SRS, A SYNTAX RETRIEVAL SYSTEM

Duane A. Adams

CGTM 12*
January 1967

*This report also submitted as a CS239 report, Fall Quarter 1966.

SRS, A Syntax Retrieval System

I. INTRODUCTION

As computer systems are developed which allow a closer man-machine interaction, we may expect the user to take greater advantage of the computer's information storing and retrieving capabilities in a way not possible heretofore. In particular, the user who has access to a computer from a terminal equipped with a cathode ray tube (CRT) may enjoy the greatest benefits from such a system. There are certain types of information which might be useful to a large class of users. Let us consider having available the syntax of the various programming languages which are implemented on a particular system. A programmer, for example, may wish to quickly find out something about the language he is using without searching through a manual for it. Or the person learning a language may wish to see the relationship between different parts of the syntax. Thus, a program has been written to see whether or not we can design a useful system for retrieving the syntax of a programming language.

The Syntax Retrieval System, SRS, is a program which allows the user to select and display on a CRT the syntax of any programming language which may be included in the system. Any language describable in Backus Normal Form (BNF) may be included. SRS consists of two major parts. The first is a program which takes a suitably coded description of the language and constructs a data base. The second is a monitor which responds to user requests at the CRT. The program is written in Fortran IV and has been run on an IBM 360/50 equipped with a 2250 scope.

II. DEFINITIONS

Most of the terminology used in this paper is quite standard. The following definitions are included to avoid any possible misinterpretation.

<u>Phrase class name</u>. A string of letters enclosed in < > and used to denote a single unique element of the programming language.

Symbol. Any element of the programming language; included are phrase class names, delimiters, and reserved words.

Production. U ::= x, where U is a phrase class name and x is a finite string of symbols. U is called the left part of the production and x is called the right part.

Production set. The set of all productions having the same left part.

Descriptor. A 3-character sequence, including blanks, used to identify a symbol of the language.

Note that in BNF a production set is indicated by only writing the left part once, and separating each of the right parts by a | .

III. DATA REPRESENTATION

For each language which is placed in SRS it is necessary to encode the syntax in the following form so that it may be recognized by the system. A 3-letter descriptor, including blanks, is assigned to each symbol of the language. This descriptor for a delimiter may be just the delimiter itself, followed by blanks, while for the phrase class names and reserved words is may be desirable to choose a descriptor having some mnemonic significance. In ALGOL, for example, we have let) be the descriptor for a right parenthesis and SAE be the descriptor for Simple Arithmetic Expression.

Each production set is entered into the system as a unit. The following example illustrates how a production set written in Backus Normal Form is encoded for input to SRS.

BNF: <simple arithmetic expression> ::= <term>|<adding operator> <term>|
<simple arithmetic expression>
<adding operator> <term>

SRS: SAE <SIMPLE ARITHMETIC EXPRESSION>. TRM,ADO TRM,SAE ADO TRM.

The order in which the production sets are entered into the system is immaterial; however, the following features should be noted:

1. Each descriptor is defined by its appearance as the leading element in a production set.
2. Each phrase class name appears only once, and is thereafter referred to by its descriptor.
3. A period (.) denotes the end of a phrase class name, and also the end of a production set.
4. A comma(,) is used in place of the | to separate right parts.
5. A space denotes juxtaposition of symbols within a right part.
6. As it is presently coded, each right part descriptor must begin on a word boundary when read in on an A4 format, and there must be at least one space between the period following the left part definition and the first right part descriptor of the production set.

Throughout this report we will frequently be referring to the use of descriptors. The following reasons are offered as to why descriptors were used instead of including the entire phrase class name each time it was encountered.

1. Each descriptor is unique and can be used as a one-word table entry on which to search for input requests.
2. Descriptors take up considerably less storage than do the corresponding phrase class names.
3. Descriptors can be used as input requests when working at the CRT, thus making the retrieval easier and faster. More will be said about this later.

