CGTM 95
July 1970
Charles Holbrow

FCONC: A Faster Concordance Generator which Reads Input Text from Tape

A previous memo, CGTM #91, described a simple selective concordance generator, CONC, written in SNOBOL4. An important feature of CONC is its buffered input, necessary because the text to be analyzed was read in on cards containing no more than 72 characters each. A string was used as the buffer by successively concatenating the contents of 69 cards into a single string nearly 5000 characters long. The string was then scanned by the concordance generator to within ~150 characters of its end. All but this last ~150 characters were thrown away and a new string was constructed from the last ~150 characters and the contents of the next 68 cards. String construction and scanning were repeated until no more input text was available.

Although effective, this approach is time consuming. To fill the buffer string requires construction of 68 intermediate strings which are destroyed almost at once. Construction of so many intermediate strings has two serious disadvantages. SNOBOL4 processes long strings slowly, so that as the buffer string becomes nearly full, the time to manipulate it becomes unduly long. Also formation of so many intermediate strings forces frequent storage regeneration and lengthens the time devoted to dynamic storage allocation.

The need for intermediate strings can be eliminated by filling the buffer string in a single step. This goal is achieved by writing the input text on tape, blocked almost to the size of the buffer string. CONC is easily modified to read a single buffer-sized block at a time. In this way the number of strings formed by CONC is reduced by a factor of 68, and the overall running time is reduced by almost 20 percent.

Figure 1 presents a simple SNOBOL4 program to move card images 72 characters long onto tape, delete trailing blanks, and reblock them in block and record lengths of 4968 bytes. The program assumes the tape being written is logical unit FT08F001.

```
-LIST LEFT                                                          00000040
  INPUT('INPUT',5,72)                                               00000080
        OUTPUT('OUT',8,'(24(20A1))')                                00000100
RPT     GT( (4968 - SIZE(A)),71)                    :S(AGN)         00000200
        OUT = A                                                     00000300
        A =                                                         00000400
AGN     A = A TRIM(INPUT) ' '                       :S(RPT)F(OUT)   00000500
OUT     OUT = A                                                     00000600
        ENDFILE(8)                                                  00000700
END                                                                 00000800
```

Fig. 1: A program to read text from cards onto tape in blocks of 4968 bytes.

The reblocked data can then be processed by a slightly modified version of CONC called FCONC. FCONC is shown in Figure 2 along with JCL appropriate for the SLAC 360/91. Note especially the need to use the 'R = 25000' parameter in the EXEC card. FCONC is basically the same as CONC but has had the comment cards removed (for these see CGTM 91) and has had the other changes indicated in Table 1.

Because 4968 plus the last ~150 characters of the buffer exceed 5000, the maximum permissible string length in FCONC was increased to 6000 by insertion of &MAXLNGTH = 6000. The statement numbers of CONC given in the table are those of the listing in the appendix of CGTM 91.

Table 1. Changes to convert CONC to FCONC

CONC		Replaced by	FCONC
Statement#	Statement	Statement#	Statement
------	------	200	&MAXLNGTH = 6000
900	INPUT(.IN,5,72)	300	INPUT(.IN,8,4968)
13600	LL = LT(LL,68) LL + 1 :S(RT1)	deleted	
13900	LL = 1	deleted	

To process a 15600 word sample with FCONC takes only 48.50 CPU sec compared to 59.27 CPU sec for CONC, a better than 18% reduction.

```
//JOBLIB DD DSN=SYS2.PROGLIB,DISP=(SHR,PASS),UNIT=2314                    00000200
// EXEC PGM=SNOBOL4,PARM='R=25000'                                        00000300
//FT06F001 DD SYSOUT=A,DCB=(LRECL=133,BLKSIZE=3458,BUFNO=2,RECFM=FBA),    00000500
//     SPACE=(TRK,(100,50))                                               00000600
//FT08F001 DD DSN=DCGMA,DCB=(LRECL=4968,BLKSIZE=4968,RECFM=FB,DEN=2),    00000640
//     UNIT=TAPE9,VOL=SER=CHH001,LABEL=(12,SL)                            00000680
//FT05F001 DD *                                                          00000700
-LIST LEFT                                                              00000800
 INPUT('INPUT',5,72)                                                    00000900
         DEFINE('INSERT(STR,T2,S1,S2)')
         &MAXLNGTH = 6000
         INPUT(.IN,8,4968)
         OUTPUT('PAGE',6,'(A1)')
         &ANCHOR = 1
         BLANKS = '                                                                '
         Q = 'RT5'
         T = ARRAY(500)
         DATA('LIST(NODE,LINK)')
         B2 = BREAK(' -') SPAN(' "-(')
         B = BREAK(' ') SPAN(' ')
         STR1 = '* * * * * * * * '
         P = TRIM(INPUT)
         N = 1
INP1     L1 = LT(L1,P) L1 + 1                                  :F(PAT)
         STR = TRIM(INPUT) ' '
         B1 = STR
RPT1     STR (BREAK(' ') . T<N> SPAN(' ')) =                   :F(RPT4)
         $T<N> = B1
         N = N + 1                                             :(RPT1)
RPT4     $('N' $T<N - 1>) = LIST($T<N - 1>)
         $('H' $T<N - 1>) = $('N' $T<N - 1>)                   :(INP1)
PAT      M = N - 1
         PAT = T<1>
         N = 1
RPT2     N = LT(N,M) N + 1                                     :F(RT0)
         PAT = PAT | T<N>                                      :(RPT2)
RT0      P1 = SUCCEED TAB(*STRT) (@S1 (B | (*GT(S2,SS) REM) $ STR1
.    ABORT) @STRT B B B B B B2 @S2 (PAT $ T2)   B B B B B B B B) $ STR
.    *INSERT(STR,T2,S1,S2)
RT5      STRT =
RT1      STR1 = STR1 TRIM(IN) ' '                              :F(RT4)
         SS = SIZE(STR1) - 104                                 :(RT2)
RT4      STR1 = STR1 '* * * * * * * * '
         Q = 'OUT'
         SS = SIZE(STR1) - 18
RT2      &FULLSCAN = 1
RT7      STR1 P1                                          :F($Q)S(RT7)
OUT      N = 1
         &FULLSCAN = 0
RPT6     N = N + R
         R = LE(N,M)          :F(END)
         PAGE = '1'
         $('N' $T<N>) = $('H' $T<N>)
RPT5     OUTPUT = NODE($('N' $T<N>))
         OUTPUT =
         $('N' $T<N>) = LINK($('N' $T<N>))
         IDENT($('N' $T<N>))                                   :F(RPT5)
RPT7     $T<N> B =                                            :F(RPT6)
         R = R + 1                                      :(RPT7)

INSERT   RR = 60 + S1 - S2
RT6      RR = LT(RR,0) 0
         BLANKS LEN(RR) . LPAD
         STR = LPAD STR
         LINK($('N' $T2)) = LIST(STR)
         $('N' $T2) = LINK($('N' $T2))                        :(RETURN)
END
```

Fig. 2: FCONC: A modified version of CONC which reads text from tape in blocks of 4968 bytes. The select terms must still be read in from cards.