3-1-1980

Implementation of an LALR(1) Parser Generator

G. Svenheim

ROCHESTER INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

IMPLEMENTATION OF AN LALR(1) PARSER GENERATOR

A Thesis Submitted in Partial Fulfillment of

Master of Science in Computer Science Degree Program

BY: G. Svenheim

APPROVED BY:

G. Johnson

Name Illegible

Kenneth A. Reek

Name Illegible

DATE: March 1980

G-

Implementation of an LALR(1) Parser Generator

by

G. Svenheim

March 1980

Thesis Advisor: Mr. G. Johnson

Table of Contents

Implementation of an LALR(1) Parser Generator

Introduction

The purpose of this thesis was to implement an LALR(1) parser generator using the algorithm and methods presented in Aho and Ullman (1977). Generally speaking this meant to input the definition of a LR(1) grammar and output tables that could be used by a parser to decide whether or not arbitrary sentences from the grammar are syntactically correct.

This paper is organized into the following sections:

I. Background. A brief summary of languages, grammar, parsing, and parsing techniques is presented.

II. LALR(1) Parser Generator. A review of the method suggested by Aho and Ullman (1977).

III. LALR(1) Parser Generator and Parser Overview. The Overview shows the interaction between the parser generator and the parser.

IV. LALR(1) Parser Generator. A detailed description of the procedures used to implement the parser generator.

V. Shift-Reduce Parser. A detailed description of the parser implemented to test the parser generator tables.

VI. Sample runs of selected grammars through the parser generator and parser are presented and explained.

VII. Conclusion. Discussion of implementation problems encountered, possible extensions and concluding remarks.

I. Background

Languages

According to Cleaveland and Uzgalis (1977) a "language" is a defined set of strings. "Strings" are a sequence of symbols using the alphabet of the language.

Grammars

Grammars define languages. "Phrase structured" grammars are defined by Barrett and Couch (1979) as a four-tuple (Σ, N, P, S) where:

Σ Is the set of all terminal symbols within the grammar.

N Is the set of all nonterminal symbols within the grammar where Σ and N are disjoint.

P Is the set of all productions of the form $y \rightarrow x$ where y and x are in $(N \cup \Sigma)^*$ and y contains at least one element in N, and

S Is a designated start symbol in N.

Chomsky (1965) defined four classes of phrase structured grammars:

Type 3 which defines Regular languages,

Type 2 which defines Context Free languages,

Type 1 which defines Context Sensitive languages, and

Type 0 which defines Recursively Enumerable languages.

Each type of grammar defines a set of languages which is a proper subset of the set defined by any lower numbered grammar type. For example, Type 0 grammars can define all of the languages definable by Type 1, Type 2 or Type 3.

The difference between the four classes of grammar are explained below using the four-tuple mentioned above.

Type 3 grammars have productions of the form:

$A \rightarrow xB$ or
$A \rightarrow x$ where A and B are in N and x is in Σ^*.

Type 2 grammars have productions of the form:

$x \rightarrow y$ where x is a member of N and y is any string in $(N \cup \Sigma)^*$

Type 1 grammars have productions of the form:

$x \rightarrow y$ where x and y are members of $(N \cup \Sigma)^*$, x contains at least one member of N, and $|x| \leq |y|$.

Type 0 grammars have productions of the form:

$x \rightarrow y$ where x is a member of $(N \cup \Sigma)^+$ and y is a member of $(N \cup \Sigma)^*$.

It can be seen by examining the types of grammars that as we proceed from Type 3 to Type 0 the restrictions on the productions decrease thereby allowing greater freedom in the resultant languages and also increasing the complexity of the parsing problem.

Context free languages are the most practical because they strike a happy medium between generality and reasonable cost. In other words these languages can be general enough to look like plain english (i.e. PL/1 and ALGOL) yet restrictive enough so that practical compilers can be written for them. PL/1 and ALGOL are examples of context free languages although there are context sensitive elements in both these languages that are handled by other means than through the language definition.

One very important part of the compiler is the parser. It is the parser's job to check incoming sentences from the language for correct syntax. Does the string fulfill the requirements of the grammar?

As an example of the parsing problem consider the following grammar:

G: $(\{+,*,(,),a\}, \{E,T,F\}, P, E)$ where P is the set of productions:

1. $E \rightarrow E+T$
2. $E \rightarrow T$
3. $T \rightarrow T*F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow a$

a + a is a valid sentence in this language because it can be derived from the following productions:

$E \rightarrow E+T$	rule 1
$\rightarrow E+F$	rule 4
$\rightarrow E+a$	rule 6
$\rightarrow T+a$	rule 2
$\rightarrow F+a$	rule 4
$\rightarrow a+a$	rule 6

However, the sentence a+ is invalid because we can see by visual inspection that the terminal symbol + cannot be the last terminal in the sentence.

The parsing problem is to take the input sentences and determine if, by applying the correct sequence of productions, the sentence is valid for this grammar.

Three possible approaches to the parsing problem are:

1. Backtracking whereby we start with the first production and then resolve a nonterminal on the right part of the production arbitrarily by another production and keep on picking other productions until it is determined that an error has been made. At that point the parser must "backtrack" through the parse tree to try another path until either all paths have been exhausted or the sentence is accepted. Obviously this approach would be very time consuming.

2. Deterministic Top-Down Parser where the parser starts with the first production and by looking at the next input symbol can positively determine the next production to apply. This is also called a LL(1) parser where LL means Left to right scan of input sentence with Leftmost nonterminal resolved first.

3. Deterministic Bottom-up Parser where the parser starts with the input string and works "backward" through the productions to obtain the correct parse. One of the broad classes of bottom-up parsers is the LR(k) parser where LR means Left to Right scan of input with rightmost nonterminal resolved first. The (k) means that the parser will look at the next k symbols to decide which production to use. The LR(1) parser is a special case where the parser looks ahead one symbol to determine which production to use.

   Another name for this type of parsing is "shift-reduce" parsing and in this name lies the key to this technique. As the input string of terminals is examined the parser looks for any substring that matches the right part of any production (also called a "handle") in the grammar. Until a match is found the input terminals are shifted onto a stack. When the match is found then the stacked symbols are replaced with the left side of the production. The replacement process is called a reduction, hence the name shift-reduce parsing. The shifting and reducing continues until either: (a) the final reduction gives the start symbol in which case the input string is valid for this grammar or (b) no more reductions can be made and the start symbol has not been reached in which case the string is invalid for this grammar.

   Later on in this paper the complete algorithm for shift-reduce parsing is presented.

As summarized by Barrett and Couch (1979) the technique that is most general yet is able to detect a syntax error at the very earliest point is the last technique described above; the LR(1) parser.

This technique is very similar to the one outlined in Aho and Ullman (1977) except Aho and Ullman go one step beyond the LR(1) parser to the LALR(1) parser where LA stands for "lookahead". As it turns out LR(1) parsers are very expensive when it comes to space. The LALR(1) parser is a significant improvement over the LR(1) parser in this respect.

The original work done on LR(k) parsers is Knuth (1965). The k indicates the general case where k symbols are looked at before deciding which production to apply next.

II. LALR(1) Parser Generator

One method described in Aho and Ullman (1977) of obtaining the LALR(1) parsing tables is to construct the LR(1) parsing tables and then by examining the generated items reduce the tables to LALR(1) tables.

The key element of the LR(1) parsing table construction is to generate the LR(1) sets-of-items. According to Barrett and Couch (1979) an item is a production carrying a position marker and a lookahead symbol. Intuitively everything to the left of the marker has been recognized and the lookahead symbol represents a possible terminal grammar symbol that could appear next on the input stream for this production. The LR(1) sets-ot-items are a series of states that define the possible parser for the given grammar. As an input string is analyzed the parser moves through the various states to determine if the sentence is correct. As an example of an item consider the following:

    A -> B.CD,+

This is an item that references production:

    A -> BCD

and that the first symbol in this production, B has been recognized and that the lookahead symbol is a '+'. The lookahead can be any terminal symbol in the grammar or a '$' signifying that the end of the string has been reached.

The following procedures, taken from Aho and Ullman (1977), generate the LR(1) sets-of-items.

```
Procedure Main;

    begin
      C := {CLOSURE( {S' -> ·S,$} )};
      repeat
        for each set of items I in C and each grammar
            symbol X such that GOTO( I ,X) is not empty
            and not already in C do
                add GOTO( I ,X) to C
      until no more sets of items can be added to C
    end
    procedure CLOSURE( I );
    begin
      repeat
        for each item [A -> α·Bβ ,a] in I , each
            production B -> y, and each terminal b in FIRST(β a)
            such that [B -> ·y,b] is not in I do
                add [B -> ·y,b] to I ;
      until no more items can be added to I ;
      return
    end;

    procedure GOTO( I ,X);
    begin
      let J be the set of items [A -> αX·β ,a] , such that
         [A -> α·Xβ ,a] is in I;
      return CLOSURE(J)
    end;
```

The function FIRST (B) returns the set of terminals that begin strings derived from B.

Procedure CLOSURE calculates the transitive completion mentioned in Knuth (1965) and means intuitively that if item $A \rightarrow \cdot B\beta$ is in CLOSURE (I) then we would expect, at some point in the parsing process, to see a string derivable from $B\beta$. Therefore CLOSURE adds items to this set of the form $B \rightarrow .\gamma$.

The GOTO procedure calculates, for a given state and grammar symbol, the next state the parser should go to.

The following algorithm, extracted from Aho and Ullman (1977) (Algorithm 6.3 in their book) gives the method for constructing the LR(1) parsing tables GOTO and ACTION.

Construction of LR(1) parsing tables.

Input. A grammar G augmented by production $S' \rightarrow S$.

Output. If possible, the canonical LR parsing action function ACTION and goto function GOTO.

Method.

1. Construct $C = \{I_0, I_1, \ldots, I_n\}$, the collection of sets of LR(1) items for G.
2. State i of the parser is constructed from I_i. The parsing actions for state i are determined as follows:
   a) If $[A \rightarrow \alpha \cdot a\beta, b]$ is in I_i and GOTO$(I_i, a) = I_j$, then set ACTION $[i,a]$ to "shift j."
   b) If $[A \rightarrow \alpha \cdot, a]$ is in I_i, then set ACTION $[i,a]$ to "reduce $A \rightarrow \alpha$."
   c) If $[S' \text{--} S\cdot, \$]$ is in I_i, then set ACTION $[i,\$]$ to "accept."

If a conflict results from the above rules, the grammar is said not to be LR(1) and the algorithm is said to fail.

3. The goto transitions for state i are determined as follows: If GOTO$(I_i, A) = I_j$, then GOTO $[i,A] = j$
4. All entries not defined by rules (2) through (3) are made "error."
5. The initial state of the parser is the one constructed from the set containing item $[S' \rightarrow \cdot S, \$]$.

The action function ACTION and the goto function GOTO are two dimensional arrays where the first dimensions represents the states and the second dimension represents the grammar symbols. The elements of the goto function are states whereas the elements of the action function indicate the action that the parser is to take: shift, reduce by a production number, accept sentence, or error meaning the sentence cannot be parsed.

To obtain the LALR(1) parsing tables Aho and Ullman (1977) present the following algorithm (Algorithm 6.4 in their text).

Input. A grammar G augmented by production $S' \rightarrow S$.

Output. The LALR parsing tables ACTION and GOTO.

Method.

1. Construct $C = \{I_0, I_1, \ldots, I_n\}$, the collection of sets of LR(1) items.
2. For each core present among the sets of LR(1) items, find all sets having that core, and replace these sets by their union.
3. Let $C' = \{J_0, J_1, \ldots, J_m\}$ be the resulting sets of LR(1) items. The parsing actions for state i are constructed from J_i in the same manner as in Algorithm 6.3. If there is a parsing-action conflict, the algorithm fails to produce a parser, and the grammar is said not to be LALR(1).
4. The GOTO table is constructed as follows: If J is the union of one or more sets of LR(1) items, i.e., $J = I_1 \cup I_2 \cup \ldots \cup I_m$, then the cores of $GOTO(I_1,X)$, $GOTO(I_2,X), \ldots, GOTO(I_k,X)$ are the same, since $I_1, I_2, \ldots, I_k$ all have the same core. Let K be the union of all sets of items having the same core as $GOTO(I_1,X)$. Then $GOTO(J,X)=K$.

III. LALR(1) Parser Generator and Parser Overview

General Flowchart (see figure 1.)

The project, as developed, ended up with two distinct parts: The parser generator and a parser. The parser generator basically inputs a grammar definition and output five files:

1. A report file which describes the actions taken by the parser generator plus generates the action and goto tables.

2. Symbol table which defines the symbols in the grammar and the associated shorthand notations.

3. Production table which contains the internal representation of the productions.

4. The LALR size tables. This file contains the dimensions of the resultant LALR action and GOTO tables.

5. The LALR action and GOTO tables.

The parser was needed to test various grammar strings to see if they could be parsed correctly. Besides the four files generated from the Parser Generator (see figure 1) the parser has an input file of grammar strings to be tested and an output report file which gives the parse of the input strings or the partial parse if the string is not part of the grammar.

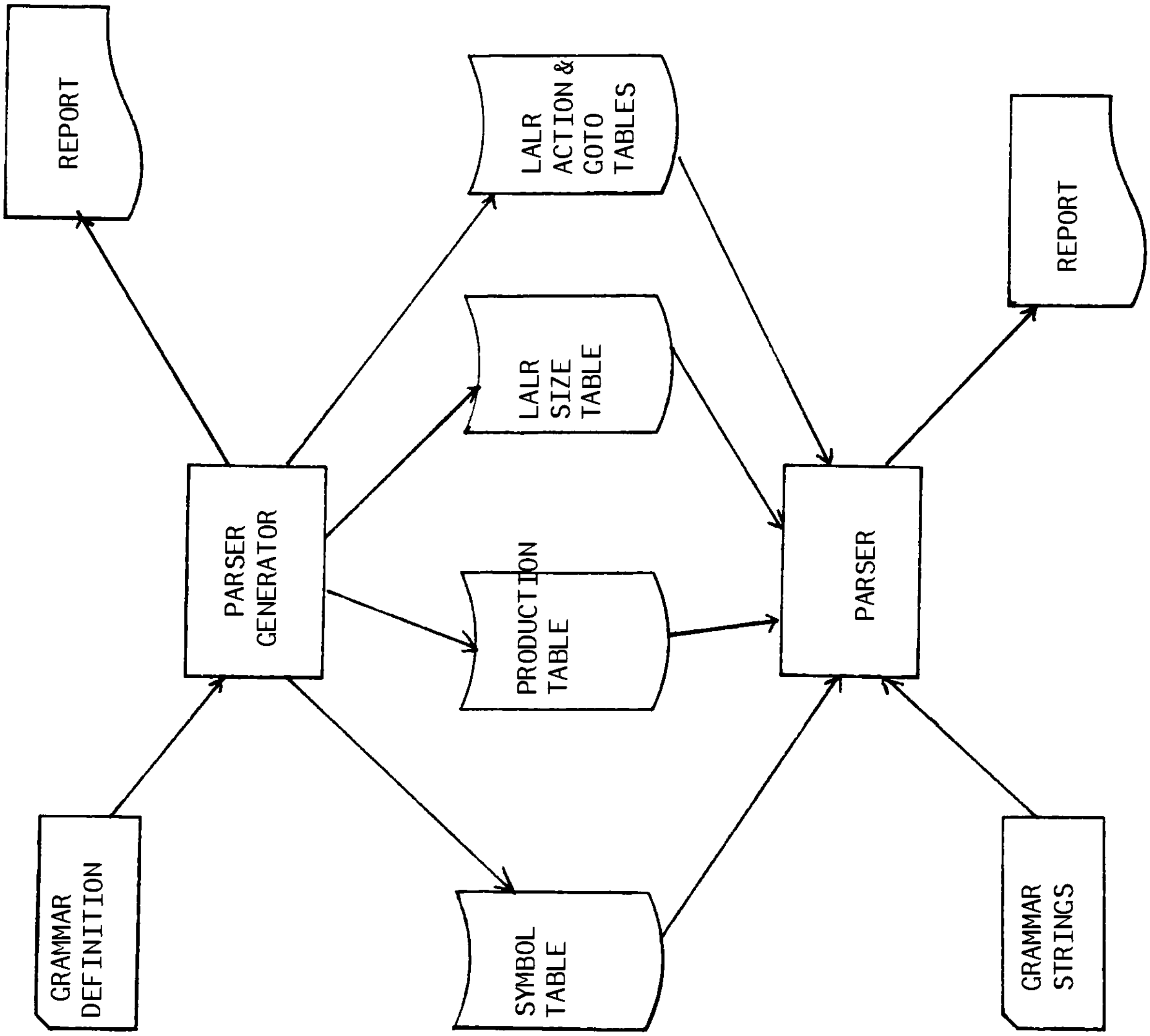
REPORT
LALR ACTION & GOTO TABLES
REPORT
LALR SIZE TABLE
PARSER GENERATOR
PRODUCTION TABLE
PARSER
GRAMMAR DEFINITION
SYMBOL TABLE
GRAMMAR STRINGS

Figure 1

IV. LALR(1) Parser Generator

Important Data Structures

A. SYMBOLS (Appendix D p.2)

An array where each element has the following structure:

1. 1 character that is internal representation of symbol.

2. 1 bit that indicates whether symbol is a terminal ('0') or a non-terminal ('1').

3. Fixed binary number that gives the length of the external representation of symbol.

4. Complete identifier (up to 30 characters) of external identifier.

B. PROD (Appendix D p.2)

An array where each element is a production in the grammar and has the following structure:

1. The number of this production. Numbering starts from one and increases by one for each production defined in the grammar. The next element after the last production has this field set to zero as a delimiter.

2. One character (internal representation) that is the left part of the production.

3. A fixed binary number that is the number of symbols in the right part of the production.

4. Up to six characters of the right part of the production. Each character is the internal representation of a grammar symbol.

C. ITEMS (Appendix D p.2)

This array stores the generated items. Each element of the array has the following structure:

1. STATE is a fixed binary number that is the state number of the generated item.

2. NEXT which is a fixed binary number that is the element number in the ITEMS array of the next item in this state. A -1 indicates that this is the last item in this state.

3. PROD is a fixed binary number that references the production (in PROD array) that this item has been generated from.

4. POSIT is a fixed binary number that indicates the recognized part of this item.

5. LA is one character that is the look ahead symbol.

D. FIRSTS (Appendix D p.2)

This array contains, for each grammar symbol, the possible terminal symbols that can derive the object symbol. The symbols in this table are in the internal format. Each element of this array has the following structure.

1. NO_FIRSTS which is a binary number that is the number of first symbols contained in the next element.

2. FIRSTS which is in itself an array where each element is a possible FIRST symbol of the symbol in question.

Notice that this array does not have the object symbol itself stored. That is because the relative position of each symbol in array SYMBOLS (see above) is the same for array FIRSTS.

E. STATE_ITEMS (Appendix D p.3)

This array contains the number of generated items in each state. This array speeds up the process of determining identically equal states. By checking this array two states can be checked for equality in number of elements before checking for identically equal states.

