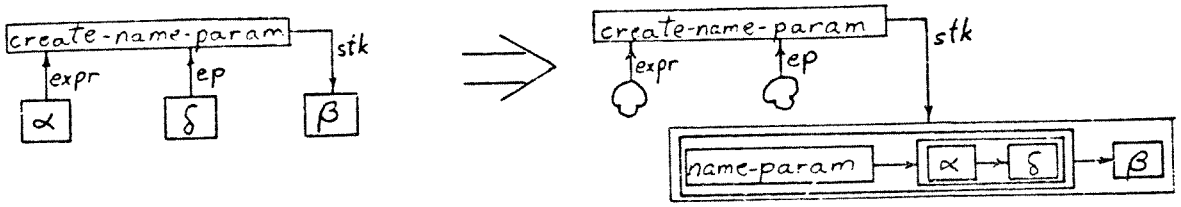
4. Exit block, procedure, or for statement body



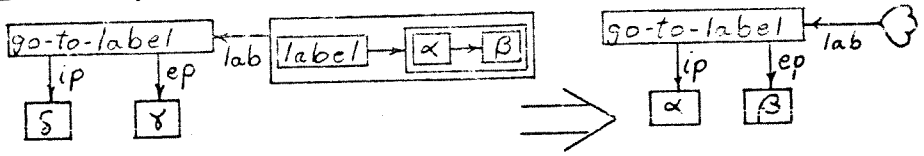
5. Call procedure and set return point in new activation record.



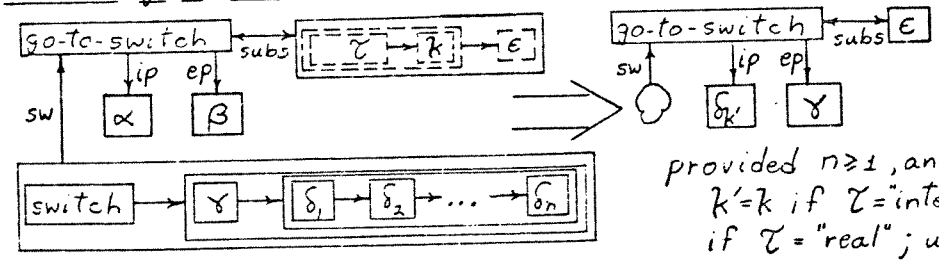
6. Transmit actual parameter by name using parameter stack.



7. Execute go to with label.

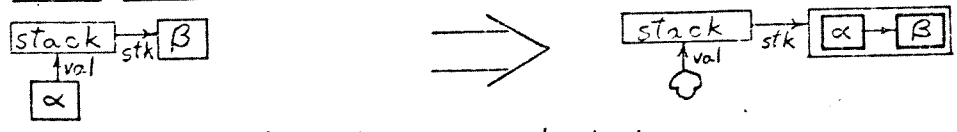


8. Execute go to with switch.

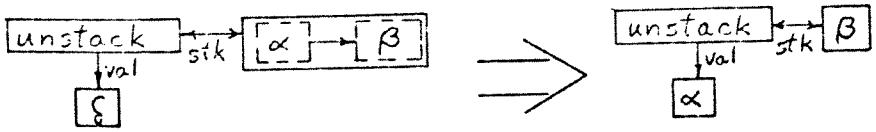


provided $n \geq 1$, and $1 \leq k' \leq n$ where
$k' = k$ if τ = "integ" or k' = entier(k)
if τ = "real"; undefined otherwise.

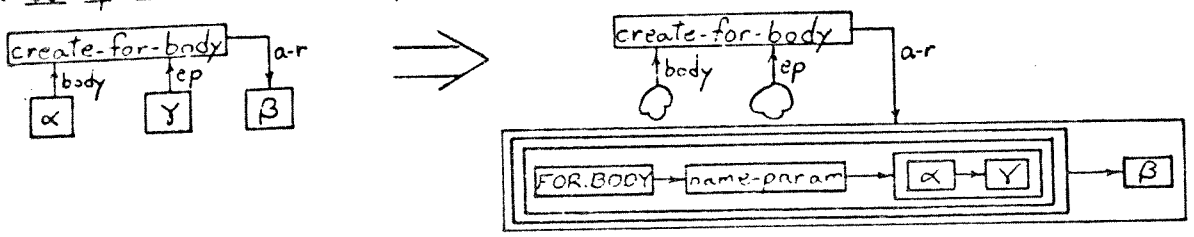
9. Stack value on hierarchical stack.



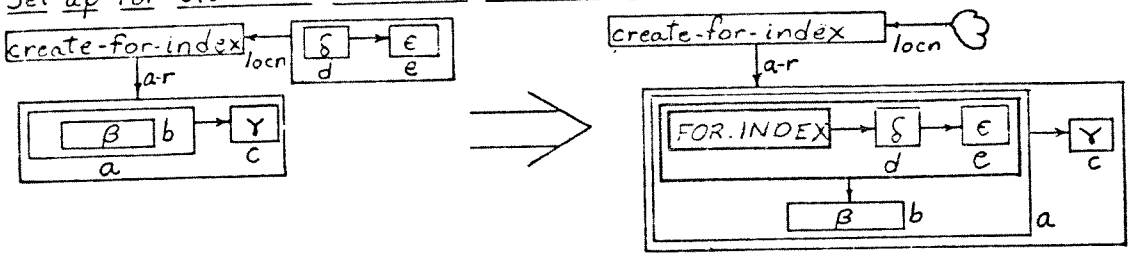
10. Unstack value from hierarchical stack.