Internal Representation

The internal representation for the data is intended to minimize the amount of core storage required and to make the retrieval as fast as possible. To accomplish the first objective, each phrase class name appears only once in core in its full Extended Binary-Coded-Decimal Interchange Code (EBCDIC), and is thereafter referred to by its descriptor. To accomplish the second, there is a sorted table of descriptors containing pointers to the various productions. Also, associated with each appearance

of a descriptor in the array MAA is a pointer to the definition of that descriptor; and finally, the symbols are cross referenced in such a way that all references to any given symbol may be quickly located.

As the data is read in, two arrays are created. The array MAA contains a slightly edited version of the card image for each production set. Figure 1 illustrates the format in which the data is stored. The array TABLE contains the descriptor for each symbol and the location in the array MAA where the symbol is defined. The descriptors are sorted alphabetically so that a standard binary search may be used to locate any given one. Figure 2 shows the table that was produced for ALGOL 60. Note that each symbol of the language is defined. The order in which the syntax is read in is immaterial.

Once the data has been read in, the cross referencing begins. For each right part descriptor, beginning with the first production set in MAA, we must locate the address of the data item which contains the full EBCDIC description. The address is placed in the word immediately following the right part descriptors.

Besides being able to retrieve the right parts of a production set, one would also like to be able to answer the question, "In which productions does a particular symbol appear?" Thus, for each symbol in the language we have a list containing pointers to each production in which that symbol appears in the right part. It takes two words of storage to make each cross reference; one word contains a pointer to the production in question, while the second word points to the next cross reference on the list. The cross references are built onto the array MAA. The first element of the list for each symbol appears in a word immediately following the description of that symbol. See arrow 5 of Fig. 1. Each list is ended with the terminator ****. Figure 3 shows the cross linking scheme.

IV. CRT MONITOR

When this program was originally written (and at the time this report was written), the IBM 2250 scope at SLAC was not equipped with either a

character generator or a buffer. This was particularly inconvenient for this project since we wished to display only text on the scope. As a result it was necessary to make each letter from a sequence of vectors. On the average it takes about six words of core to represent a letter in vector form. The lack of a buffer on the 2250 meant that a buffer had to be simulated in core on the 360. A display is generated on the scope by calling a subroutine CRT, which sequentially takes words from the buffer area and transmits them to the 2250. To keep a constant display on the scope, the display must be regenerated about 30 to 40 times per second With a large buffer (i.e., 2000 words) this rate cannot be maintained and the display on the screen flickers. Despite the inconvenience caused by the hardware, a suitable monitor has been written to allow an evaluation of the system. Various additions to and changes in the system will be made, but not until after the character generator and buffer have been installed on the scope. In the following paragraphs the monitor is described from the standpoint of what the user must request and what he receives while working at the CRT. A flow chart of the program may be found in the appendix.

The monitor has been designed so that it will be very simple to understand and to use. From the keyboard the user can ask for a particular production set, ask for all productions containing a given symbol in the right part, or delete everything which appears on the scope. Besides deleting the entire display, the user may selectively delete a production or a production set by pointing the light pen at the line he wishes to delete.

As has already been mentioned, each symbol in the language is assigned a descriptor. These descriptors are used to identify the user's request. For example, to find out what a TERM is the user types

R TRM.

The R indicates that he wants the production set defined by TERM, and the period (.) terminates his request. To find all the productions in which TERM appears the user types

L TRM.

In order to retrieve a particular piece of information, it is necessary to know the descriptor. Many of them have enough mnemonic significance so that after only a short period of use one knows what to request. However, there are three different ways in which the user may obtain the descriptor. The first is to consult an alphabetical listing of the symbols of the language along with the corresponding descriptors. Figure 4 shows such a listing for ALGOL 60. The second method is to request help from the system. By typing an H. a set of instructions appears on the scope describing how the system works. By typing H 1. the first part of the dictionary is obtained, which gives in alphabetical order the correspondence between symbols and descriptors, H 2. gives the second part, and so on. The dictionary may be called at any time without destroying the present display. It is saved in a buffer and automatically restored at the next command. The third method of obtaining descriptors is closely related to the way in which the data is displayed. The following example indicates how the request R TRM. is displayed:

```
TRM <TERM> ::=
    FCT <FACTOR>
    TRM <TERM> MOP <MULTIPLYING OPERATOR> FCT <FACTOR>
```

For the above production set each right part begins on a new line and is indented to distinguish right parts from the left part. Also, in smaller letters, the descriptor for each phrase class name preceeds the name so that it is easy to "chain" requests without reference to a dictionary. In the event that a wrong descriptor is entered, a message to that effect will appear on the scope. Figure 6 shows how the display actually appears to the user.