F. GOTOS (Appendix D p.3)

This array has two dimensions; one for the grammar symbols and one for the parser states. When the parser does a shift action it then references this array to determine the next state to go to.

G. ACTION (Appendix D p.3)

This array has two dimensions; one for the grammar symbols and one for the parser states. The elements of the array indicate whether the parser is to shift, reduce or accept the input grammar string. There are also invalid actions which could be the result of a non LR(1) grammar.

H. LALR_GOTO (Appendix D p.3)

Same as the GOTOS array except this goto array is for the LALR parsing table whereas the GOTOS array was for the LR(1) parsing table.

I. LALR_ACT (Appendix D p.3)

Same as the ACTION array except this action array is for the LALR parsing table whereas the ACTION array was for the LR(1) parsing table.

Parser Generator Procedure Relationships

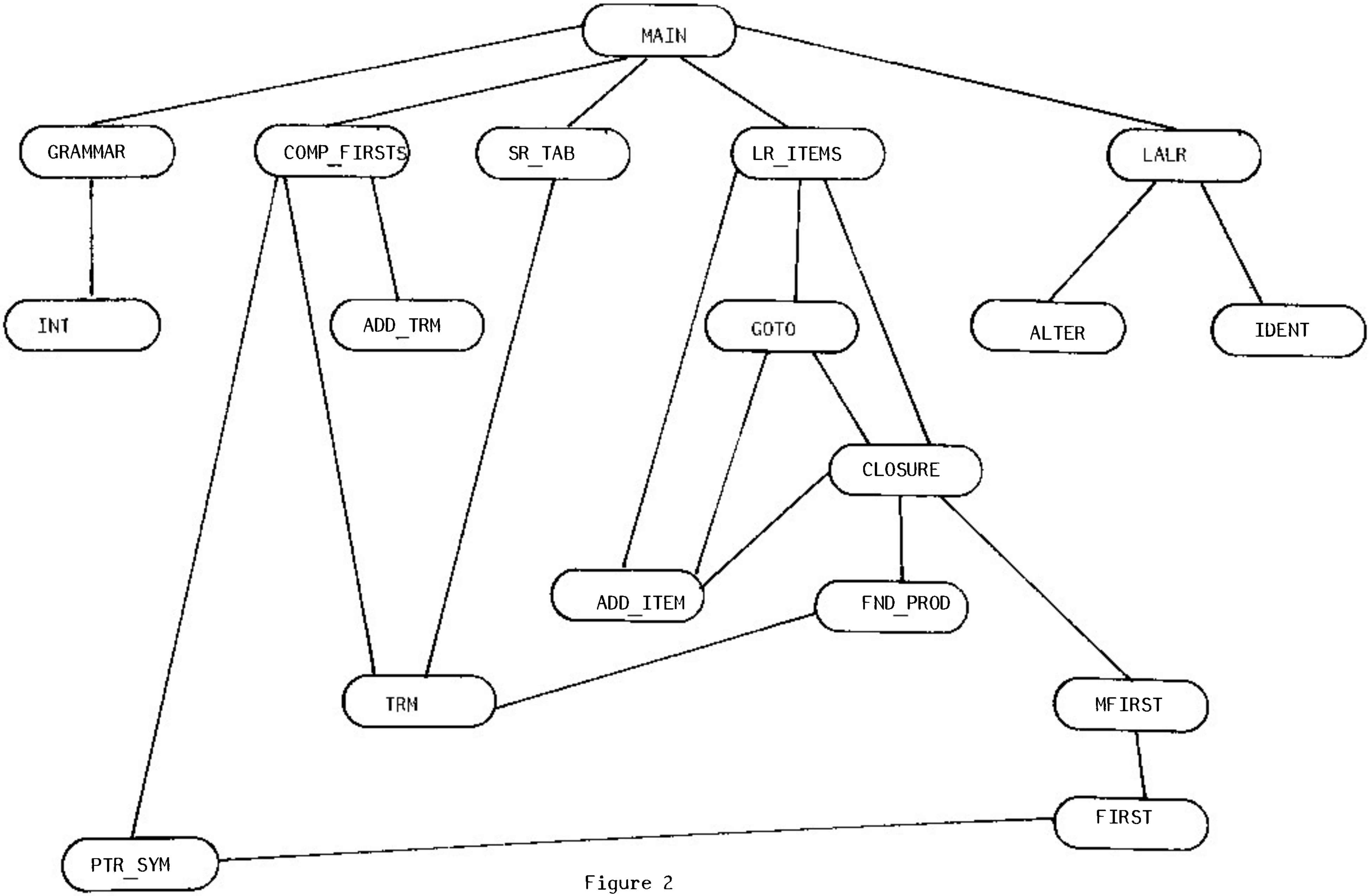

Figure 2

A. Main Program (Appendix D pp 6-9)

The main processing breaks down into the following steps:

1. Call GRAMMAR

   This procedure (see full description below) reads in the grammar definition and sets up the terminals, non-terminals, production and internal grammar symbols.

2. The grammar in its internal format is printed.

3. Call COMP_FIRSTS computes the FIRST function for each grammar symbol and stores it in the array FIRSTS.

4. Call LR_ITEMS generates the LR(1) items for this grammar.

5. Call SR_TAB generates the LR(1) shift-reduce tables which are contained in arrays ACTION and GOTOS.

6. The LR(1) ACTION and GOTOS arrays are printed.

7. Call LALR generates the LALR(1) shift-reduce tables.

8. The LALR(1) shift-reduce tables are printed.

9. Finally the files needed by the parser, namely the symbol table, the production table, the LALR size table and the action and goto tables are written out to disk.

B. LR_ITEMS (Appendix D p.10)

The LR_ITEMS procedure causes the set of LR(1) items to be generated primarily by calling the procedures GOTO and CLOSURE. LR_ITEMS sets up the first item, adds it to the ITEMS array by calling ADD_ITEM and then calls CLOSURE to perform closure on that set of items. The result of this first call to CLOSURE is that state zero items have been created.

The procedure then makes repeated calls to GOTO (for the goto function) until all possible sets of items have been added to the ITEMS array.

Lastly the item table is printed.

C. CLOSURE (Appendix D p.11)

This procedure performs closure on the set of items contained in array ITEMS_SET where N_ITST is the number of items in the set. Each element of the array ITEMS_SET is an integer that references an item in the array ITEMS.

The main do loop executed is performed until all items in the set are exhausted. For each item in the set to be closed procedure FND_PROD and procedure MFIRST are called. FND_PROD returns a set of applicable productions and MFIRST returns a set of terminals that satisfy the lookahead function. New items are then created by taking all possible combinations of the productions and the lookahead symbols. These new items are added to the ITEMS array by procedure ADD_ITEM.

D. FND_PROD (Appendix D p.12)

This procedure is passed an item of the format $[A \rightarrow \alpha . B \beta , a]$ and returns a set of productions in the grammar that are of the form: $B \rightarrow \gamma$

In other words it returns all productions in the grammar whose left part is the first symbol of the unrecognized part of the item passed it.

The production numbers; of the returned productions, are stored in array PROD_# and the number of productions returned is stored in N_PROD.

E. MFIRST (Appendix D p.13)

This procedure is passed an item and it examines the second symbol of the unrecognized portion of the right part of the item. It then returns either:

1. The set of first symbols (by calling procedure FIRST) for that examined symbol, or

2. The lookahead symbol for the input items if the examined symbol is null.

F. FIRST (Appendix D p.13)

This procedure is passed a grammar symbol and it returns the set of first symbols for the input symbol. The set of first symbols for each grammar symbol has already been computed by procedure COMP_FIRSTS early in the program therefore this procedure is rather trivial.

G. PTR_SYM (Appendix D p.14)

This is a utility procedure that returns the index number to the passed grammar symbol. This procedure is used by procedures FIRST and COMP_FIRSTS because they have the grammar symbol and need the index to that symbol for use in array SYMBOLS.

H. ADD_TRM (Appendix D p.14)

This procedure is used by procedure COMP_FIRSTS to add symbols to set of first symbols in array FIRSTS. The input symbol is not added to the object symbols' list if it is already there.

I. TRM (Appendix D p.14)

This is a utility procedure that given a grammar symbol it returns a bit that indicates whether the passed symbol is a terminal ('0') or non-terminal ('1').

J. COMP_FIRSTS (Appendix D p.15)

This procedure computes the set of first symbols for each grammar symbol. The SYMBOLS array is setup such that all non-terminals appear before all terminal symbols. COMP_FIRSTS takes advantage of that structure by starting with the last entry in the SYMBOLS array and working towards the first entry. Each terminal symbol has as its set of first symbols itself. When the scan encounters a non-terminal it looks for a production whose left part is that non-terminal then it gets the first symbol of the right part and obtains the set of first symbols for it to use for the object non-terminal.

By using this procedure to compute the set of first symbols for each grammar symbol at the beginning of the program, a considerable amount of processing is saved by not doing recursive calls later on.

K. ADD_ITEM (Appendix D p.16)

This procedure adds items to the ITEMS array. It is passed the following information:

1. The state that the item is to be added to.

2. The production number that it is based on.

3. A number that represents the number of symbols recognized in the right part of the production.

4. The lookahead symbol.

The procedure scans the ITEMS array to find the last item of appropriate state and chains it to the new item to be added and then it adds the new item.

A count is kept for the number of items in each state in array STATE_ITEMS; this count is updated.

L. GOTO (Appendix D pp. 17-18)

The GOTO procedure computes the goto function for construction of the LR(1) sets of items. It is provided upon entry the state it is to perform the goto function for and the symbol it is to look for. The third value passed is the relative position of the symbol within the symbol table.

The procedure first finds the first item, within the ITEMS array, of the passed state. Secondly, the procedure copies out all items within that state whose next symbol, beyond the recognized part, is the passed symbol. The items that are copied are placed in two places.

1. in the ITEMS array under a new state and

2. in ITEMS_SET so that closure can be done on this new state.

A test is made on variable N_ITST, which is the number of items to close, and if its zero the procedure is exited. If it is not zero then there are items to perform closure on and procedure CLOSURE is called.

Because of the iterative nature of the LR(1) item generation procedure it is possible to generate duplicate GOTO states therefore this procedure then does a check to see if the newly created set of items has indeed already been added to the universal set. Since the number of generated items in each state is saved in array STATE_ITEMS a quick check of that array will tell which of the states have equal number of items as compared to the new goto state. For those states that have an equal number of items as the new state they must be checked for identically equal by doing an item by item comparison.

If the new state is unique, then it is left as created and the element selected by the input state and symbol of the two dimensional GOTOS array is updated with the number of the newly created state. However, if the new state had already existed then that state number is placed in the GOTOS element as referenced by the input state and symbol.

M. SR_TAB (Appendix D p.19)

This procedure creates the canonical LR parsing table for the input grammar. The technique used is outlined in algorithm 6.3 of Aho and Ullman (1977).

Each item is examined in the ITEMS array and depending upon the item it will result in an array element of ACTION to be filled in with a code that signifies Accept, Reduce or Shift. Accept is the case if the item is the completely recognized first production with no more look ahead characters. Reduce is the case if the item is a completely recognized production. Shift is the case for any item that is a partially recognized production whose next symbol is a terminal.

Before the appropriate action is stored in array ACTION the selected element is examined to see if it has already been filled in with a different action. If it has been then we have an action conflict which means that this grammar is not an LR(1) grammar.

N. GRAMMAR (Appendix D pp. 20-21)

The purpose of this procedure is to read in the grammar definition and set up the internal one character grammar symbols.

The rules for the grammar definition are as follows:

1. Each record must contain a valid 4 character transaction code of:

   NONT for Nonterminal
   TERM for Terminal, or
   PROD for Production.

2. The first non-terminal defined is considered to be the start symbol.

3. Terminals and non-terminals are separated by at least one blank and have a maximum length of 30 characters.

4. Only one production allowed per record and the left part is separated from the right part by a colon.

For transaction codes NONT and TERM the records are scanned picking out each grammar symbol and saved in the SYMBOLS array. Array REP_SYM is used to sequentially assign a single character internal grammar symbol to each input terminal and non-terminal.

Each production definition is scanned for its grammar symbols and these are converted into the internal format and the whole production stored into the PROD array.

O. INT (Appendix D p.22)

This is a utility procedure which converts a full grammar symbol to its internal representation. The conversion is done by scanning the SYMBOLS array.

P. ALTER (Appendix D p.22)

This procedure is used in the conversion process from the LR(1) parsing tables to the LALR parsing tables. In the transformation process when two states are deemed equal then this is called to merge the action entries of the two states into one.

The procedure is passed two state numbers and it merges the action entries from the second state into the first only if the action entry of the first was undefined.

Q. LALR (Appendix D pp.23-24)

This procedure does the conversion from the LR(1) parsing tables to the LALR parsing tables by examining the canonical set of LR(1) items. If two sets of items, in the LR(1) items, are equal without respect to their lookahead symbols then they can be combined.

This procedure does that processing in the following steps:

1. Looks at all combinations of the states taken two at a time to determine equality. Equality of any two states is determined by procedure IDENT. If equality is determined then procedure ALTER is called and array TRANSFORM is updated to reflect the combination of the two states.

2. By looking at array TRANSFORM a list of deleted states is created.

3. The end result is to cause a decrease in the number of states and in the process of state deletion there are valid states that are beyond the state limits of the LALR tables. This portion reassigns the valid states to the places previously occupied by the deleted states. This necessitates moving the appropriate vectors of arrays GOTOS and ACTION to the deleted places plus transforming any reference to a deleted state to a valid state.

4. Arrays LALR_GOTO and LALR_ACT are dynamically allocated with the new sizes.

5. The newly allocated LALR action and goto tables entries are filled in from the transformed LR(1) action and goto tables.

R. IDENT (Appendix D p.25)

This procedure is called by procedure LALR to determine if two states are equal so that they can be reduced to one state for the LALR parsing tables. Equality is determined by checking that each item in one state has an equal item in the second state (meaning that they both reference the same production and have the same number of recognized symbols) and vice versa.

The procedure returns a bit value of one if the states are equal and zero if not.

V. Shift-Reduce Parser

The purpose of this program is to take strings from the grammar defined to the parser generator and to parse them. If they do not parse correctly then we can assume that the strings are not from the grammar.

Besides inputting the parsing tables and the production tables, which are necessary to perform the basic function of this program, the symbol table from the parser generator is also input. By doing this it was not necessary to write a lexical analyzer for each grammar tested. All the information necessary for lexical analysis is contained within the symbol table therefore we have in effect a general lexical analysis within this program.

Important Data Structures

SYMBOLS, PROD, GOTOS, and ACTION are the inportant data structures to this program and they are the same ones as described under the parser generator program.

The GOTOS and ACTION arrays can be either LR(1) or LALR. It makes no difference to this program since it is only concerned with using the actions described within each to perform its function.

Program Description

A. Main Procedure (Appendix E pp.4-6)

1. Declarations, initializations, and program housekeeping are done first.

2. After all the files are opened, SYMTAB, PROD and LALRSZ arrays are initialized by reading the appropriate files.

3. The LALRSZ array gives the dimension of the GOTOS and ACTION arrays so that they can be dynamically allocated. Those arrays are initialized by reading file LALRTAB.

4. The first symbol is obtained by calling procedure GETSYM.

5. Based upon the current state on top of the stack and the input symbol one of the following actions, based upon the entry in the ACTION array, takes place:

   a. Invalid action which results in an invalid string message being produced and the string being flushed.

   b. Accept action which means the string was correctly parsed.

   c. Shift which causes the input symbol to be pushed onto the stack plus the appropriate goto state and another symbol obtained.

   d. Reduce which causes reduction via a grammar production. The top 2n objects on the stack are popped off and replaced by the left part of the production and a goto state, where n is the number of symbols in the right part of the production.

Parser Procedure Relationships

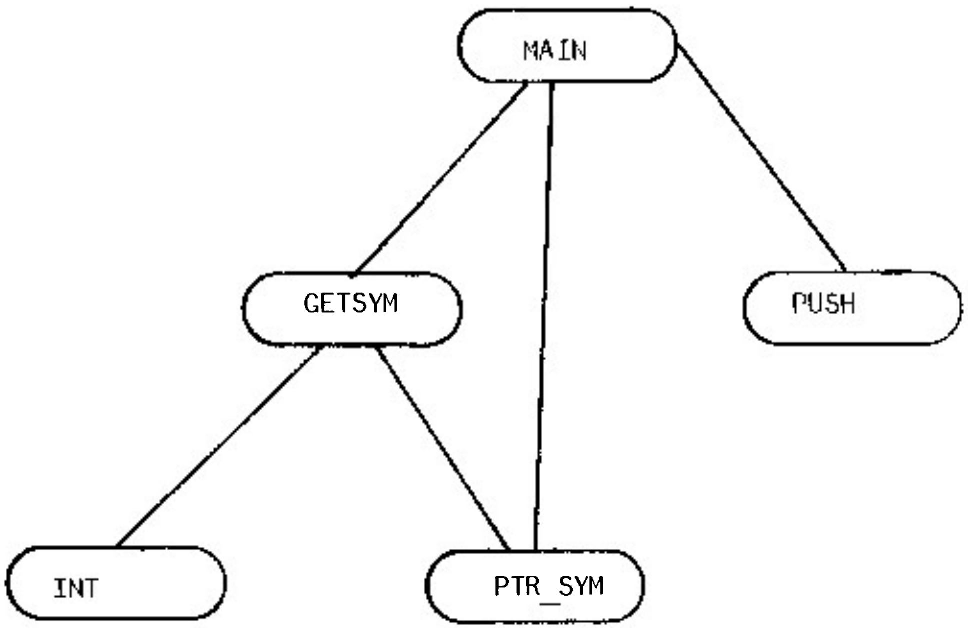

Figure 3

B. PUSH (Appendix E p.7)

This is a utility procedure that takes a value and puts it on the stack.

C. PTR_SYM (Appendix E p.7)

This utility procedure returns an integer index to the passed grammar symbol. The grammar symbol needs to be converted to a relative number so that it can be used to index into GOTOS and ACTION arrays.

D. GETSYM (Appendix E p.8)

This procedure reads in a grammar string, converts the input grammar symbols to their internal representation and returns the next symbol to the calling procedure. When it runs out of symbols in the current string it obtains another record to process.

E. INT (Appendix E p.9)

This is a utility procedure which converts a full grammar symbol to its internal representation. The conversion is done by scanning the SYMBOLS array.

VI. Example Explanation

Three examples of the automatic parser output and symtactical analysis of selected input strings are presented in appendices A, B, and C.

Specifically, Appendix A uses the same grammar example that Aho and Ullman (1977) use in their example 6.10. Appendix B uses the grammar for standard algebraic expressions and Appendix C uses a subset of the EASY language defined by Wetherell (1978) for IF THEN ELSE statements.

Each appendix presents a different LR(1) grammar and is organized as follows:

1. The input grammar definition specifying the grammar nonterminals, terminals and production rules.

2. The internal grammar definition used by the parser generator. The purpose of having an internal grammar is so that grammar symbols can be reduced to single characters which facilitates the processing.