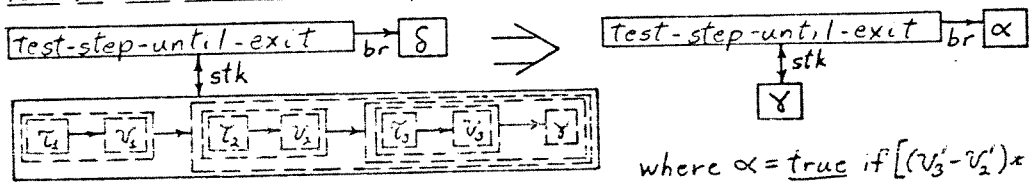
11. Set up for statement body in new activation record.



12. Set up for statement controlled variable in activation record.



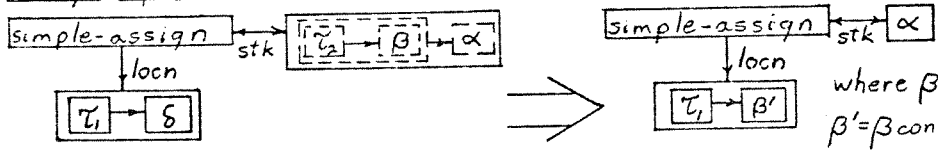
13. Test for exit condition on step-until element in for statement.



where α = true if $[(v_3' - v_2') \times \text{sign}(v_1')] > 0$

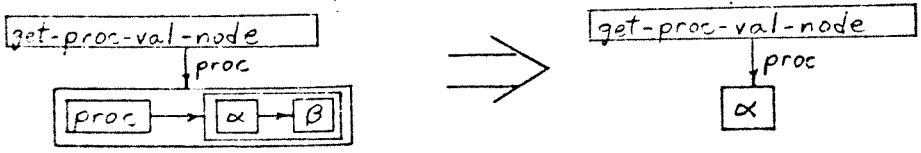
α = false if ≤ 0, provided $\tau_1, \tau_2, \tau_3 \in \{\text{real, integ.}\}$. v_1', v_2', v_3' are v_1, v_2, v_3 converted to type real if τ_1, τ_2 or τ_3 = real, or v_1, v_2, v_3 otherwise.

14. Assign top stack value to a location.

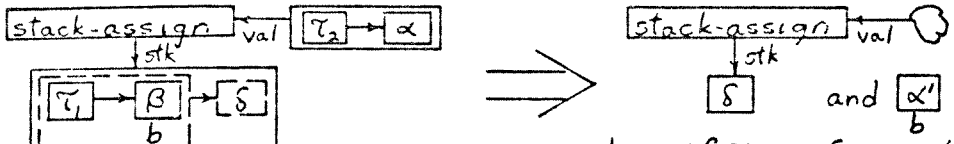


where $\beta' = \beta$ if $\tau_2 = \tau_1$ and $\beta' = \beta$ converted to type τ_1 otherwise, provided $\tau_2, \tau_1 \in \{\text{real, integ.}\}$

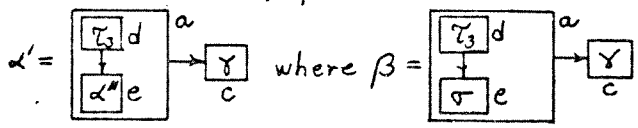
15. Retrieve location for procedure (function) value.



16. Assign value to top element of stack and unstack.

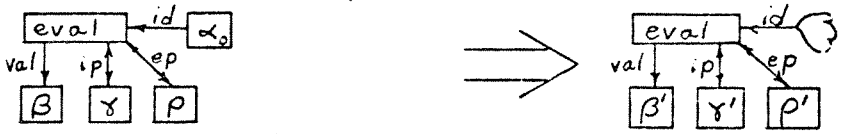


where if $\tau_1, \tau_2 \in \{real, integ.\}$, then $\alpha' = \alpha$ if $\tau_1 = \tau_2$ and $\alpha' = \alpha$ converted to type τ_1, otherwise. If $\tau_1 = proc$ then α' has the form:

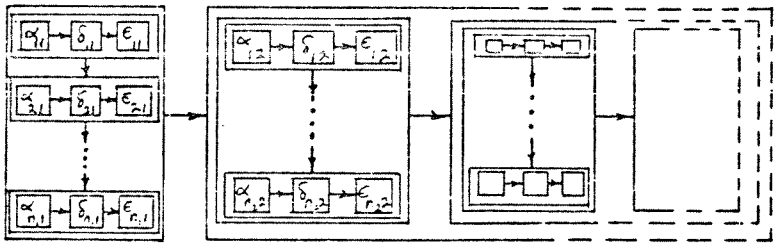


and $\alpha'' = \alpha$ if $\tau_3 = \tau_2$ and $\alpha'' = \alpha$ converted to type τ_3 otherwise.

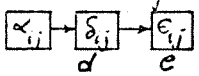
17. Find identifier binding in current environment; if by-name parameter, evaluate it.



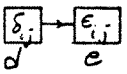
where ρ has the form:



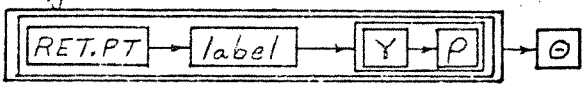
Let α_{ij} be the first α (min i) such that $\alpha_{ij} = \alpha_0$ and let the graph containing α_{ij} be:



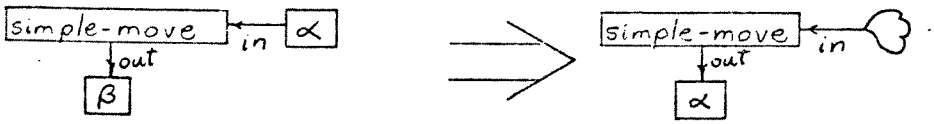
Case 1. If $\delta_{ij} \neq$ name-param then $\gamma' = \gamma$, $\rho' = \rho$ and $\beta' =$