The display is generated from the top of the scope, with each new request appearing sequentially below the previous request. At the bottom of the screen is a line which is reserved for displaying the request as

it is being typed in. A cursor points to the location where the next letter will be entered. Errors can be corrected by backspacing the cursor and retyping the correct letters. At present this is done by typing a <, but when the buffer is installed on the 2250 the regular keyboard backspacer may be used. All keyboard requests are terminated with a period.

The light pen may be used to delete a particular right part of a production set by pointing the light pen at that line on the scope, or it may be used to delete an entire production set by pointing at the left part of the production set. In this manner the user may display on the scope only those productions he is currently using. When a line is removed from the scope, the remaining text is moved up to fill in the gap. A summary of the commands available through the monitor is given in Figure 5.

V. SUMMARY AND CONCLUSIONS

The Syntax Retrieval System allows one to store and selectively retrieve the syntax of any programming language expressable in Backus Normal Form. The program is written in FORTRAN IV and is designed to be used with a CRT equipped with a standard keyboard and light pen. The system, as it appears on the scope to the user, is completely self-explanatory and is designed to be easy to understand and to use. From the limited amount of use which the SRS has received so far, it appears that the system does fulfill its objectives in that the user can easily and quickly locate detailed information about the syntax of a programming language.

It has already been mentioned that a certain amount of reprogramming will be necessary to take advantage of a buffered CRT with a character generator. This will include putting in the dictionary as described in this report. I intend to make these changes as soon as the buffer and character generator are installed on the 2250.

Besides the above changes which will be made to SRS, there are several other ways in which the system could be refined and expanded. The present version of SRS requires about 3000 words of storage for the ALGOL 60 syntax and the cross referencing. This storage could be greatly reduced by replacing each right part descriptor in the array MAA by a pointer to its defining location rather than using a word of storage for each of these items. This can be done since once the cross referencing has been performed, the descriptors are redundant. In Fig. 1 this means combining the locations given by 2 and 3 with their adjacent locations given by 6.

At present only ALGOL 60 has been put in SRS. Various other languages could be added to the system, for example PL/I. Once a language has been entered into the system, it is completely described by the arrays MAA and TABLE. Thus, by saving these arrays on auxiliary storage for each language in SRS, it would be possible for the user to specify from the CRT console which language he wished to be able to display.

The language designer who uses SRS would probably want the capability of modifying his language from the CRT console, although care must be taken to insure that a language available for general use cannot be so modified. Beyond that comes the problem of trying to include semantics in a retrieval system. Finally, it seems that SRS may be only one of a number of such programs which might be useful to many users. It should be possible to make these programs readily available through some form of general information system. To design and write such a system would be a major project.