3. The computed first symbols for each grammar symbol.

4. The generated LR(1) items.

5. The LR(1) goto and action tables.

6. The LR(1) to LALR(1) transformation which shows which LR(1) states were deleted and the final state transformation from LR(1) to LALR(1).

7. The LALR(1) goto and action tables.

8. Examples of the parsing action taken on selected strings from the grammar.

In order to analyze input grammar strings the parser uses the method presented by Aho and Ullman (1972) in their algorithm 5.7 which follows:

ALGORITHM 5.7

LR(k) parsing algorithm.

Input. A set T of LR(k) tables for an LR(k) grammar $G=(N,\Sigma,P,S,)$, with $T_0 \in T$ designated as the initial table, and an input string $z \in \Sigma^*$, which is to be parsed.

Output. If $z \in L(G)$, the right parse of G. Otherwise, an error indication.

Method. Perform steps (1) and (2) until acceptance occurs or an error is encountered. If acceptance occurs, the string in the output buffer is the right parse of z.

(1) The lookahead string u, consisting of the next k input symbols, is determined.

(2) The parsing action function f of the table on top of the pushdown list is applied to the lookahead string u.

(a) If f(u)=shift, then the next input symbol, say a, is removed from the input and shifted onto the pushdown list. The goto function g of the table on top of the pushdown list is applied to a to determine the new table to be placed on top of the pushdown list. We then return to step (1). If there is no next input symbol or g(a) is undefined, halt and declare error.

(b) If f(u)=reduce i and production i is $A \rightarrow \alpha$, then $2|\alpha|$ symbols* are removed from the top of the pushdown list, and production number i is placed in the input buffer. A new table T' is then exposed as the top table of the pushdown list, and the goto function of T' is applied to A to determine the next table to be placed on top of the pushdown list. We place A and this new table on top of the pushdown list and return to step (1).

(c) If f(u)=error, we halt parsing (and, in practice, transfer to an error recovery routine).

(d) If f(u)=accept, we halt and declare the string in the output buffer to be the right parse of the original input string.

*If $\alpha=X_m \ldots X_r$, at this point the top of the pushdown list will be of the form $T_0X_1T_1X_2T_2 \ldots X_r T_r$. Removing $2|\alpha|$ symbols removes the handle from the top of the pushdown list along with any intervening LR tables.

To illustrate how this technique works let's consider the example on p. A5 considering the following:

1. Use algorithm 5.7 in Aho and Ullman (1972) where k=1.

2. In the example on p. A5:

   D = E means that D is the external grammar symbol and E is the internal grammar symbol.

3. Use the LALR(1) goto and action tables on p. A4.

4. The initial state is zero.

5. $ represents the end of the input string.

STACK	INPUT	COMMENT
0	EE$	
0E4	E$	Stack input character and state 4
0C2	E$	Reduce using Production number 4
0C2E4	$	Stack input character and state 4
0C2C5	$	Reduce using production number 4
0B1	$	Reduce using production number 2

With state 1 on top of the stack and no more input the LALR(1) Action function says accept.

To obtain the rightmost parse of a particular grammar sentence one just has to apply the productions in reverse order from any example. To illustrate let's apply the production numbers in reverse order for the example on p. B10.

Production		
1	GRAMMAR →	EXPRESSION
3	→	TERM
4	→	TERM * FACTOR
7	→	TERM * A
5	→	FACTOR * A
6	→	(EXPRESSION) * A
2	→	(EXPRESSION + TERM) * A
5	→	(EXPRESSION + FACTOR) * A
7	→	(EXPRESSION + A) * A
3	→	(TERM +A) * A
5	→	(FACTOR + A) * A
7	→	(A + A) * A

Which does indeed yield the expected grammar string.

All the other examples presented are valid input strings for their particular grammars except for the following ones:

The input string shown on p. A6 fails because the grammar defined generates strings in the format of B^mDB^nD where m does not have to equal n and either m or n can equal zero. Therefore, the string BD must not be from this grammar.

The input string shown on p. B9 fails because the operators * and + cannot be next to each other.

The input string shown on p. C6 fails because in the defined grammar the IF THEN ELSE statement must be terminated with FI.

VII. Conclusion

Problems Encountered

During the implementation there were several notable problem areas. The first problem was that the end result (the parser generator) was quite large taking approximately 800 PL/1 statements. The algorithms and techniques presented in pseudo code by Aho and Ullman expanded significantly when actually set down in a real language.

The second problem encountered had to do with the computation of FIRST for each grammar symbol. The first attempt called for the FIRST calculation to be done in the CLOSURE procedure each time CLOSURE was called. This technique proved to be wasteful and overly complicated. Using the recursive technique was academically appealing but unnecessary. A far simpler and direct approach was to examine the grammar productions and calculate FIRST for each grammar symbol prior to generating the items.

The third problem encountered was dealing with duplicate generated states. Due to the iterative nature of generating the LR(1) items it is possible to generate a new state that is identically equal to a previous state. Therefore, when each new state is generated it must be checked against each previously generated state to ensure its uniqueness. Although some steps were taken to optimize this process it could not be eliminated entirely.

The last problem encountered was during array compression after the LALR(1) array dimensions had been established. In order to achieve the true benefit of the LALR(1) parsing tables, the arrays must contain only valid states. Generally there are more LR(1) states then the resultant LALR(1) states and the deleted states are, unfortunately, not the last column vectors in the arrays. Therefore, it becomes necessary to move valid states, that are beyond the LALR(1) state limits, to deleted LR(1) states. This process includes moving the valid states into the deleted states and changing any reference to the moved state to its new state number.

Possible Extensions

There are two extensions that can be made to this implementation. The first one deals with generating the LALR(1) tables directly instead of going through the intermediate process of generating the LR(1) tables. Using the intermediate step of producing the LR(1) tables is instructive on the technique but in reality takes up much more space than is necessary. Generating the LALR(1) tables directly would be much more efficient.

The second extension that could be investigated is in the area of error processing. In the current implementation when an error is detected in the parsing of a sentence from the grammar the sentence is flushed and the next sentence is processed. Instead of flushing the sentence an error processing function could be invoked to attempt to use the partial parse and perhaps "correct" the sentence so that it is syntactically correct.

Concluding Remarks

There are four benefits to the parser generator described in this paper. The first benefit is that the LALR(1) parser generator can accept a broad class of grammars. Secondly it detects syntax errors at the earliest possible time inthe parsing process. Thirdly the LALR(1) parsing tables are space efficient. Lastly, the generated output of the parser generator greatly simplifies the parser. The parser's actions are determined in a straight forward manner based upon the table entries. In a real implementation the parsing tables could be generated ahead of time and the parser could be directed to use the appropriate set of tables. The end result is that one parser could parse sentences from several different grammars.

BIBLIOGRAPHY

Aho (1977), A.V., J. D. Ullman, Principles of Compiler Design, Addison-Wesley Publishing Company, Reading Massachusetts.

Aho (1972), A.V., J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. 1 Parsing, Prentice-Hall, Englewood Cliffs, N.J.

Barrett (1979), W.A., J. D. Couch, Compiler Construction: Theory and Practice, Science Research Associates, Inc.

Chomsky, N. (1965). "Three Models for the Description of Language", IFEE Transactions on Information Theory, 2:113-124.

Cleaveland (1977), J. C., Uzgalis, R. C., Grammars for Programming Languages, Elsevier North-Holland, Inc., New York, New York.

Knuth (1965), D. E., "On the Translation of Languages from Left to Right", Information and Control, 8, 607-639.

Korenjak, A. J. (1969). "A Practical Method for Constructing LR(k) Processors", Communications of the ACM 12:11, 613-623.

Wetherell, C. (1978). "Etudes for Programmers", Prentice-Hall, Inc., Englewood Cliffs, N.J.

```
NONT G S C
TERM B D
PROD G : S
PROD S : C C
PROD C : B C
PROD C : D

INTERNAL GRAMMAR:
   NONTERMINALS: A B C
   TERMINALS: D E
   START SYMBOL: A
   PRODUCTIONS:
        ( 1) A -> B
        ( 2) B -> CC
        ( 3) C -> DC
        ( 4) C -> E

COMPUTED FIRSTS:
 A ED
 B ED
 C ED
 D D
 E E
```

GENERATED ITEMS

NO.	STATE	ITEMS
1	0	A -> .B,$
2	0	B -> .CC,$
3	0	C -> .DC,E
4	0	C -> .DC,D
5	0	C -> .E,E
6	0	C -> .E,D
7	1	A -> B.,$
8	2	B -> C.C,$
9	2	C -> .DC,$
10	2	C -> .E,$
11	3	C -> D.C,E
12	3	C -> D.C,D
13	3	C -> .DC,E
14	3	C -> .E,E
15	3	C -> .DC,D
16	3	C -> .E,D
17	4	C -> E.,E
18	4	C -> E.,D
19	5	B -> CC.,$
20	6	C -> D.C,$
21	6	C -> .DC,$
22	6	C -> .E,$
23	7	C -> E.,$
24	8	C -> DC.,E
25	8	C -> DC.,D
26	9	C -> DC.,$

LR(1) GOTO FUNCTION:

	0	1	2	3	4	5	6	7	8	9
A										
B	1									
C	2		5	8			9			
D	3		6	3			6			
E	4		7	4			7			

LR(1) ACTION FUNCTION:

	0	1	2	3	4	5	6	7	8	9
A										
B										
C										
D	S		S	S	4		S		3	
E	S		S	S	4		S		3	
$		A				2		4		3

LALR TRANSFORMATION

LR(1) STATES TO DELETE:
6 7 9

LR TO LALR STATE TRANSFORM:

LR	0	1	2	3	4	5	6	7	8	9
LALR	0	1	2	3	4	5	3	4	6	6

LALR(1) GOTO FUNCTION:

	0	1	2	3	4	5	6
A							
B	1						
C	2		5	6			
D	3		3	3			
E	4		4	4			

LALR(1) ACTION FUNCTION:

	0	1	2	3	4	5	6
A							
B							
C							
D	S		S	S	4		3
E	S		S	S	4		3
$		A			4	2	3

```
INPUT STRING:
D D

D => E
   STACK INPUT CHARACTER AND STATE    4
D => E
   REDUCE USING PRODUCTION NO.    4
   STACK INPUT CHARACTER AND STATE    4
 => $
   REDUCE USING PRODUCTION NO.    4
   REDUCE USING PRODUCTION NO.    2
   REDUCE USING PRODUCTION NO.    1
   VALID INPUT STRING
```

```
INPUT STRING:
B D

B => D
   STACK INPUT CHARACTER AND STATE   3
D => E
   STACK INPUT CHARACTER AND STATE   4
 => $
   REDUCE USING PRODUCTION NO.   4
   REDUCE USING PRODUCTION NO.   3
   INVALID STRING FOR THIS GRAMMAR
   INPUT STRING FLUSHED
```

```
INPUT STRING:
B B D B D

B => D
   STACK INPUT CHARACTER AND STATE    3
B => D
   STACK INPUT CHARACTER AND STATE    3
D => E
   STACK INPUT CHARACTER AND STATE    4
B => D
   REDUCE USING PRODUCTION NO.   4
   REDUCE USING PRODUCTION NO.   3
   REDUCE USING PRODUCTION NO.   3
   STACK INPUT CHARACTER AND STATE    3
D => E
   STACK INPUT CHARACTER AND STATE    4
 => $
   REDUCE USING PRODUCTION NO.   4
   REDUCE USING PRODUCTION NO.   3
   REDUCE USING PRODUCTION NO.   2
   REDUCE USING PRODUCTION NO.   1
   VALID INPUT STRING
```

```
NONT GRAMMAR EXPRESSION TERM FACTOR
TERM + * ( ) A
PROD GRAMMAR : EXPRESSION
PROD EXPRESSION : EXPRESSION + TERM
PROD EXPRESSION : TERM
PROD TERM : TERM * FACTOR
PROD TERM : FACTOR
PROD FACTOR : ( EXPRESSION )
PROD FACTOR : A

INTERNAL GRAMMAR:
   NONTERMINALS: A B C D
   TERMINALS: E F G H I
   START SYMBOL: A
   PRODUCTIONS:
        ( 1) A -> B
        ( 2) B -> BEC
        ( 3) B -> C
        ( 4) C -> CFD
        ( 5) C -> D
        ( 6) D -> GBH
        ( 7) D -> I

COMPUTED FIRSTS:
 A IG
 B IG
 C IG
 D IG
 E E
 F F
 G G
 H H
 I I
```

GENERATED ITEMS

NO.	STATE	ITEMS
1	0	A -> .B,$
2	0	B -> .BEC,$
3	0	B -> .C,$
4	0	B -> .BEC,E
5	0	B -> .C,E
6	0	C -> .CFD,$
7	0	C -> .D,$
8	0	C -> .CFD,E
9	0	C -> .D,E
10	0	C -> .CFD,F
11	0	C -> .D,F
12	0	D -> .GBH,$
13	0	D -> .I,$
14	0	D -> .GBH,E
15	0	D -> .I,E
16	0	D -> .GBH,F
17	0	D -> .I,F
18	1	A -> B.,$
19	1	B -> B.EC,$
20	1	B -> B.EC,E
21	2	B -> C.,$
22	2	B -> C.,E
23	2	C -> C.FD,$
24	2	C -> C.FD,E
25	2	C -> C.FD,F
26	3	C -> D.,$
27	3	C -> D.,E
28	3	C -> D.,F
29	4	D -> G.BH,$
30	4	D -> G.BH,E
31	4	D -> G.BH,F
32	4	B -> .BEC,H
33	4	B -> .C,H
34	4	B -> .BEC,E
35	4	B -> .C,E
36	4	C -> .CFD,H
37	4	C -> .D,H
38	4	C -> .CFD,E
39	4	C -> .D,E
40	4	C -> .CFD,F
41	4	C -> .D,F
42	4	D -> .GBH,H
43	4	D -> .I,H
44	4	D -> .GBH,E
45	4	D -> .I,E
46	4	D -> .GBH,F
47	4	D -> .I,F
48	5	D -> I.,$
49	5	D -> I.,E
50	5	D -> I.,F
51	6	B -> BE.C,$
52	6	B -> BE.C,E
53	6	C -> .CFD,$
54	6	C -> .D,$
55	6	C -> .CFD,E
56	6	C -> .D,E
57	6	C -> .CFD,F

58	6	C	->	.D,F
59	6	D	->	.GBH,$
60	6	D	->	.I,$
61	6	D	->	.GBH,E
62	6	D	->	.I,E
63	6	D	->	.GBH,F
64	6	D	->	.I,F
65	7	C	->	CF.D,$
66	7	C	->	CF.D,E
67	7	C	->	CF.D,F
68	7	D	->	.GBH,$
69	7	D	->	.I,$
70	7	D	->	.GBH,E
71	7	D	->	.I,E
72	7	D	->	.GBH,F
73	7	D	->	.I,F
74	8	D	->	GB.H,$
75	8	D	->	GB.H,E
76	8	D	->	GB.H,F
77	8	B	->	B.EC,H
78	8	B	->	B.EC,E
79	9	B	->	C.,H
80	9	B	->	C.,E
81	9	C	->	C.FD,H
82	9	C	->	C.FD,E
83	9	C	->	C.FD,F
84	10	C	->	D.,H
85	10	C	->	D.,E
86	10	C	->	D.,F
87	11	D	->	G.BH,H
88	11	D	->	G.BH,E
89	11	D	->	G.BH,F
90	11	B	->	.BEC,H
91	11	B	->	.C,H
92	11	B	->	.BEC,E
93	11	B	->	.C,E
94	11	C	->	.CFD,H
95	11	C	->	.D,H
96	11	C	->	.CFD,E
97	11	C	->	.D,E
98	11	C	->	.CFD,F
99	11	C	->	.D,F
100	11	D	->	.GBH,H
101	11	D	->	.I,H
102	11	D	->	.GBH,E
103	11	D	->	.I,E
104	11	D	->	.GBH,F
105	11	D	->	.I,F
106	12	D	->	I.,H
107	12	D	->	I.,E
108	12	D	->	I.,F
109	13	B	->	BEC.,$
110	13	B	->	BEC.,E
111	13	C	->	C.FD,$
112	13	C	->	C.FD,E
113	13	C	->	C.FD,F
114	14	C	->	CFD.,$
115	14	C	->	CFD.,E
116	14	C	->	CFD.,F
117	15	B	->	BE.C,H

```
118   15        B  ->  BE.C,E
119   15        C  ->  .CFD,H
120   15        C  ->  .D,H
121   15        C  ->  .CFD,E
122   15        C  ->  .D,E
123   15        C  ->  .CFD,F
124   15        C  ->  .D,F
125   15        D  ->  .GBH,H
126   15        D  ->  .I,H
127   15        D  ->  .GBH,E
128   15        D  ->  .I,E
129   15        D  ->  .GBH,F
130   15        D  ->  .I,F
131   16        D  ->  GBH.,$
132   16        D  ->  GBH.,E
133   16        D  ->  GBH.,F
134   17        C  ->  CF.D,H
135   17        C  ->  CF.D,E
136   17        C  ->  CF.D,F
137   17        D  ->  .GBH,H
138   17        D  ->  .I,H
139   17        D  ->  .GBH,E
140   17        D  ->  .I,E
141   17        D  ->  .GBH,F
142   17        D  ->  .I,F
143   18        D  ->  GB.H,H
144   18        D  ->  GB.H,E
145   18        D  ->  GB.H,F
146   18        B  ->  B.EC,H
147   18        B  ->  B.EC,E
148   19        B  ->  BEC.,H
149   19        B  ->  BEC.,E
150   19        C  ->  C.FD,H
151   19        C  ->  C.FD,E
152   19        C  ->  C.FD,F
153   20        C  ->  CFD.,H
154   20        C  ->  CFD.,E
155   20        C  ->  CFD.,F
156   21        D  ->  GBH.,H
157   21        D  ->  GBH.,E
158   21        D  ->  GBH.,F
```

LR(1) GOTO FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A																						
B	1				8																	
C	2				9		13															
D	3				10		3	14														
E		6							15													
F			7							17												
G	4				11		4	4														
H									16													
I	5				12		5	5														

LR(1) ACTION FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A																						
B																						
C																						
D																						
E		S	3	5		7			S	3	5		7	2	4		6		S	2	4	6
F			S	5		7				S	5		7	S	4		6			S	4	6
G	S				S		S	S				S				S		S				
H									S	3	5		7						S	2	4	6
I	S				S		S	S				S				S		S				
$		A	3	5		7								2	4		6					

LALR TRANSFORMATION

LR(1) STATES TO DELETE:
9 10 11 12 15 17 18 19 20 21

LR TO LALR STATE TRANSFORM:

LR	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
LALR	0	1	2	3	4	5	6	7	8	2	3	4	5	9	10	6	11	7	8	9	10	11

LALR(1) GOTO FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11
A												
B	1				8							
C	2				2		9					
D	3				3		3	10				
E		6							6			
F			7							7		
G	4				4		4	4				
H									11			
I	5				5		5	5				