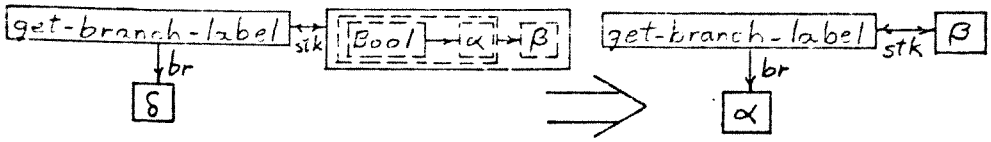
Case 2. If δ_{ij} = name-param then ϵ_{ij} has the form: φ → Θ and $\beta' = \beta$, $\gamma' = \varphi$, and $\rho' =$



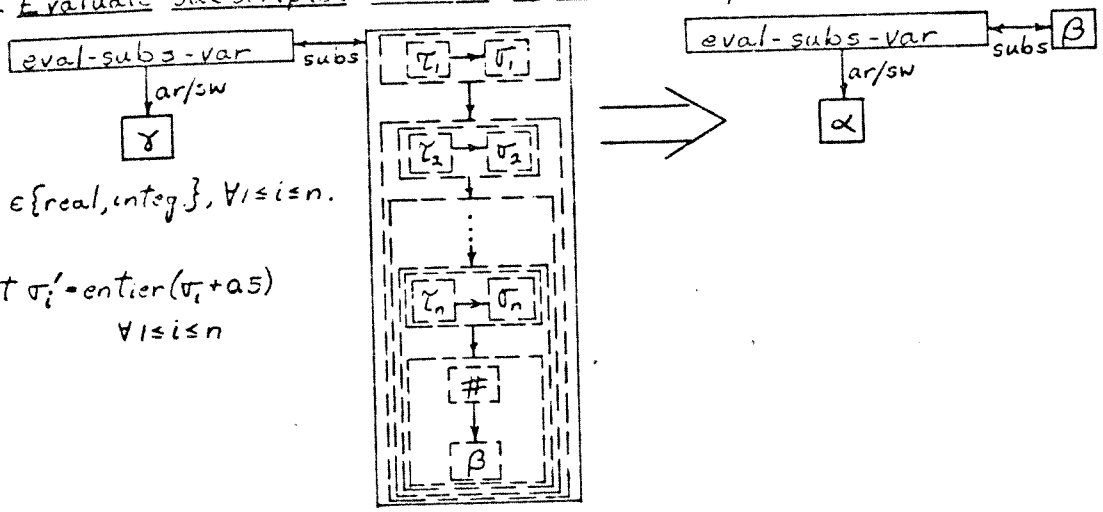
18. Simple inter-node transfer of value.



19. Move Boolean value to branch control node.



20. Evaluate subscripted variable or switch designator.

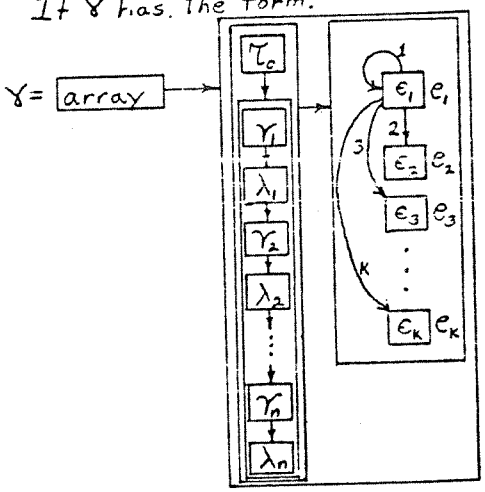


$\tau_i \in \{real, integ\}$, $\forall 1 \le i \le n$.

Let $\sigma_i' \leftarrow entier(\sigma_i + .5)$
$\forall 1 \le i \le n$

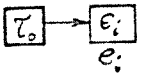
Case 1. (subscripted variable designating array element)

If γ has the form:



then letting $\delta_j = \gamma_j - \lambda_j + 1$, $\forall 1 \le j \le n$

$K = \delta_1 * \delta_2 * \dots * \delta_n$

and $\alpha =$ 

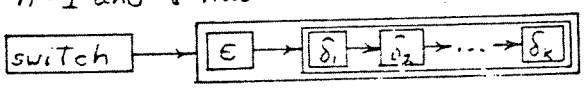
if $1 \le i \le K$

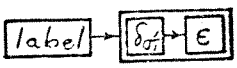
where $i = \sigma_1' + \delta_1[(\sigma_2' - 1) + \delta_2[(\sigma_3' - 1) + \delta_3[\dots + \delta_{n-1}(\sigma_n' - 1)]]]$.

Note: Both the subscripts, $\sigma_1, \sigma_2, \dots \sigma_n$, and the bound pairs, $\lambda_1 : \gamma_1, \lambda_2 : \gamma_2, \dots, \lambda_n : \gamma_n$, are in right-to-left order, i.e. the designated array element has subscript list $[\sigma_n', \sigma_{n-1}', \dots, \sigma_1']$.

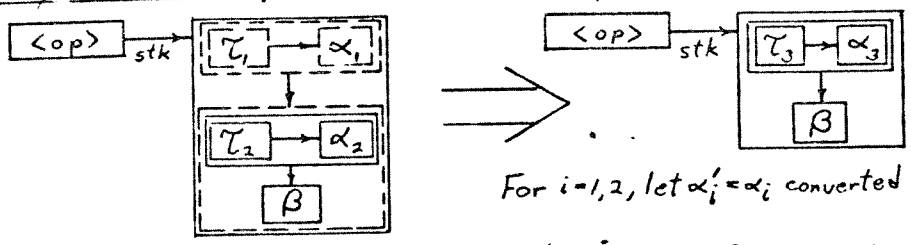
Case 2. (switch designator used as actual parameter and parsed as subscripted variable)

If $n = 1$ and γ has the form:



and $1 \le \sigma_1' \le K$, then $\alpha =$ 

21. Binary arithmetic, logical and relational operations.



For $i=1,2$, let $\alpha_i' = \alpha_i$ converted to real if τ_i = integ.

