SETL Newsletter # 114

Abraham Getzler
August 17, 1973

A SETLB to PUBLICATION SETL TRANSLATOR

As a byproduct of Malcolm Harrison's Artificial Intelligence Algorithms Project, there's now available on the 6600 a working program to turn SETLB source decks into publication SETL manuscripts.

The output of the program should be listed using Malcolm Harrison's deluxe teletype to take advantage of its lower case capabilities, backspacing, underlining, and extended character set.

Use of this program on working SETLB decks is expected to result in:

Typo-free published algorithms
Lower typist turnover
More stringently tested algorithms.

To obtain complete details of the process, run this job:

```
ID,cm4ØØØØ,t5. your name
cimget(AIAl2Øl=A)
-EOF-
```

This scheme would have been impossible were it not for C. Hornback's improvements to Harrison's teletype interface.

```
    ** CONSTITUENTS = CONSTTS **
    ** BUILDDAUGHTER = BILDAWT **
    ** NEWOPTION = NEWOPTN **
    **TESTFORWORDMATCH = T4WDMAT**

/*                    ALGORITHM 4                                          */
/*                                                                         */
/*     TOP - DOWN PARSER WITH SAVING                                       */
/*          ... A COMBINATION OF ALGORITHMS 1 + 2                          */
/*     GRAMMAR AS USUAL                                                    */
/*     TREE: TWO DIMENSIONAL ARRAY                                         */
/*          FIRST SUBSCRIPT = NODE NUMBER                                  */
/*          SECOND SUBSCRIPT =                                             */
/*               ≠NAME≠     ELEMENT IS NAME OF NODE                       */
/*               ≠PARENT≠   NUMBER OF PARENT NODE, =0 FOR ROOT            */
/*             ≠DAUGHTERS≠  N-TUPLE OF NUMBERS OF DAUGHTER NODES          */
/*               ≠WFS≠      NUMBER WELL-FORMED SUBSTRING MATCHING NODE    */
/*          ≠CURRENT OPTION≠  N-TUPLE OF CURRENT PRODUCTION               */
/*          ≠ALTERNATIVES≠  EITHER  SET OF N-TUPLES OF PRODUCTIONS       */
/*                                  NOT YET TRIED FOR THIS NODE          */
/*                          OR   SET OF NUMBERS OF WELL-FORMED           */
/*                               SUBSTRINGS NOT YET TRIED FOR THIS       */
/*                               NODE                                    */
/*               ≠FW≠       NUMBER OF FIRST WORD SPANNED BY NODE         */
/*                                                                       */
/* WFS (WELL-FORMED SUBSTRINGS): TWO DIMENSIONAL ARRAY                     */
/*     FIRST SUBSCRIPT = SUBSTRING NUMBER                                  */
/*     SECOND SUBSCRIPT =                                                  */
/*               ≠NAME≠          ELEMENT IS NAME OF SUBSTRING              */
/*               ≠FW≠            ELEMENT IS NUMBER OF FIRST WORD           */
/*                               SPANNED BY SUBSTRING                      */
/*               ≠LWP1≠          ELEMENT IS NUMBER OF LAST WORD            */
/*                               SPANNED BY SUBSTRING + 1                  */
/*          ≠CONSTITUENTS≠       ELEMENT IS N-TUPLE OF NUMBERS OF          */
/*                               IMMEDIATE CONSTITUENT SUBSTRINGS          */
/*                                                                         */
/* EXPANDED: TWO DIMENSIONAL ARRAY.                                        */
/*     FIRST SUBSCRIPT = NAME OF NON-TERMINAL SYMBOL                       */
/*     SECOND SUBSCRIPT = NUMBER OF SENTENCE WORD                          */