LALR(1) ACTION FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11
A												
B												
C												
D												
E		S	3	5		7			S	2	4	6
F			S	5		7				S	4	6
G	S				S		S	S				
H			3	5		7			S	2	4	6
I	S				S		S	S				
$		A	3	5		7				2	4	6

```
INPUT STRING:
A + A + A * A

A => I
   STACK INPUT CHARACTER AND STATE    5
+ => E
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   REDUCE USING PRODUCTION NO.    3
   STACK INPUT CHARACTER AND STATE    6
A => I
   STACK INPUT CHARACTER AND STATE    5
+ => E
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   REDUCE USING PRODUCTION NO.    2
   STACK INPUT CHARACTER AND STATE    6
A => I
   STACK INPUT CHARACTER AND STATE    5
* => F
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   STACK INPUT CHARACTER AND STATE    7
A => I
   STACK INPUT CHARACTER AND STATE    5
  => $
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    4
   REDUCE USING PRODUCTION NO.    2
   REDUCE USING PRODUCTION NO.    1
   VALID INPUT STRING
```

```
INPUT STRING:
A * A + A + A

A => I
   STACK INPUT CHARACTER AND STATE    5
* => F
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   STACK INPUT CHARACTER AND STATE    7
A => I
   STACK INPUT CHARACTER AND STATE    5
+ => E
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    4
   REDUCE USING PRODUCTION NO.    3
   STACK INPUT CHARACTER AND STATE    6
A => I
   STACK INPUT CHARACTER AND STATE    5
+ => E
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   REDUCE USING PRODUCTION NO.    2
   STACK INPUT CHARACTER AND STATE    6
A => I
   STACK INPUT CHARACTER AND STATE    5
 => $
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   REDUCE USING PRODUCTION NO.    2
   REDUCE USING PRODUCTION NO.    1
   VALID INPUT STRING
```

```
INPUT STRING:
A * A + A * + A

A => I
   STACK INPUT CHARACTER AND STATE    5
* => F
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   STACK INPUT CHARACTER AND STATE    7
A => I
   STACK INPUT CHARACTER AND STATE    5
+ => E
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    4
   REDUCE USING PRODUCTION NO.    3
   STACK INPUT CHARACTER AND STATE    6
A => I
   STACK INPUT CHARACTER AND STATE    5
* => F
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   STACK INPUT CHARACTER AND STATE    7
+ => E
   INVALID STRING FOR THIS GRAMMAR
   INPUT STRING FLUSHED
A => I
  => $
```

```
INPUT STRING:
( A + A ) * A

( => G
   STACK INPUT CHARACTER AND STATE    4
A => I
   STACK INPUT CHARACTER AND STATE    5
+ => E
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   REDUCE USING PRODUCTION NO.    3
   STACK INPUT CHARACTER AND STATE    6
A => I
   STACK INPUT CHARACTER AND STATE    5
) => H
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   REDUCE USING PRODUCTION NO.    2
   STACK INPUT CHARACTER AND STATE   11
* => F
   REDUCE USING PRODUCTION NO.    6
   REDUCE USING PRODUCTION NO.    5
   STACK INPUT CHARACTER AND STATE    7
A => I
   STACK INPUT CHARACTER AND STATE    5
 => $
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    4
   REDUCE USING PRODUCTION NO.    3
   REDUCE USING PRODUCTION NO.    1
   VALID INPUT STRING
```

```
INPUT STRING:
A + A * A

A => I
   STACK INPUT CHARACTER AND STATE    5
+ => E
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   REDUCE USING PRODUCTION NO.    3
   STACK INPUT CHARACTER AND STATE    6
A => I
   STACK INPUT CHARACTER AND STATE    5
* => F
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    5
   STACK INPUT CHARACTER AND STATE    7
A => I
   STACK INPUT CHARACTER AND STATE    5
 => $
   REDUCE USING PRODUCTION NO.    7
   REDUCE USING PRODUCTION NO.    4
   REDUCE USING PRODUCTION NO.    2
   REDUCE USING PRODUCTION NO.    1
   VALID INPUT STRING
```

```
NONT    <COND_STMT>  <SCS> <COND_CL>
NONT <TRUE> <FALSE> <COND_BODY>  <ELSE>
TERM    SEG_BODY FI IF EXPR THEN ELSE
PROD <COND_STMT> : <SCS>
PROD <SCS> : <COND_CL> <TRUE> FI
PROD <SCS> : <COND_CL> <TRUE> <FALSE> FI
PROD <COND_CL> : IF EXPR
PROD <TRUE> : THEN <COND_BODY>
PROD <FALSE> : <ELSE> <COND_BODY>
PROD <ELSE> : ELSE
PROD <COND_BODY> : SEG_BODY

INTERNAL GRAMMAR:
    NONTERMINALS: A B C D E F G
    TERMINALS: H I J K L M
    START SYMBOL: A
    PRODUCTIONS:
          ( 1) A -> B
          ( 2) B -> CDI
          ( 3) B -> CDEI
          ( 4) C -> JK
          ( 5) D -> LF
          ( 6) E -> GF
          ( 7) G -> M
          ( 8) F -> H

COMPUTED FIRSTS:
 A J
 B J
 C J
 D L
 E M
 F H
 G M
 H H
 I I
 J J
 K K
 L L
 M M
```

GENERATED ITEMS

NO.	STATE	ITEMS
1	0	A -> .B,$
2	0	B -> .CDI,$
3	0	B -> .CDEI,$
4	0	C -> .JK,L
5	1	A -> B.,$
6	2	B -> C.DI,$
7	2	B -> C.DEI,$
8	2	D -> .LF,I
9	2	D -> .LF,M
10	3	C -> J.K,L
11	4	B -> CD.I,$
12	4	B -> CD.EI,$
13	4	E -> .GF,I
14	4	G -> .M,H
15	5	D -> L.F,I
16	5	D -> L.F,M
17	5	F -> .H,I
18	5	F -> .H,M
19	6	C -> JK.,L
20	7	B -> CDE.I,$
21	8	E -> G.F,I
22	8	F -> .H,I
23	9	B -> CDI.,$
24	10	G -> M.,H
25	11	D -> LF.,I
26	11	D -> LF.,M
27	12	F -> H.,I
28	12	F -> H.,M
29	13	B -> CDEI.,$
30	14	E -> GF.,I
31	15	F -> H.,I

LR(1) GOTO FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A																
B	1															
C	2															
D			4													
E					7											
F						11			14							
G					8											
H						12			15							
I					9			13								
J	3															
K				6												
L			5													
M					10											

LR(1) ACTION FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A																
B																
C																
D																
E																
F																
G																
H						S			S		7					
I					S			S				5	8		6	8
J	S															
K				S												
L			S				4									
M					S							5	8			
$		A								2				3		

LALR TRANSFORMATION

LR(1) STATES TO DELETE:
15

LR TO LALR STATE TRANSFORM:

LR	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LALR	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	12

LALR(1) GOTO FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A															
B	1														
C	2														
D			4												
E					7										
F						11			14						
G					8										
H						12			12						
I					9			13							
J	3														
K				6											
L			5												
M					10										

LALR(1) ACTION FUNCTION:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A															
B															
C															
D															
E															
F															
G															
H						S			S		7				
I					S			S				5	8		6
J	S														
K				S											
L			S				4								
M					S							5	8		
$		A								2				3	

```
INPUT STRING:
IF EXPR THEN SEG_BODY FI

IF => J
   STACK INPUT CHARACTER AND STATE    3
EXPR => K
   STACK INPUT CHARACTER AND STATE    6
THEN => L
   REDUCE USING PRODUCTION NO.   4
   STACK INPUT CHARACTER AND STATE    5
SEG_BODY => H
   STACK INPUT CHARACTER AND STATE   12
FI => I
   REDUCE USING PRODUCTION NO.   6
   REDUCE USING PRODUCTION NO.   5
   STACK INPUT CHARACTER AND STATE    9
 => $
   REDUCE USING PRODUCTION NO.   2
   REDUCE USING PRODUCTION NO.   1
   VALID INPUT STRING
```

```
INPUT STRING:
IF EXPR THEN SEG_BODY

IF => J
   STACK INPUT CHARACTER AND STATE   3
EXPR => K
   STACK INPUT CHARACTER AND STATE   6
THEN => L
   REDUCE USING PRODUCTION NO.    4
   STACK INPUT CHARACTER AND STATE   5
SEG_BODY => H
   STACK INPUT CHARACTER AND STATE  12
 => $
   INVALID STRING FOR THIS GRAMMAR
   INPUT STRING FLUSHED
```

```
INPUT STRING:
IF EXPR THEN SEG_BODY ELSE SEG_BODY FI

IF => J
   STACK INPUT CHARACTER AND STATE   3
EXPR => K
   STACK INPUT CHARACTER AND STATE   6
THEN => L
   REDUCE USING PRODUCTION NO.   4
   STACK INPUT CHARACTER AND STATE   5
SEG_BODY => H
   STACK INPUT CHARACTER AND STATE  12
ELSE => M
   REDUCE USING PRODUCTION NO.   8
   REDUCE USING PRODUCTION NO.   5
   STACK INPUT CHARACTER AND STATE  10
SEG_BODY => H
   REDUCE USING PRODUCTION NO.   7
   STACK INPUT CHARACTER AND STATE  12
FI => I
   REDUCE USING PRODUCTION NO.   8
   REDUCE USING PRODUCTION NO.   6
   STACK INPUT CHARACTER AND STATE  13
 => $
   REDUCE USING PRODUCTION NO.   3
   REDUCE USING PRODUCTION NO.   1
   VALID INPUT STRING
```

```
OPTIONS SPECIFIED

MAR(2 72 1) NUM GN TERM;

OPTIONS USED

AGGREGATE       NOCOUNT          ATTRIBUTES(SHORT)
COMPILE         NODECK           CHARSET(60,EBCDIC)
GONUMBER        NOESD            FLAG(I)
INSOURCE        NOFLOW           LINECOUNT(60)
LMESSAGE        NOGOSTMT         MARGINS(2,72,1)
MAP             NOIMPRECISE      OPTIMIZE(TIME)
NEST            NOINCLUDE        SEQUENCE(73,80)
NUMBER          NOINTERRUPT      SIZE(327440)
OBJECT          NOLIST           NOSYNTAX(S)
OFFSET          NOMACRO          XREF(SHORT)
OPTIONS         NOMARGINI        TERMINAL(NOAGGREGATE,
SOURCE          NOMDECK                   NOATTRIBUTES,
STORAGE         NOSTMT                    NOESD,
                                          NOINSOURCE,
                                          NOLIST,
                                          NOMAP,
                                          NOOFFSET,
                                          NOOPTIONS,
                                          NOSOURCE,
                                          NOSTORAGE,
                                          NOXREF)
```