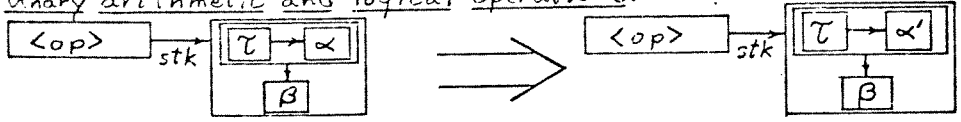
The results of the various binary operations are defined by the following table:

If <op> =	and τ_1 =	and τ_2 =	then τ_3 =	and α_3 =
add	integ	integ	integ	$\alpha_2 +_{int} \alpha_1$
"	"	real	real	$\alpha_2 +_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' +_{real} \alpha_1$
"	"	real	"	$\alpha_2 +_{real} \alpha_1$
subt	integ	integ	integ	$\alpha_2 -_{int} \alpha_1$
"	"	real	real	$\alpha_2 -_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' -_{real} \alpha_1$
"	"	real	"	$\alpha_2 -_{real} \alpha_1$
mult	integ	integ	integ	$\alpha_2 *_{int} \alpha_1$
"	"	real	real	$\alpha_2 *_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' *_{real} \alpha_1$
"	"	real	"	$\alpha_2 *_{real} \alpha_1$
real-div	integ	integ	real	$\alpha_2' /_{real} \alpha_1'$
"	"	real	"	$\alpha_2 /_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' /_{real} \alpha_1$
"	"	real	"	$\alpha_2 /_{real} \alpha_1$
int-div	integ	integ	integ	$\alpha_2 /_{int} \alpha_1$
expon	integ	integ	integ	$\alpha_2 \uparrow_{int} \alpha_1$
"	"	real	real	$\alpha_2 \uparrow_{int} \alpha_1$
"	real	integ	"	$\alpha_2' \uparrow_{real} \alpha_1$
"	"	real	"	$\alpha_2 \uparrow_{real} \alpha_1$
and	Bool	Bool	Bool	$\alpha_2 \wedge \alpha_1$
or	"	"	"	$\alpha_2 \vee \alpha_1$
equiv	"	"	"	$\alpha_2 \equiv \alpha_1$
implic	"	"	"	$\alpha_2 \supset \alpha_1$

If <op> =	and τ_1 =	and τ_2 =	then τ_3 =	and α_3 =
=	integ	integ	Bool	$\alpha_2 =_{int} \alpha_1$
"	"	real	"	$\alpha_2 =_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' =_{real} \alpha_1$
"	"	real	"	$\alpha_2 =_{real} \alpha_1$
$\neq$	integ	integ	"	$\alpha_2 \neq_{int} \alpha_1$
"	"	real	"	$\alpha_2 \neq_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' \neq_{real} \alpha_1$
"	"	real	"	$\alpha_2 \neq_{real} \alpha_1$
<	integ	integ	"	$\alpha_2 <_{int} \alpha_1$
"	"	real	"	$\alpha_2 <_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' <_{real} \alpha_1$
"	"	real	"	$\alpha_2 <_{real} \alpha_1$
$\leq$	integ	integ	"	$\alpha_2 \leq_{int} \alpha_1$
"	"	real	"	$\alpha_2 \leq_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' \leq_{real} \alpha_1$
"	"	real	"	$\alpha_2 \leq_{real} \alpha_1$
>	integ	integ	"	$\alpha_2 >_{int} \alpha_1$
"	"	real	"	$\alpha_2 >_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' >_{real} \alpha_1$
"	"	real	"	$\alpha_2 >_{real} \alpha_1$
$\geq$	integ	integ	"	$\alpha_2 \geq_{int} \alpha_1$
"	"	real	"	$\alpha_2 \geq_{real} \alpha_1'$
"	real	integ	"	$\alpha_2' \geq_{real} \alpha_1$
"	"	real	"	$\alpha_2 \geq_{real} \alpha_1$

Note: Cases such as division by zero where arithmetic operations are undefined have not been detailed.

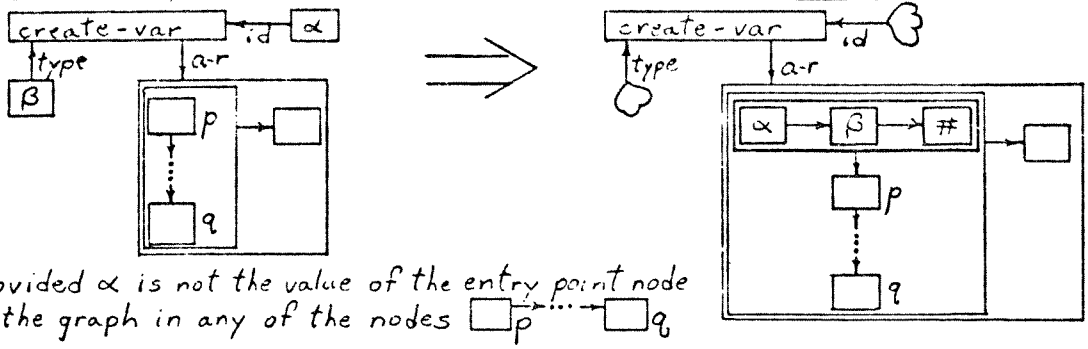
22. Unary arithmetic and logical operations.