/*     VALUE = TRUE IF ALL WFS FOR THIS SYMBOL STARTING AT THIS WORD       */
/*          HAVE BEEN BUILT                                                */
/*                                                                         */
DEFINE PARSE(GRAMMAR,SENTENCE);
    /* INITIALIZATION */
    TREE = NL.;
    TREE(1,≠NAME≠) = ROOT;
    TREE(1,≠PARENT≠) = 0;
    <NODES,CURRENT,WORD,WFSS,JUNK> = <1,1,1,0,0>;
    EXPANDED = NL.;
    WFS = NL.;
TESTFORWORDMATCH:
    TREE(CURRENT,≠FW≠) = WORD;
    IF EXPANDED(TREE(CURRENT,≠NAME≠),WORD) EQ. T.
    THEN TREE(CURRENT,≠ALTERNATIVES≠) = ≤ S. 1<=S<=WFSS *
              (WFS(S,≠NAME≠) EQ. TREE(CURRENT,≠NAME≠))  AND.
              (WFS(S,≠FW≠) EQ. WORD) ≥;
         GOTO NEWWFS;
    ELSE IF GRAMMAR≤TREE(CURRENT,≠NAME≠)≥ EQ. NL.
         THEN IF WORD GT. *SENTENCE
```

```
            THEN GOTO FAIL;
            ELSE IF SENTENCE(WORD) EQ. TREE(CURRENT,≠NAME≠)
                 THEN WORD = WORD + 1;  GOTO MATCH;
                 ELSE GOTO FAIL;
            END IF WORD;
       ELSE TREE(CURRENT,≠ALTERNATIVES≠) =
                 GRAMMAR≤TREE(CURRENT,≠NAME≠)≥;
            TREE(CURRENT,≠DAUGHTERS≠) = NULL,;
  END IF EXPANDED;
NEWOPTION:
  JUNK = ARB, TREE(CURRENT,≠ALTERNATIVES≠);
   TREE(CURRENT,≠ALTERNATIVES≠) = TREE(CURRENT,≠ALTERNATIVES≠) LESS,
                                                          JUNK;
  TREE(CURRENT,≠CURRENT OPTION≠) = JUNK;
BUILDDAUGHTER:
  NODES = NODES + 1;
       /* GET NEXT ELEMENT OF CURRENT OPTION */
  TREE(NODES,≠NAME≠) = TREE(CURRENT,≠CURRENT OPTION≠)
                            (1+↓TREE(CURRENT,≠DAUGHTERS≠));
  TREE(NODES,≠PARENT≠) = CURRENT;
   TREE(NODES,≠DAUGHTERS≠) = NULL,;
  TREE(CURRENT,≠DAUGHTERS≠) = TREE(CURRENT,≠DAUGHTERS≠) + <NODES>;
  CURRENT = NODES;
  GOTO TESTFORWORDMATCH;
PASS:
  /* LOOK UP ONE LEVEL IN TREE */
  CURRENT = TREE(CURRENT,≠PARENT≠);
  /* HAVE ALL DAUGHTERS OF CURRENT NODE BEEN BUILT... */
  IF (↓TREE(CURRENT,≠DAUGHTERS≠))NE.(↓TREE(CURRENT,≠CURRENT OPTION≠))
       /* NO, ADD NEXT DAUGHTER */
       THEN GOTO BUILDDAUGHTER;
  END  IF(↓ TREE;
       /* YES, ADD WELL-FORMED SUBSTRING */
MATCH:      ADDWFS(TREE(CURRENT,≠NAME≠), TREE(CURRENT,≠FW≠), WORD,
                 [+: I→TREE(CURRENT,≠DAUGHTERS≠)]<TREE(I,≠WFS≠)>);
            TREE(CURRENT,≠WFS≠) = WFSS;
            /* IF NOT AT ROOT NODE GO UP AGAIN */
            IF CURRENT NE. 1 THEN GOTO PASS;
            /* IF WE HAVE MATCHED ALL WORDS IN SENTENCE, PRINT TREE*/
            ELSE IF WORD EQ. (1 + ↓SENTENCE) THEN PRTMAP(TREE);;
            /* TRY FOR ANOTHER PARSE */
FAIL:
  /* RESET WORD COUNT */
  WORD = TREE(NODES,≠FW≠);
  /* IF NODE WAS SAVED, CHECK FOR ALTERNATE WFS */
  IF EXPANDED(TREE(NODES,≠NAME≠),WORD) NE. OM.
       THEN CURRENT = NODES; GOTO NEWWFS;
       ELSE /* ALL WFS FOR THIS NODE HAVE BEEN BUILT */
            EXPANDED(TREE(NODES,≠NAME≠),WORD) = T.;
  END IF EXPANDED;
  /* DETACH LAST NODE ATTACHED */
DETACH:
  CURRENT = TREE(NODES,≠PARENT≠);
  /* IF AT ROOT NODE, RETURN */
  IF CURRENT EQ. 0 THEN RETURN;;
  TREE(CURRENT,≠DAUGHTERS≠) = TREE(CURRENT,≠DAUGHTERS≠)
                                 (1: ↓TREE(CURRENT,≠DAUGHTERS≠)-1);
  NODES = NODES - 1;
       /* IF WE HAVE NOT DETACHED A FIRST-IN-LEVEL, LOOP */
  IF TREE(CURRENT,≠DAUGHTERS≠) NE. NULL, THEN GOTO FAIL;;
```

```
    /* IF THERE ARE NO MORE OPTIONS FOR CURRENT NODE, DETACH IT */
    IF TREE(CURRENT,≠ALTERNATIVES≠) EQ, NL, THEN GOTO FAIL;
                                            ELSE GOTO NEWOPTION;;
NEWWFS;
    /* PICK NEXT WFS FROM ALTERNATES */
    IF TREE(CURRENT,≠ALTERNATIVES≠) EQ, NL, THEN GOTO DETACH;;
    JUNK = ARR, TREE(CURRENT,≠ALTERNATIVES≠);