SOURCE LISTING

```
NUMBER LEV NT

                     /* PARSER GENERATOR     */                                             ZEF00010
  20       0  ZEFSOR: PROC OPTIONS(MAIN) REORDER;                                           ZEF00020

  30    1  0  DCL SYSPRINT STREAM OUTPUT PRINT FILE                                         ZEF00030
                         ENV ( VB RECSIZE(121) BLKSIZE(125));                               ZEF00040
  50    1  0  DCL IN FILE RECORD INPUT;                                                     ZEF00050
  60    1  0  DCL SYMTAB FILE RECORD OUTPUT;     /* SYMBOL TABLE */                         ZEF00060
  70    1  0  DCL PRODTAB FILE RECORD OUTPUT;    /* PRODUCTIONS TABLE */                    ZEF00070
  80    1  0  DCL LALRTAB FILE RECORD OUTPUT;     /* LALR TABLES */                         ZEF00080
  90    1  0  DCL LALRSZ FILE RECORD OUTPUT;     /* LALR TABLE SIZES */                     ZEF00090

               /*--------------------------------------------------------------           ZEF00100
                  WHEN CHANGING NO. OF SYMBOLS SUPPORTED, CHANGE THE FOLLOWING              ZEF00110
                  STRUCTURES:                                                               ZEF00120
                       ACTION,GOTOS,SYMBOLS,NO_SYM,REP_SYM,FIRSTS, AND REPSYM.              ZEF00130
               --------------------------------------------------------------*/             ZEF00140

 150    1  0  DCL NO_SYM FIXED BIN(15) STATIC INIT(40);                                     ZEF00150
 160    1  0  DCL 1 SYMBOLS (40) STATIC,                                                    ZEF00160
                  2 SYM CHAR(1) INIT('R','S','C','B','D',' '),                              ZEF00170
                  2 TYPE BIT(1) INIT('1','1','1','0','0'),                                  ZEF00180
                  2 L_FULL FIXED BIN(15),  /* LENGTH OF COMPLETE ID  */                     ZEF00190
                  2 FULL CHAR(30);         /* COMPLETE IDENTIFIER */                        ZEF00200
 210    1  0  DCL NO_PROD FIXED BIN(15) STATIC INIT(20);                                    ZEF00210
 220    1  0  DCL 1 PROD (20) STATIC,                                                       ZEF00220
                  2 NO FIXED BIN(15) INIT(1,2,3,4,0),                                       ZEF00230
                  2 LP CHAR(1) INIT('R','S','C','C'),                                       ZEF00240
                  2 NO_RP FIXED BIN(15) INIT(1,2,2,1),                                      ZEF00250
                  2 RP CHAR(6) INIT('S','CC','BC','D');                                     ZEF00260

 270    1  0  DCL NEW_RP CHAR(10);                                                          ZEF00270

 280    1  0  DCL STATE FIXED BIN(15);   /* NO. OF LR(1) STATES */                          ZEF00280
 290    1  0  DCL 1 ITEMS (300) STATIC,    /* SET C ON P. 216  */                           ZEF00290
                  2 STATE FIXED BIN(15) INIT((300)(-1)),                                    ZEF00300
                      /* ITEM STATE */                                                      ZEF00310
                  2 NEXT FIXED BIN(15) INIT((300)(-1)),                                     ZEF00320
                      /* POINTS TO NEXT ITEM IN STATE OR -1 FOR LAST */                     ZEF00330
                  2 PROD FIXED BIN(15), /* POINTS TO PRODUCTION */                          ZEF00340
                  2 POSIT FIXED BIN(15), /* RECOGNIZED PART */                              ZEF00350
                  2 LA CHAR(1);  /* LOOK AHEAD SYMBOL */                                    ZEF00360
 370    1  0  DCL NEXT_I FIXED BIN(15);  /* POINTS TO NEXT AVAILABLE ITEM */                ZEF00370

 380    1  0  DCL FRST(20) CHAR(1);                                                         ZEF00380
 390    1  0  DCL N_FRST FIXED BIN(15);                                                     ZEF00390

 400    1  0  DCL 1 FIRSTS(40) STATIC,                                                      ZEF00400
                  2 NO_FIRSTS FIXED BIN(15) INIT((40)0),    /* NO. OF INPUTS */             ZEF00410
                  2 FIRSTS(6) CHAR(1);     /* FIRST SYMBOLS */                              ZEF00420
 430    1  0  DCL TOT_FIRSTS FIXED BIN(15) STATIC INIT(0);                                  ZEF00430
 440    1  0  DCL SYMBOL CHAR(1);                                                           ZEF00440
```

```
NUMBER LEV NT

  450    1  0  DCL ITEMS_SET(100) FIXED BIN(15); /* SET OF ITEMS USED BY CLOSURE */    ZEF00450
  460    1  0  DCL N_ITST FIXED BIN(15);  /* NO. OF ITEMS IN ITEMS_SET  */               ZEF00460

  470    1  0  DCL PROD_# (10) FIXED BIN(15); /* USED BY FND_PROD  */                    ZEF00470
  480    1  0  DCL N_PROD FIXED BIN(15);  /* NO. OF PRODUCTIONS IN PROD_#  */            ZEF00480

  490    1  0  DCL STATE_ITEMS(0:29) FIXED BIN(15) STATIC INIT((30)0);                   ZEF00490
                    /* NO. OF ITEMS IN EACH STATE */                                     ZEF00500

  510    1  0  DCL GOTOS(15,0:29) FIXED BIN(15) STATIC INIT((450)-1);                    ZEF00510
                  /* (M,N) WHERE M = # GRAMMAR SYMBOLS AND                               ZEF00520
                                 N = # STATES.              */                           ZEF00530

  540    1  0  DCL ACTION(15,0:29) FIXED BIN(15) STATIC INIT((450)-1);                   ZEF00540
                  /* (M,N) WHERE M = # GRAMMAR SYMBOLS AND                               ZEF00550
                                 N = # STATES.              */                           ZEF00560

  570    1  0  DCL 1 LALR_SZ,        /* LALR GOTO & ACTION ARRAYS */                     ZEF00570
                    2 M FIXED BIN(15),    /* NO. OF GRAMMAR SYMBOLS  */                  ZEF00580
                    2 N FIXED BIN(15);    /* NO. OF STATES */                            ZEF00590

  600    1  0  DCL LALR_GOTO(LALR_SZ.M,0:LALR_SZ.N) FIXED BIN(15) CTL;                   ZEF00600
  610    1  0  DCL LALR_ACT(LALR_SZ.M,0:LALR_SZ.N) FIXED BIN(15) CTL;                    ZEF00610
```

```
NUMBER LEV NT

                    /* MISC DECLARATIONS */
 630    1  0  DCL (I,J,K,L) FIXED BIN(15);
 640    1  0  DCL RSND FIXED BIN(15) STATIC INIT(1);
 650    1  0  DCL (M,N,P) FIXED DEC(3,0);
 660    1  0  DCL (NONTERM,TERM) CHAR(100) VAR;
 670    1  0  DCL (ADDR,SUBSTR) BUILTIN;
 680    1  0  DCL HIGH_STATE FIXED BIN(15) STATIC INIT(0);
 690    1  0  DCL NEWSTATE FIXED BIN(15);
 700    1  0  DCL CHAR CHAR(1);
 710    1  0  DCL CHAR2 CHAR(2);
 720    1  0  DCL D_CHAR2 FIXED DEC(3,0) BASED(P_CHAR2);
 730    1  0  DCL P_CHAR2 PTR;
 740    1  0        P_CHAR2 = ADDR(CHAR2);
 750    1  0  DCL MORE BIT(1) STATIC INIT('1'B);
```

```
ZEF00620
ZEF00630
ZEF00640
ZEF00650
ZEF00660
ZEF00670
ZEF00680
ZEF00690
ZEF00700
ZEF00710
ZEF00720
ZEF00730
ZEF00740
ZEF00750
```

```
NUMBER LEV NT

                   /* ON UNITS AND INITIALIZATION */                        ZEF00760
  770   1  0    ON SUBRG SIGNAL ERROR;                                       ZEF00770
  780   1  0    ON STRG SIGNAL ERROR;                                        ZEF00780

  790   1  0        ON ERROR SNAP                                            ZEF00790
                             BEGIN;                                          ZEF00800
  810   2  0                   CLOSE FILE (SYSPRINT),                        ZEF00810
                                     FILE (IN);                              ZEF00820
  830   2  0                 END;                                            ZEF00830

  840   1  0        ON ENDFILE(IN) MORE = '0'B;                              ZEF00840

  850   1  0        OPEN FILE (SYSPRINT),                                    ZEF00850
                         FILE (IN);                                          ZEF00860
```

```
NUMBER LEV NT

                      /* MAIN PROCESSING  */                                        ZEF00870
  880    1  0      CALL GRAMMAR;  /* SET UP SYMBOL & PRODUCTION TABLES */           ZEF00880
                /* PRINT GRAMMAR (START SYMBOL,NONTERMINALS,TERMINALS,              ZEF00890
                     AND PRODUCTIONS    */                                          ZEF00900
  910    1  0      DO I = 1 TO NO_SYM WHILE (SYMBOLS(I).SYM ¬= ' ');                ZEF00910
  920    1  1         IF SYMBOLS(I).TYPE                                            ZEF00920
                        THEN NONTERM = NONTERM || SYMBOLS(I).SYM || ' ';            ZEF00930
  940    1  1           ELSE TERM = TERM || SYMBOLS(I).SYM || ' ';                  ZEF00940
  950    1  1      END;                                                             ZEF00950
  960    1  0      PUT EDIT ('INTERNAL GRAMMAR:','NONTERMINALS: ',NONTERM,          ZEF00960
                            'TERMINALS: ',TERM,'START SYMBOL: ',                    ZEF00970
                            SYMBOLS(1).SYM,'PRODUCTIONS: ')                         ZEF00980
                           ( SKIP(2),COL(1),A,                                      ZEF00990
                             SKIP,COL(4),A,SKIP(0),COL(18),A,                       ZEF01000
                             SKIP,COL(4),A,SKIP(0),COL(15),A,                       ZEF01010
                             SKIP,COL(4),A,SKIP(0),COL(18),A,                       ZEF01020
                             SKIP,COL(4),A);                                        ZEF01030
 1040    1  0      DO I = 1 TO NO_PROD WHILE (PROD(I).NO ¬= 0);                     ZEF01040
 1050    1  1         N = PROD(I).NO;                                               ZEF01050
 1060    1  1         PUT EDIT ( '(',N,') ',PROD(I).LP,' -> ',PROD(I).RP )          ZEF01060
                               ( COL(8),A,P'Z9',4 A);                               ZEF01070
 1080    1  1      END;                                                             ZEF01080
 1090    1  0      CALL COMP_FIRSTS; /* COMPUTE FIRSTS FOR EACH NONTERMINAL */      ZEF01090
 1100    1  0      PUT EDIT ('GENERATED ITEMS') (A) PAGE;                           ZEF01100
 1110    1  0      PUT EDIT ('NO.','STATE','ITEMS')                                 ZEF01110
                            (SKIP(2),COL(1),A,COL(5),A,COL(15),A);                  ZEF01120
 1130    1  0      CALL LR_ITEMS;    /* GENERATE LR(1) ITEMS */                     ZEF01130
 1140    1  0      CALL SR_TAB;  /*  GENERATE SHIFT-REDUCE TABLE */                 ZEF01140
```

```
NUMBER LEV NT

                  /* OUTPUT GOTOS ARRAY */                                        ZEF01150

 1160   1  0         PUT EDIT ('LR(1) GOTO FUNCTION:') (A) PAGE;                  ZEF01160
 1170   1  0         PUT EDIT (' ') (SKIP,A);                                     ZEF01170
 1180   1  0         DO J = 0 TO NEWSTATE;                                        ZEF01180
 1190   1  1            M = J;                                                    ZEF01190
 1200   1  1            PUT EDIT (' ',M) (A,P'Z9');                               ZEF01200
 1210   1  1         END;                                                         ZEF01210
 1220   1  0         DO I = 1 BY 1 UNTIL(SYMBOLS(I).SYM = ' ');                   ZEF01220
 1230   1  1            PUT EDIT (SYMBOLS(I).SYM) (SKIP,A);                       ZEF01230
 1240   1  1            DO J = 0 TO 9;                                            ZEF01240
 1250   1  2               IF GOTOS(I,J) = -1                                     ZEF01250
                              THEN M = 0;                                         ZEF01260
 1270   1  2                  ELSE M = GOTOS(I,J);                                ZEF01270
 1280   1  2               PUT EDIT (' ',M) (A,P'ZZ');                            ZEF01280
 1290   1  2            END;                                                      ZEF01290
 1300   1  1         END;                                                         ZEF01300

                  /* OUTPUT ACTION FUNCTION */                                    ZEF01310

 1320   1  0         PUT EDIT ('LR(1) ACTION FUNCTION:') (SKIP(2),A);             ZEF01320
 1330   1  0         PUT EDIT (' ') (SKIP,A);                                     ZEF01330
 1340   1  0         DO J = 0 TO NEWSTATE;                                        ZEF01340
 1350   1  1            M = J;                                                    ZEF01350
 1360   1  1            PUT EDIT (' ',M) (A,P'Z9');                               ZEF01360
 1370   1  1         END;                                                         ZEF01370
 1380   1  0         DO I = 1 BY 1 UNTIL(SYMBOLS(I).SYM = ' ');                   ZEF01380
 1390   1  1            IF SYMBOLS(I).SYM = ' '                                   ZEF01390
                           THEN CHAR = '$';                                       ZEF01400
 1410   1  1               ELSE CHAR = SYMBOLS(I).SYM;                            ZEF01410
 1420   1  1             PUT EDIT (CHAR) (COL(1),A);                              ZEF01420
 1430   1  1             DO J = 0 TO NEWSTATE;                                    ZEF01430
 1440   1  2                O_CHAR2 = ACTION(I,J);   /* ASSUME REDUCE */          ZEF01440
 1450   1  2                IF ACTION(I,J) = -1                                   ZEF01450
                               THEN CHAR2 = ' ';     /* ERROR CONDITION */        ZEF01460
 1470   1  2                IF ACTION(I,J) = 0                                    ZEF01470
                               THEN CHAR2 = ' A';     /* ACCEPT */                ZEF01480
 1490   1  2                IF ACTION(I,J) = -2                                   ZEF01490
                               THEN CHAR2 = ' S';     /* SHIFT */                 ZEF01500
 1510   1  2                IF ACTION(I,J) > 0                                    ZEF01510
                               THEN PUT EDIT (' ',O_CHAR2) (A,P'Z9');             ZEF01520
 1530   1  2                   ELSE PUT EDIT (' ',CHAR2) (2 A);                   ZEF01530
 1540   1  2             END;                                                     ZEF01540
 1550   1  1         END;                                                         ZEF01550

 1560   1  0         CALL LALR;  /* CREATE LALR TABLES */                         ZEF01560
```

```
 NUMBER LEV NT

                         /* OUTPUT LALR GOTOS ARRAY */                                        ZEF01570

   1580    1  0          PUT EDIT ('LALR(1) GOTO FUNCTION:') (SKIP(2),A);                     ZEF01580
   1590    1  0          PUT EDIT (' ') (SKIP,A);                                             ZEF01590
   1600    1  0          DO J = 0 TO LALR_SZ.N;                                               ZEF01600
   1610    1  1             M = J;                                                            ZEF01610
   1620    1  1             PUT EDIT (' ',M) (A,P'Z9');                                       ZEF01620
   1630    1  1          END;                                                                 ZEF01630
   1640    1  0          DO I = 1 BY 1 WHILE(SYMBOLS(I).SYM ¬= ' ');                          ZEF01640
   1650    1  1             PUT EDIT (SYMBOLS(I).SYM) (SKIP,A);                               ZEF01650
   1660    1  1             DO J = 0 TO LALR_SZ.N;                                            ZEF01660
   1670    1  2                IF LALR_GOTO(I,J) = -1                                         ZEF01670
                                  THEN M = 0;                                                 ZEF01680
   1690    1  2                   ELSE M = LALR_GOTO(I,J);                                    ZEF01690
   1700    1  2                PUT EDIT (' ',M) (A,P'ZZ');                                    ZEF01700
   1710    1  2             END;                                                              ZEF01710
   1720    1  1          END;                                                                 ZEF01720

                         /* OUTPUT LALR ACTION FUNCTION */                                    ZEF01730

   1740    1  0          PUT EDIT ('LALR(1) ACTION FUNCTION:') (SKIP(2),A);                   ZEF01740
   1750    1  0          PUT EDIT (' ') (SKIP,A);                                             ZEF01750
   1760    1  0          DO J = 0 TO LALR_SZ.N;                                               ZEF01760
   1770    1  1             M = J;                                                            ZEF01770
   1780    1  1             PUT EDIT (' ',M) (A,P'Z9');                                       ZEF01780
   1790    1  1          END;                                                                 ZEF01790
   1800    1  0          DO I = 1 BY 1 UNTIL(SYMBOLS(I).SYM = ' ');                           ZEF01800
   1810    1  1             IF SYMBOLS(I).SYM = ' '                                           ZEF01810
                               THEN CHAR = '$';                                               ZEF01820
   1830    1  1                ELSE CHAR = SYMBOLS(I).SYM;                                    ZEF01830
   1840    1  1             PUT EDIT (CHAR) (COL(1),A);                                       ZEF01840
   1850    1  1             DO J = 0 TO LALR_SZ.N;                                            ZEF01850
   1860    1  2                D_CHAR2 = LALR_ACT(I,J);  /* ASSUME REDUCE */                  ZEF01860
   1870    1  2                IF LALR_ACT(I,J) = -1                                          ZEF01870
                                  THEN CHAR2 = ' ';    /* ERROR CONDITION */                  ZEF01880
   1890    1  2                IF LALR_ACT(I,J) = 0                                           ZEF01890
                                  THEN CHAR2 = ' A';    /* ACCEPT */                          ZEF01900
   1910    1  2                IF LALR_ACT(I,J) = -2                                          ZEF01910
                                  THEN CHAR2 = ' S';    /* SHIFT */                           ZEF01920
   1930    1  2                IF LALR_ACT(I,J) > 0                                           ZEF01930
                                  THEN PUT EDIT (' ',D_CHAR2) (A,P'Z9');                      ZEF01940
   1950    1  2                   ELSE PUT EDIT (' ',CHAR2) (2 A);                            ZEF01950
   1960    1  2             END;                                                              ZEF01960
   1970    1  1          END;                                                                 ZEF01970
```

```
NUMBER LEV NT

                   /*--------------------                                        ZEF01960
                     OUTPUT KEY TABLES                                           ZEF01990
                   --------------------*/                                        ZEF02000
  2010   1  0      OPEN FILE(SYMTAB),                                            ZEF02010
                        FILE(PRODTAB),                                           ZEF02020
                        FILE(LALRTAB),                                           ZEF02030
                        FILE(LALRSZ);                                            ZEF02040

  2050   1  0      WRITE FILE(SYMTAB) FROM(SYMBOLS);                             ZEF02050
  2060   1  0      WRITE FILE(PRODTAB) FROM(PROD);                               ZEF02060
  2070   1  0      WRITE FILE(LALRSZ) FROM(LALR_SZ);                             ZEF02070

  2080   1  0  DCL TBL(2) FIXED BIN(15);                                         ZEF02080
  2090   1  0  DCL C_TBL CHAR(4) BASED (P_TBL);                                  ZEF02090
  2100   1  0  DCL P_TBL PTR;                                                    ZEF02100
  2110   1  0      P_TBL = ADDR (TBL(1));                                        ZEF02110

  2120   1  0      DO I = 1 TO LALR_SZ.M;                                        ZEF02120
  2130   1  1         DO J = 0 TO LALR_SZ.N;                                     ZEF02130
  2140   1  2            TBL(1) = LALR_GOTO (I,J);                               ZEF02140
  2150   1  2            TBL(2) = LALR_ACT (I,J);                                ZEF02150
  2160   1  2            WRITE FILE (LALRTAB) FROM (C_TBL);                      ZEF02160
  2170   1  2         END;                                                       ZEF02170
  2180   1  1      END;                                                          ZEF02180

  2190   1  0      CLOSE FILE(SYMTAB),                                           ZEF02190
                         FILE(PRODTAB),                                          ZEF02200
                         FILE(LALRTAB),                                          ZEF02210
                         FILE(LALRSZ);                                           ZEF02220
```

```
NUMBER LEV NT

  2230   1  0  LR_ITEMS: PROC;                                                  ZEF02230

  2240   2  0  DCL(I,J,K) FIXED BIN(15);                                        ZEF02240
  2250   2  0  DCL (P,N) FIXED DEC(3,0);                                        ZEF02250

                  /* SET UP FIRST ITEM */                                       ZEF02260
  2270   2  0        NEXT_I = 1;                                                ZEF02270
  2280   2  0        CALL ADD_ITEM(0,1,0,'$');                                  ZEF02280
  2290   2  0        ITEMS_SET(1) = 1;                                          ZEF02290
  2300   2  0        N_ITST = 1;                                                ZEF02300

  2310   2  0        CALL CLOSURE;                                              ZEF02310

  2320   2  0        NEWSTATE = 0;                                              ZEF02320
  2330   2  0        STATE = 0;                                                 ZEF02330
  2340   2  0  AGAIN: DO I = 1 TO NO_SYM WHILE(SYMBOLS(I).SYM ¬= ' ');          ZEF02340
  2350   2  1           NEWSTATE = NEWSTATE + 1;                                ZEF02350
  2360   2  1           STATE_ITEMS(NEWSTATE) = 0;                              ZEF02360
  2370   2  1           CALL GOTO (STATE,SYMBOLS(I).SYM,I);                     ZEF02370
  2380   2  1         END;                                                      ZEF02380
  2390   2  0         STATE = STATE + 1;                                        ZEF02390
  2400   2  0         DO I = 1 TO NEXT_I-1;                                     ZEF02400
  2410   2  1            IF ITEMS(I).STATE = STATE                              ZEF02410
                            THEN GOTO AGAIN;                                    ZEF02420
  2430   2  1         END;                                                      ZEF02430

                /* PRINT ITEM TABLE */                                          ZEF02440
  2450   2  0         DO I = 1 TO NEXT_I-1;                                     ZEF02450
  2460   2  1            P = I;                                                 ZEF02460
  2470   2  1            N = ITEMS(I).STATE;                                    ZEF02470
  2480   2  1            J = ITEMS(I).PROD;                                     ZEF02480
  2490   2  1            K = ITEMS(I).POSIT;                                    ZEF02490
  2500   2  1            IF K = 0 THEN NEW_RP = '.' || PROD(J).RP;              ZEF02500
  2510   2  1            IF K >= PROD(J).NO_RP                                  ZEF02510
                            THEN NEW_RP = SUBSTR(PROD(J).RP,1,PROD(J).NO_RP) || '.';   ZEF02520
  2530   2  1               ELSE NEW_RP = SUBSTR(PROD(J).RP,1,K) || '.' ||      ZEF02530
                                  SUBSTR(PROD(J).RP,K+1,PROD(J).NO_RP-K);       ZEF02540
  2550   2  1            NEW_RP = SUBSTR(NEW_RP,1,PROD(J).NO_RP+1) ||           ZEF02550
                            ',' || ITEMS(I).LA;                                 ZEF02560
  2570   2  1            PUT EDIT (P,N,PROD(J).LP,' -> ',NEW_RP)                ZEF02570
                               (COL(1),P'ZZ9',COL(6),P'Z9',COL(15),3 A);        ZEF02580
  2590   2  1         END;                                                      ZEF02590

  2600   2  0  END;    /* LR_ITEMS */                                           ZEF02600
```

```
NUMBER LEV NT

  2610    1  0  CLOSURE: PROC;                                                                  ZEF02610
                        /* SET OF ITEMS TO BE CLOSED IS DEFINED BY ARRAY                        ZEF02620