Defined by:

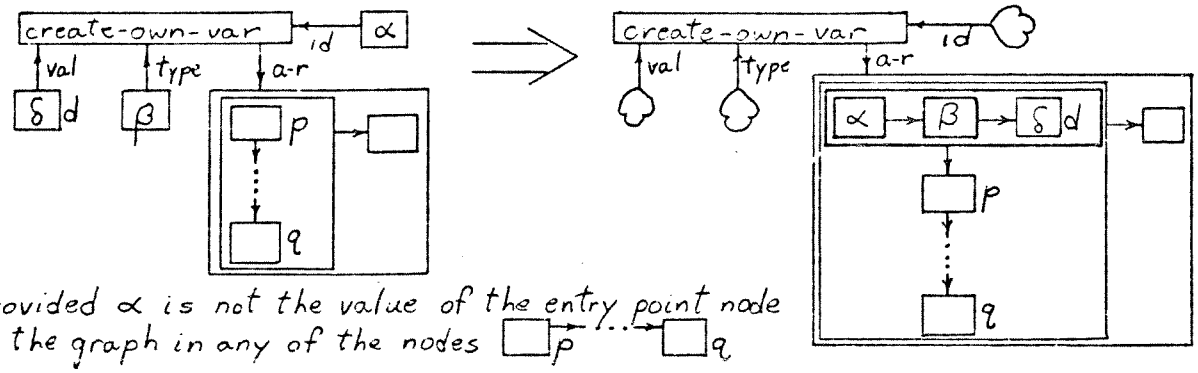
If <op> =	and τ =	then α' =
negate	integ	$-_{int} \alpha$
"	real	$-_{real} \alpha$
not	Bool	$\neg \alpha$

23. Create simple variable and enter in new activation record.



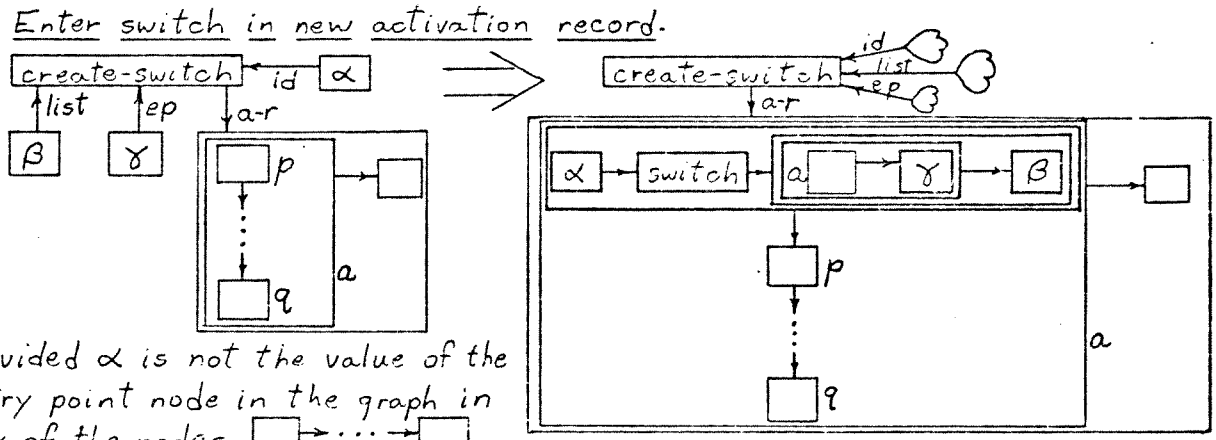
provided α is not the value of the entry point node in the graph in any of the nodes

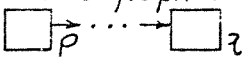
24. Retrieve own variable and enter in new activation record.



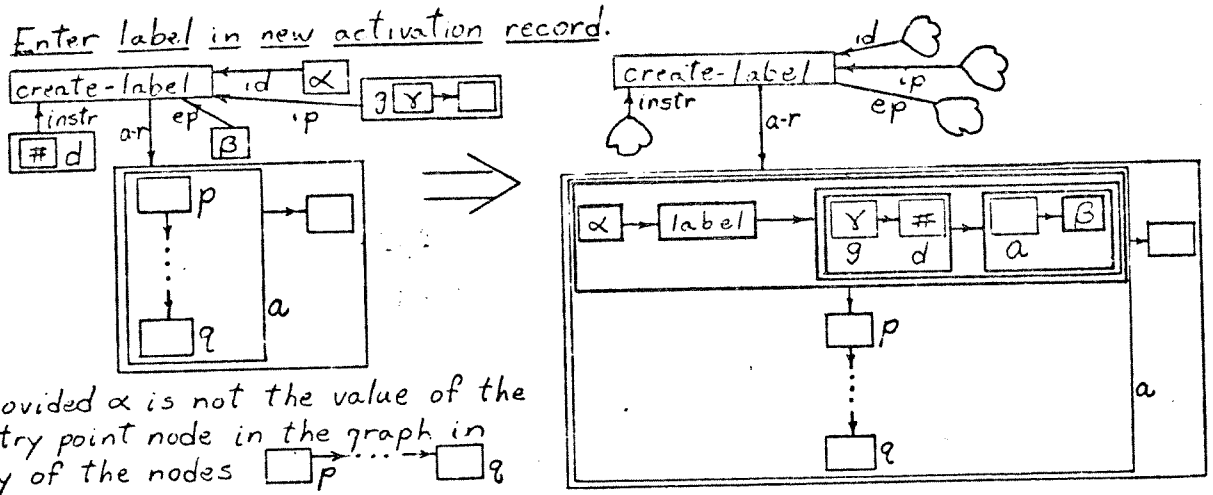
provided α is not the value of the entry point node in the graph in any of the nodes