     TREE(CURRENT,≠ALTERNATIVES≠) = TREE(CURRENT,≠ALTERNATIVES≠) LESS,
                                                              JUNK;
     TREE(CURRENT,≠WFS≠) = JUNK;
    /* ADVANCE  WORD  PAST END OF WFS */
    WORD = WFS(TREE(CURRENT,≠WFS≠), ≠LWP1≠);
    GOTO PASS;
END PARSE;

DEFINE ADDWFS(NAME,FW,LWF1,CONSTITUENTS);
    WFSS = WFSS + 1;
    WFS(WFSS,≠NAME≠) = NAME;
    WFS(WFSS,≠FW≠) = FW;
    WFS(WFSS,≠LWP1≠) = LWP1;
    WFS(WFSS,≠CONSTITUENTS≠) = CONSTITUENTS;
    RETURN;
END ADDWFS;

     GRAMMAR = NL,;
     GRAMMAR≤≠E$≠≥ = ≤:<≠T$≠>,<≠T$≠,≠+≠,≠E$≠>≥; /* EXPRESSION */
     GRAMMAR≤≠T$≠≥ = ≤:<≠F$≠>,<≠F$≠,≠*≠,≠T$≠>≥; /* TERM */
     GRAMMAR≤≠F$≠≥ = ≤:<≠(≠,≠E$≠,≠)≠>,<≠N≠>≥;  /* FACTOR */
     ROOT = ≠E$≠;

     SENTENCE = <≠N≠, ≠*≠, ≠N≠ > ;

     PARSE(GRAMMAR,SENTENCE);

    PRTMAP(TREE);  PRTMAP(WFS);
COMPUTE; FINISH;
```

```
        ¢  this is a translation of setlb text

DO;
     macro  CONSTITUENTS = CONSTTS  endm;
     macro  BUILDDAUGHTER = BILDAWT  endm;
     macro  NEWOPTION = NEWOPTN  endm;
     macro TESTFORWORDMATCH = T4WDMAT endm;

¢                    algorithm 4
¢
¢     top - down parser with saving
¢          ... a combination of algorithms 1 + 2
¢     grammar as usual
¢     tree: two dimensional array
¢          first subscript = node number
¢          second subscript =
¢               'name'     element is name of node
¢               'parent'   number of parent node, =0 for root
¢             'daughters'  n-tuple of numbers of daughter nodes
¢               'wfs'      number well-formed substring matching node
¢          'current option'  n-tuple of current production
¢          'alternatives'  either  set of n-tuples of productions
¢                                  not yet tried for this node
¢                          or  set of numbers of well-formed
¢                              substrings not yet tried for this
¢                              node
¢               'fw'       number of first word spanned by node
¢
¢ wfs (well-formed substrings): two dimensional array
¢     first subscript = substring number
¢     second subscript =
¢               'name'          element is name of substring
¢               'fw'            element is number of first word
¢                               spanned by substring
¢               'lwp1'          element is number of last word
¢                               spanned by substring + 1
¢          'constituents'       element is n-tuple of numbers of
¢                               immediate constituent substrings
¢
¢ expanded: two dimensional array
¢     first subscript = name of non-terminal symbol
¢     second subscript = number of sentence word
¢     value = true if all wfs for this symbol starting at this word
¢          have been built
¢
define PARSE(GRAMMAR,SENTENCE);
    ¢ initialization
    TREE = nl;
    TREE(1,'NAME') = ROOT;
    TREE(1,'PARENT') = 0;
    <NODES,CURRENT,WORD,WFSS,JUNK> = <1,1,1,0,0>;
    EXPANDED = nl;
    WFS = nl;
```

```
    TREE(CURRENT,'FW') = WORD;
    if EXPANDED(TREE(CURRENT,'NAME'),WORD) eq t
    then TREE(CURRENT,'ALTERNATIVES') = ( S, 1≤S≤NFSS :
              (NFS(S,'NAME') eq TREE(CURRENT,'NAME'))  and
              (NFS(S,'FW') eq WORD) );
         goto NEWNFS;
    else if GRAMMAR(TREE(CURRENT,'NAME')) eq nl
         then if WORD gt #SENTENCE
              then goto FAIL;
              else if SENTENCE(WORD) eq TREE(CURRENT,'NAME')
                   then WORD = WORD + 1;  goto MATCH;
                   else goto FAIL;
              end if WORD;
         else TREE(CURRENT,'ALTERNATIVES') =
                   GRAMMAR(TREE(CURRENT,'NAME'));