                           ITEMS_SET AND THE NUMBER OF ITEMS IN THE SET                         ZEF02630
                           IS CONTAINED IN N_ITST                                        */     ZEF02640

  2650    2  0  DCL (I,J,K,L,M) FIXED BIN(15);                                                  ZEF02650

  2660    2  0        DO I = 1 BY 1 UNTIL (I >= N_ITST); /* UNTIL NO MORE ITEMS CAN             ZEF02660
                                                           BE ADDED      */                     ZEF02670
  2680    2  1           M = ITEMS_SET(I);  /* POINTS TO ITEM IN ITEM TABLE */                  ZEF02680
  2690    2  1           CALL FND_PROD(M);                                                      ZEF02690
  2700    2  1           IF N_PROD = 0                                                          ZEF02700
                            THEN GOTO CLOEND;  /* NEXT SYMBOL IN ITEM MUST BE                   ZEF02710
                                                  NONTERMINAL WITH PRODUCTIONS */               ZEF02720
  2730    2  1           CALL MFIRST(M);                                                        ZEF02730
  2740    2  1           DO J = 1 TO N_PROD;                                                    ZEF02740
  2750    2  2              DO K = 1 TO N_FRST;                                                 ZEF02750
  2760    2  3                 CALL ADD_ITEM (ITEMS(M).STATE,PROD_#(J),0,FRST(K));              ZEF02760
  2770    2  3              END;  /* END K LOOP  */                                             ZEF02770
  2780    2  2           END;  /* END J LOOP  */                                                ZEF02780
  2790    2  1  CLOEND:                                                                         ZEF02790
                      END;  /* END I LOOP  */                                                   ZEF02800

  2810    2  0  END;  /* CLOSURE */                                                             ZEF02810
```

```
NUMBER LEV NT

  2820   1  0  FND_PROD: PROC(ITEM);                                                      ZEF02820

                         /* POINTS TO ITEM IN ITEM ARRAY .                                ZEF02830
                            RETURNS LIST OF PRODUCTIONS.    */                            ZEF02840

  2850   2  0  DCL ITEM FIXED BIN(15);                                                    ZEF02850
  2860   2  0  DCL I FIXED BIN(15);                                                       ZEF02860
  2870   2  0  DCL SYMBOL CHAR(1);                                                        ZEF02870

  2880   2  0       N_PROD = 0;                                                           ZEF02880
  2890   2  0       SYMBOL = SUBSTR(PROD(ITEMS(ITEM).PROD).RP,ITEMS(ITEM).POSIT+1,1);     ZEF02890
  2900   2  0       IF (¬TRM(SYMBOL)) | (SYMBOL = ' ')                                    ZEF02900
                       THEN RETURN;  /* SYMBOL EITHER TERMINAL OR BLANK */                ZEF02910
  2920   2  0       DO I = 1 BY 1 WHILE (PROD(I).NO ¬= 0);                                ZEF02920
  2930   2  1          IF SYMBOL = PROD(I).LP                                             ZEF02930
                          THEN DO;                                                        ZEF02940
  2950   2  2                   N_PROD = N_PROD + 1;                                      ZEF02950
  2960   2  2                   PROD_#(N_PROD) = PROD(I).NO;                              ZEF02960
  2970   2  2                END;                                                         ZEF02970
  2980   2  1       END;                                                                  ZEF02980

  2990   2  0  END;  /* FND_PROD  */                                                      ZEF02990
```

```
NUMBER LEV NT

 3000   1  0  MFIRST: PROC (ITEM);                                                    ZEF03000

 3010   2  0  DCL ITEM FIXED BIN(15);                                                 ZEF03010
 3020   2  0        N_FRST = 0;                                                       ZEF03020
 3030   2  0        SYMBOL = SUBSTR(PROD(ITEMS(ITEM).PROD).RP,ITEMS(ITEM).POSIT+2,1);ZEF03030

 3040   2  0        IF SYMBOL = ' '                                                   ZEF03040
                       THEN ;                                                         ZEF03050
 3060   2  0           ELSE CALL FIRST(SYMBOL);                                       ZEF03060
 3070   2  0        IF N_FRST = 0                                                     ZEF03070
                       THEN DO;                                                       ZEF03080
 3090   2  1                 N_FRST = 1;                                              ZEF03090
 3100   2  1                 FRST(1) = ITEMS(ITEM).LA;                                ZEF03100
 3110   2  1                END;                                                      ZEF03110

 3120   2  0  END;  /* MFIRST */                                                      ZEF03120

 3130   1  0  FIRST: PROC (SYM);                                                      ZEF03130

 3140   2  0  DCL SYM CHAR(1);                                                        ZEF03140
 3150   2  0  DCL (I,J) FIXED BIN(15);                                                ZEF03150
 3160   2  0        J = PTR_SYM(SYM);                                                 ZEF03160
 3170   2  0        N_FRST = FIRSTS(J).NO_FIRSTS;                                     ZEF03170
 3180   2  0        DO I = 1 TO FIRSTS(J).NO_FIRSTS;                                  ZEF03180
 3190   2  1           FRST(I) = FIRSTS(J).FIRSTS(I);                                 ZEF03190
 3200   2  1        END;                                                              ZEF03200

 3210   2  0  END; /* FIRST */                                                        ZEF03210
```

```
NUMBER LEV NT

  3220   1  0  PTR_SYM: PROC(SYM);                                          ZEF03220

  3230   2  0  DCL SYM CHAR(1);                                             ZEF03230
  3240   2  0  DCL I FIXED BIN(15);                                         ZEF03240

  3250   2  0        DO I = 1 TO NO_SYM UNTIL(SYMBOLS(I).SYM = SYM); END;   ZEF03250
  3260   2  0        RETURN(I);                                             ZEF03260

  3270   2  0  END; /* PTR_SYM */                                           ZEF03270

  3280   1  0  ADD_TRM: PROC(SYM,I);                                        ZEF03280
                    /*------------------------------------------            ZEF03290
                      ADD SYM TO FIRSTS OF SYMBOL POINTED TO BY I           ZEF03300
                      IF SYMBOL IS NOT ALREADY THERE.                       ZEF03310
                    -------------------------------------------*/           ZEF03320

  3330   2  0  DCL SYM CHAR(1);                                             ZEF03330
  3340   2  0  DCL (I,J,K) FIXED BIN(15);                                   ZEF03340

  3350   2  0        K = FIRSTS(I).NO_FIRSTS;                               ZEF03350
  3360   2  0    IF K > 0 THEN                                              ZEF03360
                     DO J = 1 TO FIRSTS(I).NO_FIRSTS;                       ZEF03370
  3380   2  1           IF SYM = FIRSTS(I).FIRSTS(J)                        ZEF03380
                           THEN RETURN;                                     ZEF03390
  3400   2  1        END;                                                   ZEF03400

  3410   2  0        FIRSTS(I).NO_FIRSTS = K + 1;                           ZEF03410
  3420   2  0        FIRSTS(I).FIRSTS(K+1) = SYM;                           ZEF03420
  3430   2  0        TOT_FIRSTS = TOT_FIRSTS + 1;                           ZEF03430

  3440   2  0  END; /* ADD_TRM */                                           ZEF03440

  3450   1  0  TRM: PROC (SYM) RETURNS ( BIT );                             ZEF03450

  3460   2  0  DCL SYM CHAR(1);                                             ZEF03460
  3470   2  0  DCL I FIXED BIN(15);                                         ZEF03470

  3480   2  0        DO I = 1 TO NO_SYM UNTIL (SYM = SYMBOLS(I).SYM);       ZEF03480
  3490   2  1        END;                                                   ZEF03490

  3500   2  0        RETURN (SYMBOLS(I).TYPE);                              ZEF03500

  3510   2  0  END;   /* TRM */                                             ZEF03510
```

```
NUMBER LEV NT

 3520   1  0  COMP_FIRSTS: PROC;                                                    ZEF03520

 3530   2  0  DCL (I,J,K,L,M,N,P) FIXED BIN(15);                                    ZEF03530
 3540   2  0  DCL SYM CHAR(1);                                                      ZEF03540

 3550   2  0         DO J = 1 TO NO_PROD UNTIL(PROD(J).NO = 0); END;                ZEF03550
 3560   2  0         IF J > 1 THEN N = J -1; ELSE SIGNAL ERROR;                     ZEF03560

 3570   2  0         DO I = 1 TO NO_SYM UNTIL(SYMBOLS(I).SYM = ' '); END;           ZEF03570
 3580   2  0         IF I > 1 THEN M = I-1; ELSE SIGNAL ERROR;                      ZEF03580

 3590   2  0  C_F1:                                                                 ZEF03590
                     P = TOT_FIRSTS;                                                ZEF03600
 3610   2  0         DO I = M TO 1 BY -1;  /* GO THRU SYMBOLS */                    ZEF03610
 3620   2  1            IF TRM(SYMBOLS(I).SYM)  /* IF NONTERMINAL */                ZEF03620
                           THEN                                                     ZEF03630
                              DO J = N TO 1 BY -1;  /* GO THRU PRODUCTIONS */       ZEF03640
 3650   2  2                     IF SYMBOLS(I).SYM = PROD(J).LP                     ZEF03650
                                    THEN                                            ZEF03660
                                       DO;                                          ZEF03670
 3680   2  3                              SYM = SUBSTR(PROD(J).RP,1,1);             ZEF03680
 3690   2  3                              IF TRM(SYM)    /* IF NONTERMINAL */       ZEF03690
                                            THEN                                    ZEF03700
                                              DO;                                   ZEF03710
 3720   2  4                                    K = PTR_SYM(SYM);                   ZEF03720
 3730   2  4                                    IF FIRSTS(K).NO_FIRSTS = 0          ZEF03730
                                                   THEN ;                           ZEF03740
 3750   2  4                                       ELSE DO L = 1 TO FIRSTS(K).NO_FIRSTS; ZEF03750
 3760   2  5                                                 CALL ADD_TRM           ZEF03760
                                                                 (FIRSTS(K).FIRSTS(L),I); ZEF03770
 3780   2  5                                              END;                      ZEF03780
 3790   2  4                                    END;                                ZEF03790
 3800   2  3                                 ELSE CALL ADD_TRM(SYM,I); /* ADD TERMINAL */ ZEF03800
 3810   2  3                              END;                                      ZEF03810
 3820   2  2                        END;                                            ZEF03820
 3830   2  1                     ELSE CALL ADD_TRM(SYMBOLS(I).SYM,I);/* TERMINAL */ ZEF03830
 3840   2  1         END;                                                           ZEF03840
 3850   2  0         IF TOT_FIRSTS = P THEN ; ELSE GOTO C_F1;                       ZEF03850

                /*-----------------------                                           ZEF03860
                   PRINT COMPUTED FIRSTS.                                           ZEF03870
                -----------------------*/                                           ZEF03880

 3890   2  0         PUT EDIT ('COMPUTED FIRSTS:',' ') (A,SKIP,A) SKIP(2);          ZEF03890
 3900   2  0         DO I = 1 BY 1 WHILE(SYMBOLS(I).SYM ¬= ' ');                    ZEF03900
 3910   2  1            PUT EDIT (SYMBOLS(I).SYM,' ') (2 A);                        ZEF03910
 3920   2  1            DO J = 1 TO FIRSTS(I).NO_FIRSTS;                            ZEF03920
 3930   2  2               PUT EDIT (FIRSTS(I).FIRSTS(J)) (A);                      ZEF03930
 3940   2  2            END;                                                        ZEF03940
 3950   2  1            PUT EDIT (' ') (A) SKIP;                                    ZEF03950
 3960   2  1         END;                                                           ZEF03960

 3970   2  0  END; /* COMP_FIRSTS */                                                ZEF03970
```

```
NUMBER LEV NT

  3980   1  0  ADD_ITEM: PROC (STATE,PROD,POSIT,LA);                              ZEF03980

  3990   2  0  DCL (STATE,PROD,POSIT,L,J,K) FIXED BIN(15);                        ZEF03990
  4000   2  0  DCL (P,N) FIXED DEC(3,0);                                          ZEF04000
  4010   2  0  DCL LA CHAR(1);                                                    ZEF04010

                          /* ADD ITEMS TO ITEM TABLE  */                          ZEF04020

  4030   2  0        DO L = 1 TO NEXT_I-1 UNTIL ((ITEMS(L).STATE = STATE)         ZEF04030
                                   & (ITEMS(L).NEXT = -1));                       ZEF04040
                          /* FIND LAST ENTRY FOR STATE  */                        ZEF04050
  4060   2  1           IF (ITEMS(L).STATE = STATE) &                             ZEF04060
                           (ITEMS(L).PROD = PROD) &                               ZEF04070
                           (ITEMS(L).POSIT = POSIT) &                             ZEF04080
                           (ITEMS(L).LA = LA)                                     ZEF04090
                           THEN RETURN; /* ITEM ALREADY EXISTS */                 ZEF04100
  4110   2  1        END;                                                         ZEF04110
  4120   2  0        IF L > NEXT_I-1                                              ZEF04120
                        THEN ;                                                    ZEF04130
  4140   2  0           ELSE ITEMS(L).NEXT = NEXT_I;                              ZEF04140
  4150   2  0        ITEMS(NEXT_I).STATE = STATE;                                 ZEF04150
  4160   2  0        ITEMS(NEXT_I).NEXT = -1;                                     ZEF04160
  4170   2  0        ITEMS(NEXT_I).PROD = PROD;                                   ZEF04170
  4180   2  0        ITEMS(NEXT_I).POSIT = POSIT;                                 ZEF04180
  4190   2  0        ITEMS(NEXT_I).LA = LA;                                       ZEF04190

  4200   2  0        N_ITST = N_ITST + 1;                                         ZEF04200
  4210   2  0        ITEMS_SET(N_ITST) = NEXT_I;                                  ZEF04210
  4220   2  0        NEXT_I = NEXT_I + 1;                                         ZEF04220
  4230   2  0        STATE_ITEMS(STATE) = STATE_ITEMS(STATE) + 1;                 ZEF04230
  4240   2  0        IF STATE > HIGH_STATE                                        ZEF04240
                        THEN HIGH_STATE = STATE;  /* SAVE HIGHEST STATE NO. WITH  ZEF04250
                                                     VALID ITEMS IN IT      */    ZEF04260

  4270   2  0  END;  /* ADD_ITEM  */                                              ZEF04270
```

```
NUMBER LEV NT

 4260   1  0  GOTO: PROC (STATE,X,SYM);                                          ZEF04280

                 /* STATE = CURRENT STATE TO PERFORM GOTO ON.                   ZEF04290
                    X = SYMBOL TO LOOK FOR.                                     ZEF04300
                    SYM = POSITION OF X IN SYMBOL TABLE.  */                    ZEF04310

 4320   2  0  DCL (STATE,SYM,I,J,K,L,S_NI) FIXED BIN(15);                       ZEF04320
 4330   2  0  DCL EQUAL BIT(1);                                                 ZEF04330
 4340   2  0  DCL X CHAR(1);                                                    ZEF04340

 4350   2  0        DO I = 1 TO NEXT_I-1 UNTIL(ITEMS(I).STATE = STATE);         ZEF04350
                        /* FIND FIRST ITEM IN CURRENT STATE */                  ZEF04360
 4370   2  1        END;                                                        ZEF04370
 4380   2  0        S_NI = NEXT_I;  /* SAVE NEXT_I VALUE */                     ZEF04380
 4390   2  0        N_ITST = 0;                                                 ZEF04390

 4400   2  0        DO WHILE(I ¬= -1);  /* DO FOR ALL ITEMS IN STATE */         ZEF04400
 4410   2  1           J = ITEMS(I).PROD;                                       ZEF04410
 4420   2  1           K = ITEMS(I).POSIT + 1;                                  ZEF04420
 4430   2  1           IF SUBSTR (PROD(J).RP,K,1) = X                           ZEF04430
                          THEN CALL ADD_ITEM (NEWSTATE,J,K,ITEMS(I).LA);        ZEF04440
 4450   2  1           I = ITEMS(I).NEXT;                                       ZEF04450
 4460   2  1        END;                                                        ZEF04460

 4470   2  0        IF N_ITST = 0                                               ZEF04470
                       THEN DO;                                                 ZEF04480
 4490   2  1                 NEWSTATE = HIGH_STATE;  /* NEEDED BECAUSE NEWSTATE  */ZEF04490
 4500   2  1                 RETURN;                 /* BLINDLY INCREMENTED BY   */ZEF04500
 4510   2  1              END;                       /* LR_ITEMS.                */ZEF04510

 4520   2  0        CALL CLOSURE;                                               ZEF04520
```

```
NUMBER LEV NT

                          /* CHECK FOR DUPLICATE GOTO STATE  */                    ZEF04530

  4540   2  0                 EQUAL = '0'B;                                            ZEF04540
  4550   2  0                 DO I = 1 TO NEWSTATE-1;                                  ZEF04550
  4560   2  1                    IF STATE_ITEMS(I) = STATE_ITEMS(NEWSTATE)             ZEF04560
                                    THEN                                               ZEF04570
                                       DO;    /* CHECK FOR STATE IDENTITY */           ZEF04580
  4590   2  2                             DO J = 1 TO NEXT_I-1 UNTIL(ITEMS(J).STATE = I); END; ZEF04590
  4600   2  2                             DO WHILE(J ¬= -1);                           ZEF04600
  4610   2  3                                K = S_NI;                                 ZEF04610
  4620   2  3                                DO WHILE(K ¬= -1);                        ZEF04620
  4630   2  4                                   IF (ITEMS(J).PROD = ITEMS(K).PROD) &   ZEF04630
                                                   (ITEMS(J).POSIT = ITEMS(K).POSIT) & ZEF04640
                                                   (ITEMS(J).LA = ITEMS(K).LA)         ZEF04650
                                                   THEN GOTO GOTO2; /* ITEMS = GET NEXT ONE */ ZEF04660
  4670   2  4                                   K = ITEMS(K).NEXT;                     ZEF04670
  4680   2  4                                END;                                      ZEF04680
  4690   2  3                                GOTO GOTO3;  /* STATES (I & NEWSTATE) ¬= */ ZEF04690

  4700   2  3   GOTO2:                       J = ITEMS(J).NEXT;                        ZEF04700
  4710   2  3                             END;                                         ZEF04710
  4720   2  2                             EQUAL = '1'B;  /* STATES (I & NEWSTATE) ARE EQUAL */ ZEF04720
  4730   2  2                             GOTO GOTO4;                                  ZEF04730
  4740   2  2                          END;                                            ZEF04740
  4750   2  1   GOTO3:                                                                 ZEF04750
                              END;                                                     ZEF04760

  4770   2  0   GOTO4:                                                                 ZEF04770
                              IF EQUAL                                                 ZEF04780
                                 THEN DO;                                              ZEF04790
  4800   2  1                          GOTOS(SYM,STATE) = I;                           ZEF04800
  4810   2  1                          NEXT_I = S_NI;          /* BACKOUT NEWSTATE ITEMS */ ZEF04810
  4820   2  1                          NEWSTATE = NEWSTATE - 1;                        ZEF04820
  4830   2  1                          HIGH_STATE = NEWSTATE;                          ZEF04830
  4840   2  1                       END;                                               ZEF04840
  4850   2  0                    ELSE GOTOS(SYM,STATE) = NEWSTATE;                     ZEF04850

  4860   2  0   END;  /* GOTO  */                                                      ZEF04860
```

```
NUMBER LEV NT

  4870   1  0  SR_TAB: PROC;                                                        ZEF04870

  4880   2  0  DCL CHAR CHAR(1);                                                    ZEF04880
  4890   2  0  DCL (I,J,VALUE) FIXED BIN(15);                                       ZEF04890
  4900   2  0  DCL (M,N) FIXED BIN(3,0);                                            ZEF04900

  4910   2  0           DO I = 1 TO NEXT_I-1;                                       ZEF04910
  4920   2  1              IF (ITEMS(I).PROD = 1)  &                                ZEF04920
                              (ITEMS(I).POSIT = PROD(ITEMS(I).PROD).NO_RP)  &       ZEF04930
                              (ITEMS(I).LA = '$')                                   ZEF04940
                              THEN DO;           /* ACCEPT  */                      ZEF04950
  4960   2  2                         CHAR = '$';                                   ZEF04960
  4970   2  2                         VALUE = 0;                                    ZEF04970
  4980   2  2                         GOTO ACTION_FILL;                             ZEF04980
  4990   2  2                      END;                                             ZEF04990

  5000   2  1             IF ITEMS(I).POSIT = PROD(ITEMS(I).PROD).NO_RP             ZEF05000
                             THEN DO;                                               ZEF05010
  5020   2  2                       CHAR = ITEMS(I).LA;                             ZEF05020
  5030   2  2                       VALUE = ITEMS(I).PROD;  /* REDUCE PROD. NO. */  ZEF05030
  5040   2  2                       IF ITEMS(I).LA = '$'                            ZEF05040
                                       THEN GOTO ACTION_FILL;                       ZEF05050
  5060   2  2                       IF ¬TRM(ITEMS(I).LA)     /* MUST BE TERMINAL */ ZEF05060
                                       THEN GOTO ACTION_FILL;                       ZEF05070
  5080   2  2                     END;                                              ZEF05080

  5090   2  1             CHAR = SUBSTR(PROD(ITEMS(I).PROD).RP,ITEMS(I).POSIT+1,1); ZEF05090
  5100   2  1             IF ¬TRM(CHAR)            /* IF CHAR IS A TERMINAL  */     ZEF05100
                             THEN DO;                                               ZEF05110
  5120   2  2                       VALUE = -2;     /* SHIFT */                     ZEF05120
  5130   2  2                       GOTO ACTION_FILL;                               ZEF05130
  5140   2  2                     END;                                              ZEF05140

  5150   2  1             GOTO END_DO;                                              ZEF05150

  5160   2  1  ACTION_FILL:                                                         ZEF05160
                          DO J = 1 BY 1 UNTIL ((SYMBOLS(J).SYM = CHAR)  |           ZEF05170
  5180   2  2                                  (SYMBOLS(J).SYM = ' '));   END;      ZEF05180
                             /* IF SYM = ' ' THEN CHAR MUST = '$'  */               ZEF05190
  5200   2  1             IF (ACTION(J,ITEMS(I).STATE) = -1)  |                     ZEF05200
                             (ACTION(J,ITEMS(I).STATE) = VALUE)                     ZEF05210