25. Enter switch in new activation record.



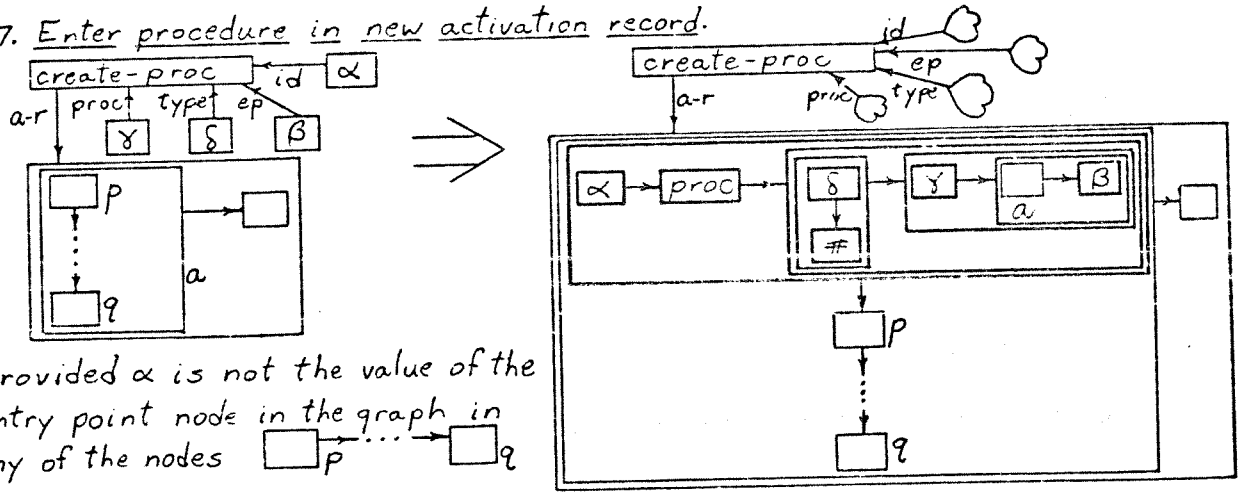
provided α is not the value of the entry point node in the graph in any of the nodes 

26. Enter label in new activation record.



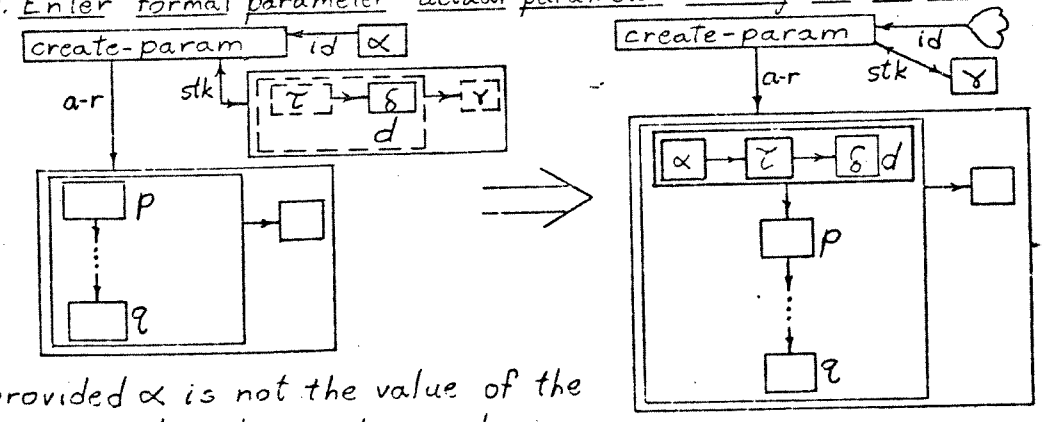
provided α is not the value of the entry point node in the graph in any of the nodes

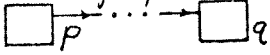
27. Enter procedure in new activation record.



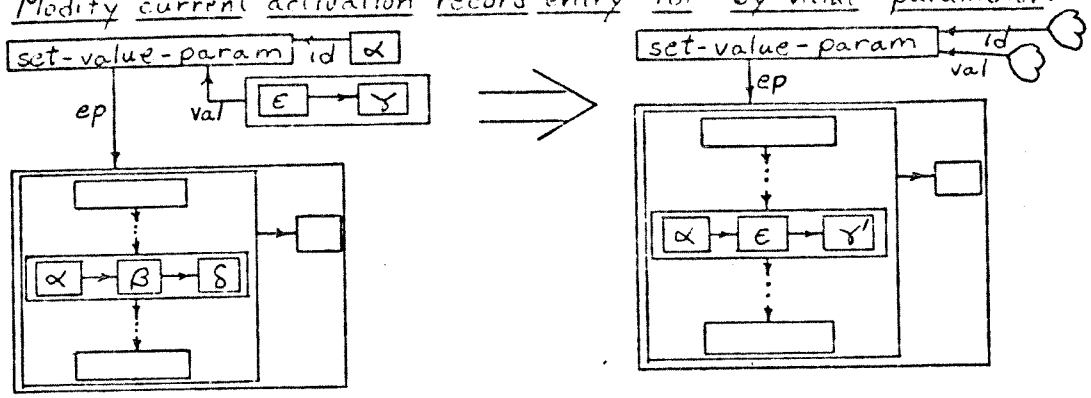
provided α is not the value of the entry point node in the graph in any of the nodes