              TREE(CURRENT,'DAUGHTERS') = nult;
    end if EXPANDED;
NEWOPTION:
    JUNK = arb TREE(CURRENT,'ALTERNATIVES');
     TREE(CURRENT,'ALTERNATIVES') = TREE(CURRENT,'ALTERNATIVES') less
                                                              JUNK;
    TREE(CURRENT,'CURRENT OPTION') = JUNK;
BUILDDAUGHTER:
    NODES = NODES + 1;
         ¢ get next element of current option
    TREE(NODES,'NAME') = TREE(CURRENT,'CURRENT OPTION')
                              (1+#TREE(CURRENT,'DAUGHTERS'));
    TREE(NODES,'PARENT') = CURRENT;
     TREE(NODES,'DAUGHTERS') = nult;
    TREE(CURRENT,'DAUGHTERS') = TREE(CURRENT,'DAUGHTERS') + <NODES>;
    CURRENT = NODES;
    goto TESTFORWORDMATCH;
PASS:
    ¢ look up one level in tree
    CURRENT = TREE(CURRENT,'PARENT');
    ¢ have all daughters of current node been built...
    if (#TREE(CURRENT,'DAUGHTERS'))ne(#TREE(CURRENT,'CURRENT OPTION'))
         ¢ no, add next daughter
         then goto BUILDDAUGHTER;
    end  if(# TREE;
         ¢ yes, add well-formed substring
MATCH:        ADDWFS(TREE(CURRENT,'NAME'), TREE(CURRENT,'FW'), WORD,
                   [+: I∈TREE(CURRENT,'DAUGHTERS')]<TREE(I,'WFS')>);
              TREE(CURRENT,'WFS') = WFSS;
              ¢ if not at root node go up again
              if CURRENT ne 1 then goto PASS;
              ¢ if we have matched all words in sentence, print tree
              else if WORD eq (1 + #SENTENCE) then PRTMAP(TREE);;
              ¢ try for another parse
FAIL:
    ¢ reset word count
    WORD = TREE(NODES,'FW');
    ¢ if node was saved, check for alternate wfs
    if EXPANDED(TREE(NODES,'NAME'),WORD) ne om
         then CURRENT = NODES; goto NEWWFS;
         else ¢ all wfs for this node have been built
              EXPANDED(TREE(NODES,'NAME'),WORD) = t;
    end if EXPANDED;
    ¢ detach last node attached
DETACH:
```

```
        CURRENT = TREE(NODES,'PARENT');
        ¢ if at root node, return
        if CURRENT eq 0 then return;;
        TREE(CURRENT,'DAUGHTERS') = TREE(CURRENT,'DAUGHTERS')
                                          (1: #TREE(CURRENT,'DAUGHTERS')-1);
        NODES = NODES - 1;
             ¢ if we have not detached a first-in-level, loop
        if TREE(CURRENT,'DAUGHTERS') ne null then goto FAIL;;
        ¢ if there are no more options for current node, detach it
        if TREE(CURRENT,'ALTERNATIVES') eq nl then goto FAIL;
                                                  else goto NEWOPTION;;
    NEWWFS:
        ¢ pick next wfs from alternates
        if TREE(CURRENT,'ALTERNATIVES') eq nl then goto DETACH;;
        JUNK = arb TREE(CURRENT,'ALTERNATIVES');
         TREE(CURRENT,'ALTERNATIVES') = TREE(CURRENT,'ALTERNATIVES') less
                                                                    JUNK;
         TREE(CURRENT,'WFS') = JUNK;
        ¢ advance word past end of wfs
        WORD = WFS(TREE(CURRENT,'WFS'), 'LWPI');
        goto PASS;
    end PARSE;

    define ADDWFS(NAME,FW,LWPI,CONSTITUENTS);
        WFSS = WFSS + 1;
        WFS(WFSS,'NAME') = NAME;
        WFS(WFSS,'FW') = FW;
        WFS(WFSS,'LWPI') = LWPI;
        WFS(WFSS,'CONSTITUENTS') = CONSTITUENTS;
        return;
    end ADDWFS;

         GRAMMAR = nl;
         GRAMMAR('E$') = (<'T$'>,<'T$','+','E$'>); ¢ expression
         GRAMMAR('T$') = (<'F$'>,<'F$','*','T$'>); ¢ term
         GRAMMAR('F$') = (<'(','E$',')'>,<'N'>);  ¢ factor
         ROOT = 'E$';

         SENTENCE = <'N', '*', 'N' > ;

         PARSE(GRAMMAR,SENTENCE);

        PRTMAP(TREE);  PRTMAP(WFS);
    compute; finish;
```

```
COMMAND - LOGOUT.
YOU NO LONGER OWN ANY PRIVATE FILES.
CP TIME      .054
PP TIME    22.335
 08/16/73  OFF AT  16.18.50.
```