                             THEN ACTION(J,ITEMS(I).STATE) = VALUE;                 ZEF05220
  5230   2  1                ELSE DO;          /* ERROR - ALREADY FILLED IN */      ZEF05230
  5240   2  2                       M = ITEMS(I).STATE;                             ZEF05240
  5250   2  2                       N = I;                                          ZEF05250
  5260   2  2                       PUT EDIT ('ACTION CONFLICT: STATE = ',M,        ZEF05260
                                              ' CHARACTER = ',CHAR,' ITEM = ',N)    ZEF05270
                                             (SKIP,A,P'Z9',3 A,P'Z9');              ZEF05280
  5290   2  2                     END;                                              ZEF05290
  5300   2  1  END_DO:                                                              ZEF05300
                       END;                                                         ZEF05310
  5320   2  0  END;    /* SR_TAB */                                                 ZEF05320
```

```
NUMBER LEV NT

 5330   1  0  GRAMMAR: PROC;                                                        ZEF05330

                  /*----------------------------------------------              ZEF05340
                    GRAMMAR DEFINITION:                                           ZEF05350
                     VALID TC'S ARE:                                              ZEF05360
                        NONT FOR NONTERMINALS SEPARATED BY BLANKS                 ZEF05370
                                 FIRST NONTERMINAL IS START SYMBOL                ZEF05380
                        TERM FOR TERMINALS SEPARATED BY BLANKS                    ZEF05390
                        PROD FOR PRODUCTIONS - 1 PER RECORD                       ZEF05400
                                 <LEFT PART> : <RIGHT PART>                       ZEF05410
                  ----------------------------------------------*/              ZEF05420

 5430   2  0  DCL CARD CHAR(80);                                                    ZEF05430
 5440   2  0  DCL A_CARD(80) CHAR(1) BASED(P_CARD);                                 ZEF05440
 5450   2  0  DCL P_CARD PTR;                                                       ZEF05450
 5460   2  0        P_CARD = ADDR(CARD);                                            ZEF05460
 5470   2  0  DCL (I,J,K,L,M) FIXED BIN(15);                                        ZEF05470
 5480   2  0  DCL REP_SYM CHAR(40) STATIC INIT                                      ZEF05480
                    ('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789)(*&');                   ZEF05490
 5500   2  0  DCL REPSYM(40) CHAR(1) BASED (P_RS);                                  ZEF05500
 5510   2  0  DCL P_RS PTR;                                                         ZEF05510
 5520   2  0        P_RS = ADDR (REP_SYM);                                          ZEF05520

 5530   2  0        READ FILE (IN) INTO (CARD);                                     ZEF05530
 5540   2  0        I = 0;                                                          ZEF05540
 5550   2  0        J = 0;                                                          ZEF05550
 5560   2  0        DO WHILE (MORE);                                                ZEF05560
 5570   2  1           PUT EDIT (CARD) (SKIP,A);                                    ZEF05570
 5580   2  1           SELECT (SUBSTR(CARD,1,5));                                   ZEF05580

 5590   2  2              WHEN ('NONT ','TERM ')  /* NONTERMINAL & TERMINALS */     ZEF05590
                             DO;                                                    ZEF05600
 5610   2  3                    L = 6;                                              ZEF05610
 5620   2  3   GNONT:                                                               ZEF05620
 5630   2  4                    DO K = L BY 1 UNTIL(A_CARD(K) ¬= ' '); END;         ZEF05630
 5640   2  3                    IF K > 80                                           ZEF05640
                                   THEN LEAVE;  /* DONE WITH SCAN */                ZEF05650
 5660   2  3                    DO L = K BY 1 UNTIL(A_CARD(L) = ' '); END;          ZEF05660
 5670   2  3                    IF L > 80                                           ZEF05670
                                   THEN L = 80;  /* ASSUME MAX VALUE */             ZEF05680
 5690   2  3                    I = I + 1;                                          ZEF05690
 5700   2  3                    SUBSTR(SYMBOLS(I).FULL,1,L-K) = SUBSTR(CARD,K,L-K); ZEF05700
 5710   2  3                    SYMBOLS(I).SYM = REPSYM(RSNO);                      ZEF05710
 5720   2  3                    SYMBOLS(I+1).SYM = ' ';                             ZEF05720
 5730   2  3                    RSNO = RSNO + 1;                                    ZEF05730
 5740   2  3                    IF SUBSTR(CARD,1,5) = 'NONT '                       ZEF05740
                                  THEN SYMBOLS(I).TYPE = '1'B;                      ZEF05750
                                  ELSE SYMBOLS(I).TYPE = '0'B;                      ZEF05760
 5760   2  3                    SYMBOLS(I).L_FULL = L-K;                            ZEF05770
 5770   2  3                    GOTO GNONT;                                         ZEF05780
 5780   2  3                 END;                                                   ZEF05790
 5790   2  3
 5800   2  2              WHEN ('PROD ')               /* PRODUCTION */             ZEF05800
                             DO;                                                    ZEF05810
```

```
NUMBER LEV NT

  5820   2  3                    M=0;                                                          ZEF05820
  5830   2  3                    J = J + 1;                                                    ZEF05830
  5840   2  3                    PROD(J).NO = J;                                               ZEF05840
  5850   2  3                    PROD(J+1).NO = 0;                                             ZEF05850
  5860   2  3                    PROD(J).RP = ' ';                                             ZEF05860
  5870   2  3                    DO K = 6 TO 80 UNTIL(A_CARD(K) ¬= ' '); END;                  ZEF05870
  5880   2  3                    DO L = 6 TO 80 UNTIL(A_CARD(L) = ':'); END;                   ZEF05880
  5890   2  3                    IF (K = 80) | (L = 80)                                        ZEF05890
                                    THEN DO;                                                   ZEF05900
  5910   2  4                            PUT EDIT ('INVALID PRODUCTION.') (A) SKIP;ZEF05910
  5920   2  4                            GOTO GPEND;                                           ZEF05920
  5930   2  4                         END;                                                     ZEF05930
  5940   2  3                    DO L = L BY -1 UNTIL(A_CARD(L) ¬= ' '); END;                  ZEF05940
  5950   2  3                    PROD(J).LP = INT(SUBSTR(CARD,K,L-K+1));                       ZEF05950
  5960   2  3                    L = L + 1;                                                    ZEF05960
  5970   2  3  GPROD:                                                                          ZEF05970
                                 DO L = L BY 1 UNTIL ((A_CARD(L) ¬= ' ')  &                    ZEF05980
  5990   2  4                                 (A_CARD(L) ¬= ':')); END;                        ZEF05990
  6000   2  3                    IF L >= 80 THEN LEAVE;                                        ZEF06000
  6010   2  3                    DO K = L BY 1 UNTIL(A_CARD(K) = ' '); END;                    ZEF06010
  6020   2  3                    M = M+1;                                                      ZEF06020
  6030   2  3                    SUBSTR(PROD(J).RP,M,1) = INT(SUBSTR(CARD,L,K-L));             ZEF06030
  6040   2  3                    L = K;                                                        ZEF06040
  6050   2  3                    PROD(J).NO_RP = M;                                            ZEF06050
  6060   2  3                    GOTO GPROD;                                                   ZEF06060
  6070   2  3  GPEND:                                                                          ZEF06070
                              END;                                                             ZEF06080

  6090   2  2            OTHERWISE PUT EDIT ('UNRECOGNIZED TC') (SKIP,A);                      ZEF06090

  6100   2  2         END;   /* END SELECT */                                                  ZEF06100

  6110   2  1         READ FILE (IN) INTO (CARD);                                              ZEF06110

  6120   2  1      END;                                                                        ZEF06120

  6130   2  0  END;    /* GRAMMAR */                                                           ZEF06130
```

```
NUMBER LEV NT

  6140   1  0  INT: PROC(GSYM) RETURNS(CHAR(1));                                   ZEF06140

               /*---------------------------------------------------------        ZEF06150
                 GIVEN FULL GRAMMAR SYMBOL THIS PROCEDURE RETURNS INTERNAL        ZEF06160
                 GRAMMAR SYMBOL.                                                  ZEF06170
               ----------------------------------------------------------*/       ZEF06180

  6190   2  0  DCL GSYM CHAR(30);                                                 ZEF06190
  6200   2  0  DCL SYM CHAR(1);                                                   ZEF06200
  6210   2  0  DCL (I,J) FIXED BIN(15);                                           ZEF06210

  6220   2  0        DO I = 1 TO NO_SYM;                                          ZEF06220
  6230   2  1          J = SYMBOLS(I).L_FULL;                                     ZEF06230
  6240   2  1          IF SUBSTR(SYMBOLS(I).FULL,1,J) = SUBSTR(GSYM,1,J)          ZEF06240
                          THEN DO;                                                ZEF06250
  6260   2  2                    SYM = SYMBOLS(I).SYM;                            ZEF06260
  6270   2  2                    RETURN (SYM);                                    ZEF06270
  6280   2  2                  END;                                               ZEF06280
  6290   2  1        END;                                                         ZEF06290
  6300   2  0        PUT EDIT ('UNDEFINED GRAMMAR SYMBOL: ',GSYM)                 ZEF06300
                              (2 A) SKIP;                                         ZEF06310
  6320   2  0        SYM = 'X';                                                   ZEF06320
  6330   2  0        RETURN (SYM);                                                ZEF06330

  6340   2  0  END;  /* INT */                                                    ZEF06340

  6350   1  0  ALTER: PROC (ONE,TWO);                                             ZEF06350
                         /* STATE ONE AND STATE TWO ARE EQUAL.                    ZEF06360
                            MERGE ACTION ENTRIES FOR STATE TWO INTO STATE ONE. */ ZEF06370

  6380   2  0  DCL (ONE,TWO,I) FIXED BIN(15);                                     ZEF06380

  6390   2  0        DO I = 1 BY 1 UNTIL (SYMBOLS(I).SYM = ' ');                  ZEF06390
  6400   2  1          IF ACTION (I,ONE) = -1  /* UNDEFINED SLOT */               ZEF06400
                          THEN ACTION (I,ONE) = ACTION (I,TWO);                   ZEF06410
  6420   2  1        END;                                                         ZEF06420

  6430   2  0  END;   /* ALTER */                                                 ZEF06430
```

```
NUMBER LEV NT

 6440   1  0  LALR: PROC;                                                             ZEF06440

 6450   2  0  DCL TRANSFORM(0:25) FIXED BIN(15) STATIC INIT (                        ZEF06450
                   0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,             ZEF06460
                   21,22,23,24,25);                                                   ZEF06470
 6480   2  0  DCL LALR_ST FIXED BIN(15);                                              ZEF06480
 6490   2  0  DCL DELETE(15) FIXED BIN(15) STATIC INIT ((15) 32767);                  ZEF06490

 6500   2  0  DCL (I,J,K,L,M,N) FIXED BIN(15);                                        ZEF06500
 6510   2  0          PUT EDIT ('LALR TRANSFORMATION') (A) PAGE;                      ZEF06510
 6520   2  0          LALR_ST = STATE;      /* GET CURRENT TOTAL NO. OF STATES */     ZEF06520
 6530   2  0          DO I = 0 BY 1 WHILE (I < STATE);                                ZEF06530
 6540   2  1              DO J = I+1 BY 1 WHILE (J < STATE);                          ZEF06540
 6550   2  2                  IF (TRANSFORM(I) = I)  &                                ZEF06550
                                 (TRANSFORM(J) = J)                                   ZEF06560
                                 THEN IF IDENT(I,J)     /* CHECK FOR STATE IDENTITY */ ZEF06570
                                          THEN DO;                                    ZEF06580
 6590   2  3                                     TRANSFORM(J) = I;                    ZEF06590
 6600   2  3                                     CALL ALTER(I,J);  /* ALTER ACTION */ ZEF06600
 6610   2  3                                     LALR_ST = LALR_ST - 1;               ZEF06610
 6620   2  3                                  END;                                    ZEF06620
 6630   2  2              END;                                                        ZEF06630
 6640   2  1          END;                                                            ZEF06640

                          /* DETERMINE DELETED STATES */                              ZEF06650
 6660   2  0          K = 1;                                                          ZEF06660
 6670   2  0          DO I = 0 TO STATE;                                              ZEF06670
 6680   2  1              IF TRANSFORM(I) = I                                         ZEF06680
                              THEN ;                                                  ZEF06690
 6700   2  1                  ELSE          /* I -> EQUIVALENT STATE */               ZEF06700
                                  DO;                                                 ZEF06710
 6720   2  2                          DELETE(K) = I;                                  ZEF06720
 6730   2  2                          K = K+1;                                        ZEF06730
 6740   2  2                          DELETE(K) = -1;                                 ZEF06740
 6750   2  2                      END;                                                ZEF06750
 6760   2  1          END;                                                            ZEF06760

 6770   2  0          PUT EDIT ('LR(1) STATES TO DELETE:') (A) SKIP(2);               ZEF06770

 6780   2  0          PUT EDIT (' ') (A) SKIP;                                        ZEF06780
 6790   2  0          DO J = 1 TO 15 WHILE (DELETE(J) ¬= -1);                         ZEF06790
 6800   2  1              PUT EDIT (DELETE(J)) (P'ZZ9');                              ZEF06800
 6810   2  1          END;                                                            ZEF06810
 6820   2  0          PUT EDIT ('LR TO LALR STATE TRANSFORM:') (A) SKIP(2);           ZEF06820
 6830   2  0          PUT EDIT ('LR   ') (A) SKIP;                                    ZEF06830
 6840   2  0          DO J = 0 TO NEWSTATE;                                           ZEF06840
 6850   2  1              PUT EDIT (J) (P'ZZ9');                                      ZEF06850
 6860   2  1          END;                                                            ZEF06860

                          /* FILL DELETED STATES WITH VALID STATES */                 ZEF06870
 6880   2  0          K = 1;                                                          ZEF06880
 6890   2  0          DO I = 0 TO STATE-1;                                            ZEF06890
 6900   2  1              IF TRANSFORM(I) ¬= I                                        ZEF06900
                              THEN ;     /* DELETED STATE - NO SWEAT */               ZEF06910
```

```
NUMBER LEV NT

  6920   2  1              ELSE                                                         ZEF06920
                              IF TRANSFORM(I) < LALR_ST                                 ZEF06930
                                 THEN ;  /* STATE WITHIN NEW LIMITS */                  ZEF06940
  6950   2  1                    ELSE    /* BEYOND UPPER LIMIT - USE VALID   */         ZEF06950
                                    DO;  /* DELETED STATE.                   */         ZEF06960
  6970   2  2                          IF DELETE(K) >= LALR_ST                          ZEF06970
                                          THEN DO;                                      ZEF06980
  6990   2  3                                    PUT EDIT ('5440',K) (A,P'Z9') SKIP;    ZEF06990
  7000   2  3                                    SIGNAL ERROR;                          ZEF07000
  7010   2  3                                 END;                                      ZEF07010
  7020   2  2                          L = TRANSFORM(I);                                ZEF07020
  7030   2  2                          DO J = 0 TO STATE;                               ZEF07030
  7040   2  3                             IF TRANSFORM(J) = L                           ZEF07040
                                             THEN TRANSFORM(J) = DELETE(K);             ZEF07050
  7060   2  3                          END;                                             ZEF07060
  7070   2  2                          GOTOS(*,DELETE(K)) = GOTOS(*,L);                 ZEF07070
  7080   2  2                          ACTION(*,DELETE(K)) = ACTION(*,L);               ZEF07080
  7090   2  2                          K = K+1;                                         ZEF07090
  7100   2  2                       END;                                                ZEF07100
  7110   2  1          END;                                                             ZEF07110
  7120   2  0          PUT EDIT ('LALR') (A) SKIP;                                      ZEF07120
  7130   2  0          DO J = 0 TO NEWSTATE;                                            ZEF07130
  7140   2  1             PUT EDIT (TRANSFORM(J)) (P'ZZ9');                             ZEF07140
  7150   2  1          END;                                                             ZEF07150
  7160   2  0          DO I = 1 TO NO_SYM UNTIL (SYMBOLS(I).SYM = ' '); END;            ZEF07160
  7170   2  0          LALR_SZ.M = I;                                                   ZEF07170
  7180   2  0          LALR_SZ.N = LALR_ST - 1;                                         ZEF07180
  7190   2  0          ALLOCATE LALR_GOTO;                                              ZEF07190
  7200   2  0          ALLOCATE LALR_ACT;                                               ZEF07200

                          /* FILL IN ENTRIES IN LALR ARRAYS */                          ZEF07210
  7220   2  0          DO I = 1 TO LALR_SZ.M;                                           ZEF07220
  7230   2  1             DO J = 0 TO LALR_SZ.N;                                        ZEF07230
  7240   2  2                IF GOTOS(I,J) >= 0                                         ZEF07240
                                THEN LALR_GOTO (I,J) = TRANSFORM (GOTOS(I,J));          ZEF07250
  7260   2  2                   ELSE LALR_GOTO (I,J) = GOTOS (I,J);                     ZEF07260
  7270   2  2                IF ACTION (I,J) >= 0                                       ZEF07270
                                THEN LALR_ACT (I,J) = TRANSFORM (ACTION(I,J));          ZEF07280
  7290   2  2                   ELSE LALR_ACT (I,J) = ACTION (I,J);                     ZEF07290
  7300   2  2             END;                                                          ZEF07300
  7310   2  1          END;                                                             ZEF07310

  7320   2  0  END;    /* LALR */                                                       ZEF07320
```

```
NUMBER LEV NT

  7330   1  0  IDENT: PROC (FIRST,SECOND) RETURNS (BIT);                  ZEF07330

                    /* RETURN '1' IF FIRST STATE IDENTICALLY               ZEF07340
                       EQUALS SECOND STATE.                                ZEF07350
                       CHECK FOR IDENTITY IF ITEM HAS SAME PRODUCTION      ZEF07360
                       AND SAME NO. OF RECOGNIZED POSITIONS.      */       ZEF07370

  7380   2  0  DCL (FIRST,SECOND,I,J,K,L,M) FIXED BIN(15);                 ZEF07380

  7390   2  0          DO I = 1 TO NEXT_I-1 WHILE (ITEMS(I).STATE ¬= FIRST); END;    ZEF07390
  7400   2  0          K = I;                                              ZEF07400
  7410   2  0          DO J = 1 TO NEXT_I-1 WHILE (ITEMS(J).STATE ¬= SECOND); END;   ZEF07410
  7420   2  0          L = J;                                              ZEF07420
  7430   2  0          DO M = 1 TO 2;                                      ZEF07430
  7440   2  1             DO UNTIL (I = -1);                               ZEF07440
  7450   2  2                IF M = 1                                      ZEF07450
                                THEN J = L;                                ZEF07460
  7470   2  2                   ELSE J = K;                                ZEF07470
  7480   2  2                DO UNTIL (J = -1);                            ZEF07480
  7490   2  3                   IF (ITEMS(I).PROD = ITEMS(J).PROD)  &      ZEF07490
                                   (ITEMS(I).POSIT = ITEMS(J).POSIT)       ZEF07500
                                   THEN GOTO IDENT1;                       ZEF07510
  7520   2  3                   IF J = -1                                  ZEF07520
                                   THEN ;                                  ZEF07530
  7540   2  3                      ELSE  J = ITEMS(J).NEXT;                ZEF07540
  7550   2  3                END;                                          ZEF07550
  7560   2  2                RETURN ('0'B);  /* STATES NOT EQUAL */        ZEF07560

  7570   2  2  IDENT1:                                                     ZEF07570
                             IF I = -1                                     ZEF07580
                                THEN ;                                     ZEF07590
  7600   2  2                   ELSE I = ITEMS(I).NEXT;                    ZEF07600
  7610   2  2             END;                                             ZEF07610
  7620   2  1             I = L;      /* NOW ENSURE ALL ITEMS IN SECOND STATE  */   ZEF07620
  7630   2  1             J = K;      /* ARE CONTAINED IN FIRST STATE.         */   ZEF07630
  7640   2  1          END;                                                ZEF07640
  7650   2  0          RETURN ('1'B);                                      ZEF07650

  7660   2  0  END; /* IDENT */                                            ZEF07660

  7670   1  0  END;   /* APARSE */                                         ZEF07670
```

```
OPTIONS SPECIFIED
MAR(2 72 1) NUM GN MACRO NIS TERM;

OPTIONS USED
AGGREGATE         NOCOUNT           ATTRIBUTES(SHORT)
COMPILE           NODECK            CHARSET(60,EBCDIC)
GONUMBER          NOESD             FLAG(I)
LMESSAGE          NOFLOW            LINECOUNT(60)
MACRO             NOGOSTMT          MARGINS(2,72,1)
MAP               NOIMPRECISE       OPTIMIZE(TIME)
NEST              NOINCLUDE         SEQUENCE(73,80)
NUMBER            NOINTERRUPT       SIZE(327440)
OBJECT            NOINSOURCE        NOSYNTAX(S)
OFFSET            NOLIST            XREF(SHORT)
OPTIONS           NOMARGINI         TERMINAL(NOAGGREGATE,
SOURCE            NOMDECK                    NOATTRIBUTES,
STORAGE           NOSTMT                     NOESD,
                                             NOINSOURCE,
                                             NOLIST,
                                             NOMAP,
                                             NOOFFSET,
                                             NOOPTIONS,
                                             NOSOURCE,
                                             NOSTORAGE,
                                             NOXREF)

NO PREPROCESSOR DIAGNOSTIC MESSAGES PRODUCED
```