28. Enter formal parameter - actual parameter binding in new activation record.



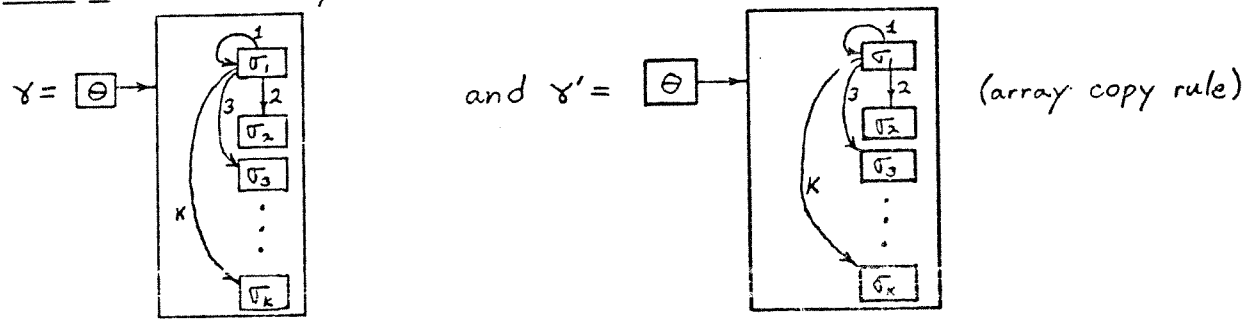
provided α is not the value of the entry point node in the graph in any of the nodes 

29. Modify current activation record entry for "by-value" parameter.



Case 1. If $\epsilon \in \{real, integ, Bool\}$ then $\gamma' = \gamma$.

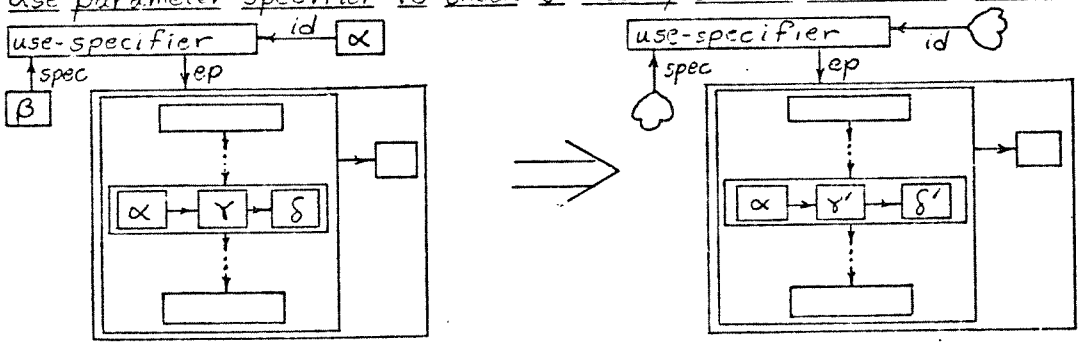
Case 2. If ϵ = array, then γ has the form:



(array copy rule)

Note: Labels, switches, procedures, and strings cannot be transmitted by value.

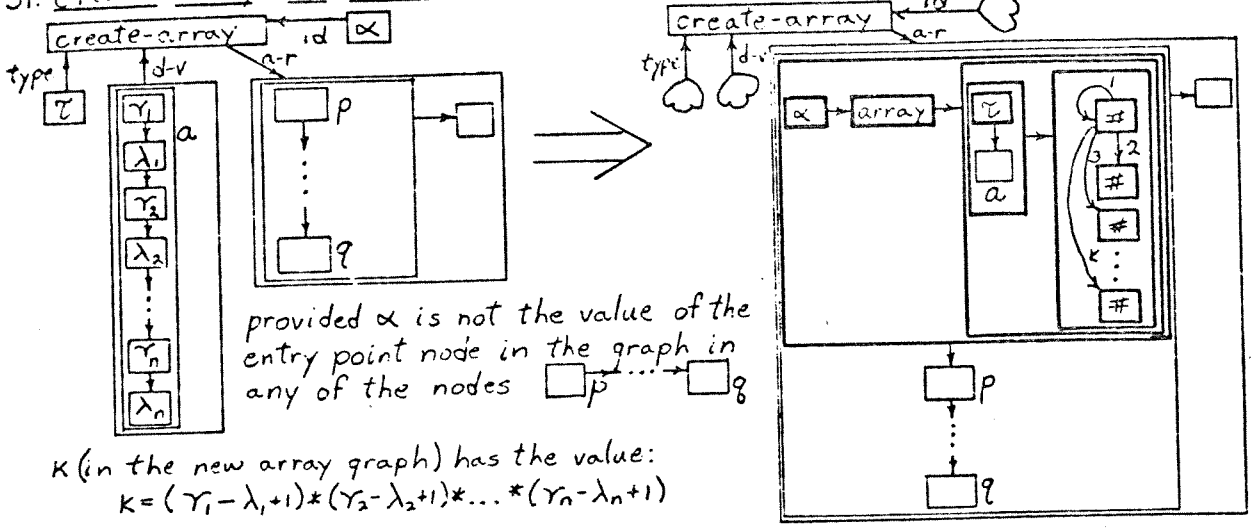
30. Use parameter specifier to check or modify current activation record entry.



Case 1. If $\gamma, \beta \in \{real, integ\}$ and $\gamma \neq \beta$ then $\gamma' = \beta$ and $\delta' = \delta$ converted to type β.

Case 2. If γ = name-param, or $\gamma = \beta$, or $\gamma \in \{array, proc\}$ and β = real γ, integer γ, or Boolean γ, then $\gamma' = \gamma$ and $\delta' = \delta$ (no change).

31. Create array and enter in new activation record.

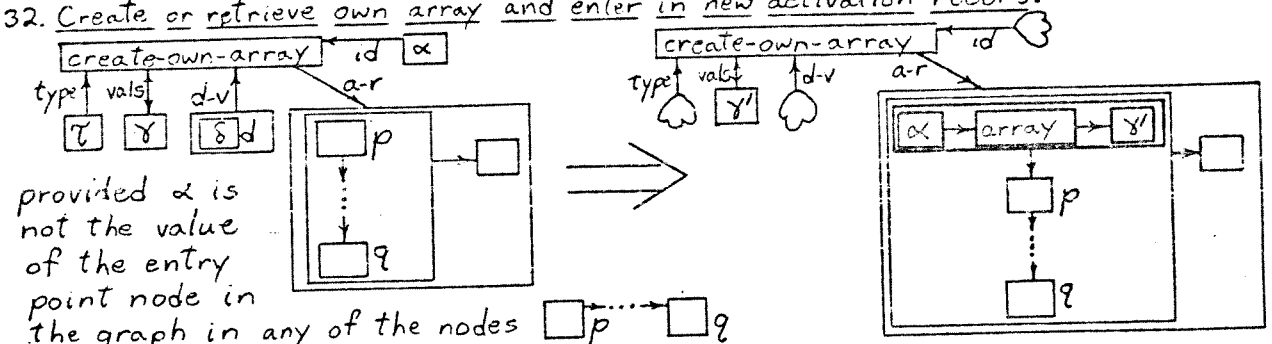


provided α is not the value of the entry point node in the graph in any of the nodes p ··· q

K (in the new array graph) has the value:

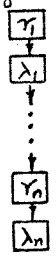
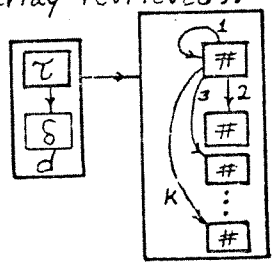
$K = (\gamma_1 - \lambda_1 + 1) * (\gamma_2 - \lambda_2 + 1) * \ldots * (\gamma_n - \lambda_n + 1)$

32. Create or retrieve own array and enter in new activation record.



provided α is not the value of the entry point node in the graph in any of the nodes p ··· q

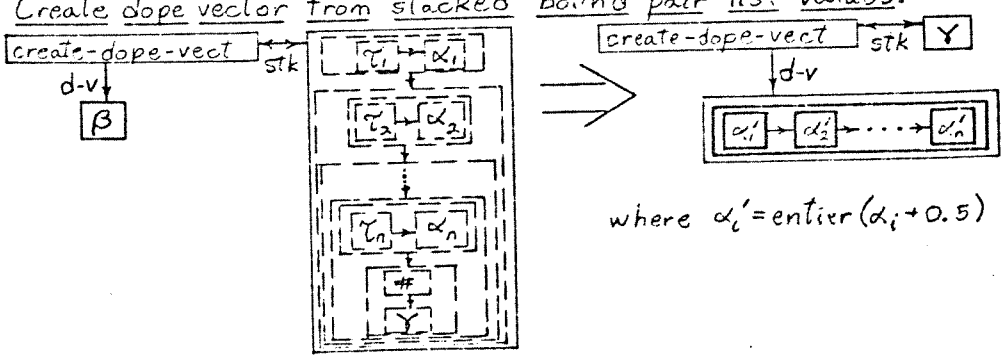
Case 1. If γ ≠ # then γ' = γ (old array retrieved).

Case 2. If γ = # and δ =  then γ' = 

where $K = (\gamma_1 - \lambda_1 + 1) * (\gamma_2 - \lambda_2 + 1) * \ldots * (\gamma_n - \lambda_n + 1)$.

(new array created)

33. Create dope vector from stacked bound pair list values.



where $\alpha_i' = \text{entier}(\alpha_i + 0.5)$, $1 \le i \le n$

VI. BIBLIOGRAPHY

1. Evans, A. An Algol 60 Compiler. Annual Review in Automatic Programming, Vol. 4, 1964, 87-124.

2. Feldman, J. A. A Formal Semantics for Computer Languages and its Application in a Compiler-compiler. Comm. ACM 9 (Jan 1966), 3-9.

3. Feldman, J. and Gries, D. Translator Writing Systems. Comm. ACM 13 (Feb 1968), 77-113.

4. Floyd, R. W. A Descriptive Language for Symbol Manipulation. J. ACM 8 (Oct 1961), 579-584.

5. Floyd, R. W. Syntactic Analysis and Operator Precedence. J. ACM 10 (July 1963), 316-333.

6. Friedman, D. and Slocum, J. GROPE Manual. In preparation at University of Texas.

7. Gries, D. Compiler Construction for Digital Computers. John Wiley and Sons, Inc. New York, N. Y., 1971.

8. McClure, R. M. TMG--A Syntax-directed Compiler. Proc. ACM 20th Natl. Conf., 1965, 262-274.

9. McKeeman, W. M., Horning, J. J., and Wertman, D. B. A Compiler Generator, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1970.

10. Plaskow, J. and Sherman, S. The TRANGEN System on the M460 Computer. AFCRL-66-516 (July 1966).

11. Pratt, T. W. Pair Grammars, Graph Grammars, and String-to-Graph Translation. J. Comp. and Sys. Sciences, Vol. 5, No. 6 (Dec 1971), 550-595.

12. Pratt, T. W. A Formal Definition of Algol 60 Using Hierarchical Graphs and Pair Grammars. Univ. of Texas Computation Center, Austin, Texas, 1973.

13. Pratt, T. W. Programming: Structures and Languages. Prentice-Hall, Englewood Cliffs, N. J. (to appear).

14. Reynolds, J. C. An Introduction to the COGENT Programming System. Proc. ACM 20th Natl. Conf., 1965, 422-436.

15. Schneider, F. W. and Johnson, G. D. META-3; A Syntax-directed Compiler-writing Compiler to Generate Efficient Code. Proc. ACM 19th Natl. Conf., 1964,D1.5-1.

16. Wirth, N. and Weber, H. EULER--A Generalization of Algol, and its Formal Definition: Parts I and II. Comm. ACM 9 (Jan, Feb 1966) 13-25, 89-99.

17. Naur, P., et al. Revised Report on the Algorithmic Language Algol 60. International Federation for Information Processing, 1962.