SOURCE LISTING

```
NUMBER LEV NT                                                                                    R

                         /*  PARSER   */                                          ZEF00010
  20        0  ZEFSDL:  PROC OPTIONS(MAIN) REORDER;                               ZEF00020

                /*-------------------                                           ZEF00030
                   FILE DEFINITIONS                                             ZEF00040
                ---------------------*/                                         ZEF00050
  60    1   0  DCL SYSPRINT STREAM OUTPUT PRINT FILE                              ZEF00060
                         ENV ( VB RECSIZE(121) BLKSIZE(125));                  ZEF00070
  80    1   0  DCL STRINGS FILE RECORD INPUT;   /* INPUT STRINGS FROM GRAMMAR */  ZEF00080
  90    1   0  DCL SYMTAB FILE RECORD INPUT;    /* SYMBOL TABLE */                ZEF00090
 100    1   0  DCL PRODTAB FILE RECORD INPUT;    /* PRODUCTIONS TABLE */          ZEF00100
 110    1   0  DCL LALRSZ FILE RECORD INPUT;    /* LALR TABLE SIZES */            ZEF00110
 120    1   0  DCL LALRTAB FILE RECORD INPUT;    /* LALR TABLES */                ZEF00120

 150    1   0  DCL NO_SYM FIXED BIN(15) STATIC INIT(      40);                    ZEF00150  1

 160    1   0  DCL 1 SYMBOLS (       40) STATIC,                                  ZEF00160  1
                      2 SYM CHAR(1) INIT('R','S','C','B','D',' '),              ZEF00170
                      2 TYPE BIT(1) INIT('1','1','1','0','0'),                  ZEF00180
                       2 L_FULL FIXED BIN(15),                                  ZEF00190
                       2 FULL CHAR(30);                                         ZEF00200

 230    1   0  DCL NO_PROD FIXED BIN(15) STATIC INIT(      20);                   ZEF00230  1

 240    1   0  DCL 1 PROD (       20) STATIC,                                     ZEF00240  1
                      2 NO FIXED BIN(15) INIT(1,2,3,4,0),                       ZEF00250
                      2 LP CHAR(1) INIT('R','S','C','C'),                       ZEF00260
                      2 NO_RP FIXED BIN(15) INIT(1,2,2,1),                      ZEF00270
                      2 RP CHAR(6) INIT('S','CC','BC','D');                     ZEF00280

 290    1   0  DCL 1 LALR_SZ,        /* LALR GOTO & ACTION ARRAYS */              ZEF00290
                      2 M FIXED BIN(15),    /* NO. OF GRAMMAR SYMBOLS  */       ZEF00300
                      2 N FIXED BIN(15);    /* NO. OF STATES */                 ZEF00310

 320    1   0  DCL GOTOS (LALR_SZ.M,0:LALR_SZ.N) FIXED BIN (15) CTL;              ZEF00320
 330    1   0  DCL ACTION (LALR_SZ.M,0:LALR_SZ.N) FIXED BIN (15) CTL;             ZEF00330

 340    1   0  DCL STACK(20) FIXED BIN(15) STATIC INIT(0);   /* PROCESSING STACK */ ZEF00340
 350    1   0  DCL SPTR FIXED BIN(15) STATIC INIT(1);        /* STACK POINTER */  ZEF00350

 360    1   0  DCL INPUT CHAR(80) STATIC INIT(' ');                               ZEF00360
 370    1   0  DCL A_INPUT(80) CHAR(1) BASED(P_INP);                              ZEF00370
 380    1   0  DCL P_INP PTR INIT(ADDR(INPUT));                                   ZEF00380

 390    1   0  DCL SYM CHAR(1);                                                   ZEF00390
 400    1   0  DCL N_SYM FIXED BIN(15);                                           ZEF00400
```

```
NUMBER LEV NT                                                                      R

                  /* MISC DECLARATIONS */                               ZEF00410
   420    1  0  DCL (I,J,K,L) FIXED BIN(15);                            ZEF00420
   430    1  0  DCL (M,N,P) FIXED DEC(3,0);                             ZEF00430

   440    1  0  DCL ADDR BUILTIN;                                       ZEF00440
   450    1  0  DCL (MORE_IN,NEW_STR) BIT(1) STATIC INIT('1'B);         ZEF00450
```

```
NUMBER LEV NT                                                                               R

                   /* ON UNITS AND INITIALIZATION */                                ZEF00460

  470   1  0       ON SUBRG SIGNAL ERROR;                                           ZEF00470
  480   1  0       ON STRG SIGNAL ERROR;                                            ZEF00480

  490   1  0       ON ERROR SNAP                                                    ZEF00490
                      BEGIN;                                                        ZEF00500
  510   2  0          END;                                                          ZEF00510

  520   1  0       ON ENDFILE(STRINGS)                                              ZEF00520
                      BEGIN;                                                        ZEF00530
  540   2  0             A_INPUT(1) = '$';                                          ZEF00540
  550   2  0             MORE_IN = '0'B;                                            ZEF00550
  560   2  0          END;                                                          ZEF00560

  570   1  0       OPEN FILE(SYSPRINT),                                             ZEF00570
                        FILE(STRINGS),                                              ZEF00580
                        FILE(SYMTAB),                                               ZEF00590
                        FILE(PRODTAB),                                              ZEF00600
                        FILE(LALRTAB),                                              ZEF00610
                        FILE(LALRSZ);                                               ZEF00620

  630   1  0       READ FILE(SYMTAB) INTO(SYMBOLS);                                 ZEF00630
  640   1  0       DO I = 1 TO NO_SYM UNTIL(SYMBOLS(I).SYM = ' '); END;             ZEF00640
  650   1  0       SYMBOLS(I).SYM = '$';                                            ZEF00650
  660   1  0       SYMBOLS(I+1).SYM = ' ';                                          ZEF00660
  670   1  0       SYMBOLS(I).TYPE = '0'B;                                          ZEF00670
  680   1  0       READ FILE(PRODTAB) INTO(PROD);                                   ZEF00680
  690   1  0       READ FILE(LALRSZ) INTO(LALR_SZ);                                 ZEF00690
  700   1  0       ALLOCATE GOTOS;                                                  ZEF00700
  710   1  0       ALLOCATE ACTION;                                                 ZEF00710

  720   1  0  DCL TBL(2) FIXED BIN(15);                                             ZEF00720
  730   1  0  DCL C_TBL CHAR(4) BASED (P_TBL);                                      ZEF00730
  740   1  0  DCL P_TBL PTR;                                                        ZEF00740
  750   1  0      P_TBL = ADDR (TBL(1));                                            ZEF00750

  760   1  0       DO I = 1 TO LALR_SZ.M;                                           ZEF00760
  770   1  1          DO J = 0 TO LALR_SZ.N;                                        ZEF00770
  780   1  2             READ FILE (LALRTAB) INTO (C_TBL);                          ZEF00780
  790   1  2             GOTOS (I,J) = TBL(1);                                      ZEF00790
  800   1  2             ACTION (I,J) = TBL(2);                                     ZEF00800
  810   1  2          END;                                                          ZEF00810
  820   1  1       END;                                                             ZEF00820

  830   1  0       CLOSE FILE(SYMTAB),                                              ZEF00830
                         FILE(PRODTAB),                                             ZEF00840
                         FILE(LALRTAB),                                             ZEF00850
                         FILE(LALRSZ);                                              ZEF00860
```

```
NUMBER LEV NT                                                              R

                      /*----------------------
                         MAIN PROCESSING
                      -----------------------*/

  900    1  0         CALL GETSYM;
  910    1  0         DO WHILE(MORE_IN);
  920    1  1          SELECT (ACTION(N_SYM,STACK(SPTR)));

  930    1  2            WHEN (-1)
                            DO;
  950    1  3                 PUT EDIT ('    INVALID STRING FOR THIS GRAMMAR',
                                        '    INPUT STRING FLUSHED')
                                       (A,SKIP,A) SKIP;
  980    1  3                 DO WHILE(SYM ¬= '$');
  990    1  4                    CALL GETSYM;
 1000    1  4                 END;
 1010    1  3                 CALL GETSYM;
 1020    1  3             END;

 1030    1  2            WHEN (0)
                            DO;
 1050    1  3                 PUT EDIT ('    REDUCE USING PRODUCTION NO. ',1)
                                       (A,P'ZZ9') SKIP;
 1070    1  3                 PUT EDIT ('    VALID INPUT STRING') (A) SKIP;
 1080    1  3                 CALL GETSYM;
 1090    1  3             END;

 1100    1  2            WHEN (-2)    /* SHIFT */
                            DO;
 1120    1  3                CALL PUSH(N_SYM);
 1130    1  3                CALL PUSH(GOTOS(N_SYM,STACK(SPTR-1)));
 1140    1  3                M = STACK(SPTR);
 1150    1  3                PUT EDIT ('    STACK INPUT CHARACTER AND STATE ',M)
                                      (A,P'ZZ9') SKIP;
 1170    1  3                CALL GETSYM;
 1180    1  3             END;

 1190    1  2            OTHERWISE          /* REDUCE */
                            DO;
 1210    1  3                I = ACTION(N_SYM,STACK(SPTR));/* REDUCE PROD  NO. */
 1220    1  3                J = PROD(I).NO_RP * 2;
 1230    1  3                SPTR = SPTR - J;  /* POP OFF STACK J ELEMENTS */
 1240    1  3                K = PTR_SYM(PROD(I).LP);
 1250    1  3                CALL PUSH(K);
 1260    1  3                CALL PUSH(GOTOS(K,STACK(SPTR-1)));
 1270    1  3     IF STACK(SPTR) < 1
                    THEN DO;
 1290    1  4             PUT EDIT('  INVALID STRING FOR THIS GRAMMAR',
                                   '  INPUT STRING FLUSHED')
                                  (A,SKIP,A);
 1320    1  4             DO WHILE(SYM ¬= '$');
 1330    1  5               CALL GETSYM;
 1340    1  5             END;
 1350    1  4             CALL GETSYM;
```

```
NUMBER LEV NT                                                                                           R

  1360   1  4          END;                                                          ZEF01360
  1370   1  3                M = I;                                                  ZEF01370
  1380   1  3                PUT EDIT ('    REDUCE USING PRODUCTION NO. ',M)         ZEF01380
                                      (A,P'ZZ9') SKIP;                               ZEF01390
  1400   1  3             END;                                                       ZEF01400
  1410   1  2         END; /* SELECT */                                              ZEF01410
  1420   1  1        END;                                                            ZEF01420
```

```
NUMBER LEV NT                                                                                  R

                    /*------------                                           ZEF01430
                      PUSH PROC                                              ZEF01440
                    --------------*/                                         ZEF01450

  1460   1  0  PUSH: PROC(VAL);                                              ZEF01460

  1470   2  0  DCL VAL FIXED BIN(15);                                        ZEF01470

  1480   2  0          SPTR = SPTR + 1;                                      ZEF01480
  1490   2  0          STACK(SPTR) = VAL;                                    ZEF01490

  1500   2  0  END; /* PUSH */                                               ZEF01500

                    /*---------------                                        ZEF01510
                      PTR_SYM PROC                                           ZEF01520
                    -----------------*/                                      ZEF01530

  1540   1  0  PTR_SYM: PROC(SYM);                                           ZEF01540

  1550   2  0  DCL SYM CHAR(1);                                              ZEF01550
  1560   2  0  DCL I FIXED BIN(15);                                          ZEF01560

  1570   2  0          DO I = 1 TO NO_SYM UNTIL(SYMBOLS(I).SYM = SYM); END;  ZEF01570
  1580   2  0          RETURN(I);                                            ZEF01580

  1590   2  0  END; /* PTR_SYM */                                            ZEF01590
```

```
NUMBER LEV NT                                                                         R

                      /*--------------                                  ZEF01600
                        GETSYM PROC                                     ZEF01610
                      ---------------*/                                 ZEF01620

 1630   1  0  GETSYM: PROC;                                             ZEF01630

 1640   2  0  DCL I FIXED BIN(15);                                      ZEF01640

 1650   2  0        IF NEW_STR                                          ZEF01650
                       THEN DO;                                         ZEF01660
 1670   2  1                   PUT EDIT ('INPUT STRING:') (A) PAGE;     ZEF01670
 1680   2  1                   NEW_STR = '0'B;                          ZEF01680
 1690   2  1                   STACK(1) = 0;                            ZEF01690
 1700   2  1                   SPTR = 1;                                ZEF01700
 1710   2  1                END;                                        ZEF01710

 1720   2  0  GETC:                                                     ZEF01720
 1730   2  1        DO I = 1 TO 80 UNTIL(A_INPUT(I) ¬= ' '); END;       ZEF01730
 1740   2  0        IF I >= 80                                          ZEF01740
                       THEN                                             ZEF01750
                          DO;                                           ZEF01760
 1770   2  1                 READ FILE(STRINGS) INTO(INPUT);            ZEF01770
 1780   2  1                 PUT EDIT (INPUT) (A) SKIP;                 ZEF01780
 1790   2  1                 PUT EDIT (' ') (A) SKIP;                   ZEF01790
 1800   2  1                 M=1;                                       ZEF01800
 1810   2  1                 N=1;                                       ZEF01810
 1820   2  1  AGAIN:                                                    ZEF01820
 1830   2  2                 DO J = M TO 80 UNTIL(A_INPUT(J) ¬= ' '); END; ZEF01830
 1840   2  1                 IF J >= 80                                 ZEF01840
                                THEN DO;                                ZEF01850
 1860   2  2                           A_INPUT(N) = '$';                ZEF01860
 1870   2  2                           N = N+1;                         ZEF01870
 1880   2  2                           SUBSTR(INPUT,N,80-N) = ' ';      ZEF01880
 1890   2  2                           GOTO GETC;                       ZEF01890
 1900   2  2                        END;                                ZEF01900
 1910   2  1                 DO K = J BY 1 UNTIL(A_INPUT(K) = ' '); END; ZEF01910
 1920   2  1                 A_INPUT(N) = INT(SUBSTR(INPUT,J,K-J));     ZEF01920
 1930   2  1                 N = N+1;                                   ZEF01930
 1940   2  1                 M = K;                                     ZEF01940
 1950   2  1                 GOTO AGAIN;                                ZEF01950
 1960   2  1              END;                                          ZEF01960

 1970   2  0        SYM = A_INPUT(I);                                   ZEF01970
 1980   2  0        A_INPUT(I) = ' ';                                   ZEF01980
 1990   2  0        N_SYM = PTR_SYM(SYM);                               ZEF01990
 2000   2  0    PUT EDIT                                                ZEF02000
                 (SUBSTR(SYMBOLS(N_SYM).FULL,1,SYMBOLS(N_SYM).L_FULL),' => ',SYM) ZEF02010
                          (3 A) SKIP;                                   ZEF02020
 2030   2  0        IF SYM = '$'                                        ZEF02030
                       THEN NEW_STR = '1'B;                             ZEF02040

 2050   2  0  END; /* GETSYM */                                         ZEF02050
```

```
NUMBER LEV NT                                                                                R

  2060   1  0  INT: PROC(GSYM) RETURNS(CHAR(1));                                ZEF02060

               /*-------------------------------------------------------        ZEF02070
                 GIVEN FULL GRAMMAR SYMBOL THIS PROCEDURE RETURNS INTERNAL      ZEF02080
                 GRAMMAR SYMBOL.                                                ZEF02090
               -------------------------------------------------------*/        ZEF02100

  2110   2  0  DCL GSYM CHAR(30);                                               ZEF02110
  2120   2  0  DCL SYM CHAR(1);                                                 ZEF02120
  2130   2  0  DCL (I,J) FIXED BIN(15);                                         ZEF02130

  2140   2  0        DO I = 1 TO NO_SYM;                                        ZEF02140
  2150   2  1          J = SYMBOLS(I).L_FULL;                                   ZEF02150
  2160   2  1          IF SUBSTR(SYMBOLS(I).FULL,1,J) = SUBSTR(GSYM,1,J)        ZEF02160
                          THEN DO;                                              ZEF02170
  2180   2  2                    SYM = SYMBOLS(I).SYM;                          ZEF02180
  2190   2  2                    RETURN (SYM);                                  ZEF02190
  2200   2  2                 END;                                              ZEF02200
  2210   2  1        END;                                                       ZEF02210
  2220   2  0        PUT EDIT ('UNDEFINED GRAMMAR SYMBOL: ',GSYM)               ZEF02220
                               (2 A) SKIP;                                      ZEF02230
  2240   2  0        SYM = 'X';                                                 ZEF02240
  2250   2  0        RETURN (SYM);                                              ZEF02250

  2260   2  0  END;  /* INT */                                                  ZEF02260

  2270   1  0  END;  /* ZEFSDL */                                               ZEF02270
```