<TERM> :: = <FACTOR>|<TERM> <MULTIPLYING OPERATOR> <FACTOR>

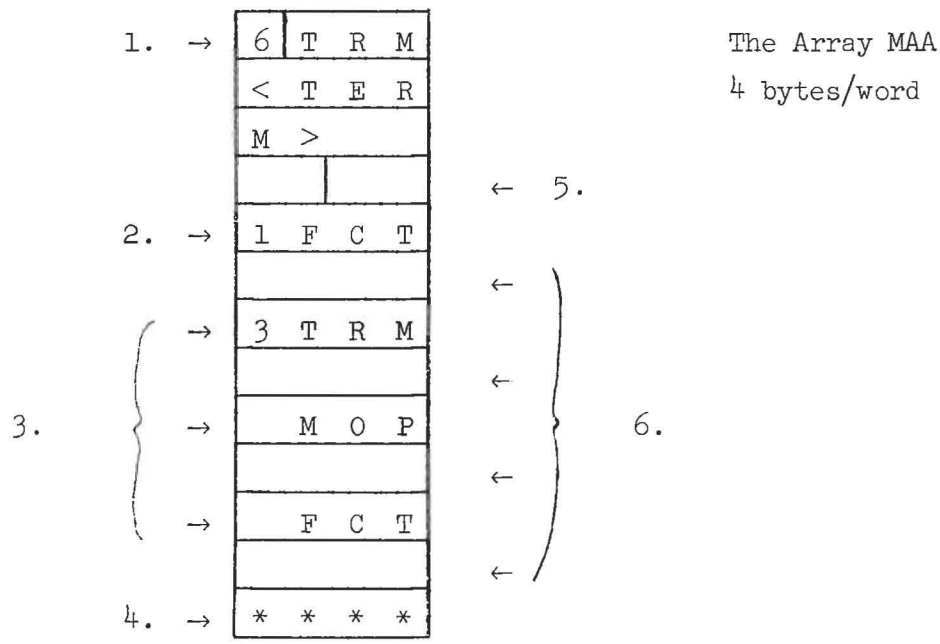

Fig. 1 Illustration of how production set of TERM stored in the array MAA.

1. The 6 indicates that there are 6 bytes defining the phrase class name having the descriptor TRM.
2. The first right part of TERM has only 1 symbol and is denoted by the descriptor FCT.
3. The second right part of TERM consists of 3 symbols, given in order by the descriptors TRM, MOP and FCT.
4. The **** indicates the end of a production set.
5. The first element of a list of cross references for TERM.
6. Pointers to the locations in MAA where each of the right part descriptors is defined.

	2100	<	1736	<=	1796	(	1742
+	1724	*	1772	**	1784	)	1748
;	1820	¬=	1814	-	1730	/	1766
,	2014	>	1754	>=	1802	:	1790
:=	1778	@	2020	'	1808	=	1760
A	1694	ADC	1426	ADD	542	AE	648
AID	397	ALT	1412	AND	1844	AOP	82
APA	445	APL	495	APP	512	ARY	1965
ASM	1391	AST	1060	B	1700	BE	790
BFT	736	BGN	1926	BKT	170	BLH	964
BLK	1193	BOL	1945	BPL	1375	BPM	700
BPR	1363	BSC	722	BST	903	BTM	751
C	1706	CDE	1280	COM	1919	CPS	1007
CST	1110	CTL	947	D	1712	DCL	190
DE	865	DEC	1236	DFR	269	DIG	27
DIV	1826	DLM	52	DNO	292	DO	1893
DST	1089	E	1718	ELS	1881	END	1933
EPT	281	EQV	1832	EXP	338	FAL	2007
FCL	1177	FCT	587	FDG	529	FL	1163
FLE	1137	FOR	1887	FPA	1475	FPL	1485
FPP	1502	FST	1206	GO	1862	GST	1077
ID	223	IDL	1519	IF	1869	IFC	636
IMP	1838	INT	1952	IPC	766	IST	1099
ITG	253	LAB	2086	LBD	1345	LBL	808
LET	1265	LOP	103	LOT	1319	LPL	1047
LPT	1033	LSG	482	LVA	16	MOP	554
MTY	2093	NO	323	NOT	1856	OPR	68
OR	1850	OSG	1643	OWN	1939	PDC	1620
PDL	463	PGM	1023	PID	434	PMY	569
PRB	1609	PRH	1591	PRO	1979	PSG	1662
PST	1223	REL	688	RL	1959	ROP	667
RW	2107	SAE	613	SBL	1	SBO	781
SBV	407	SDC	1459	SDE	843	SDG	827
SEP	142	SEX	371	SID	817	SLT	382
SMV	361	SOP	121	SPC	210	SPF	1549
SPT	1575	ST	935	STG	1987	STP	1899
STR	1683	SWL	1444	SWT	1972	TDC	1333
THN	1875	TLT	1251	TRM	600	TRU	2001
TYP	1308	UBD	1354	UBL	993	UBS	885
UCP	980	UNI	239	UNO	307	UST	919
UTL	1905	V	424	VAL	1994	VID	350
VPT	1535	WHL	1912	0	2026	1	2032
2	2038	3	2044	4	2050	5	2056
6	2062	7	2068	8	2074	9	2080

FIGURE 2

DROP BACKGROUND ON THIS ONE

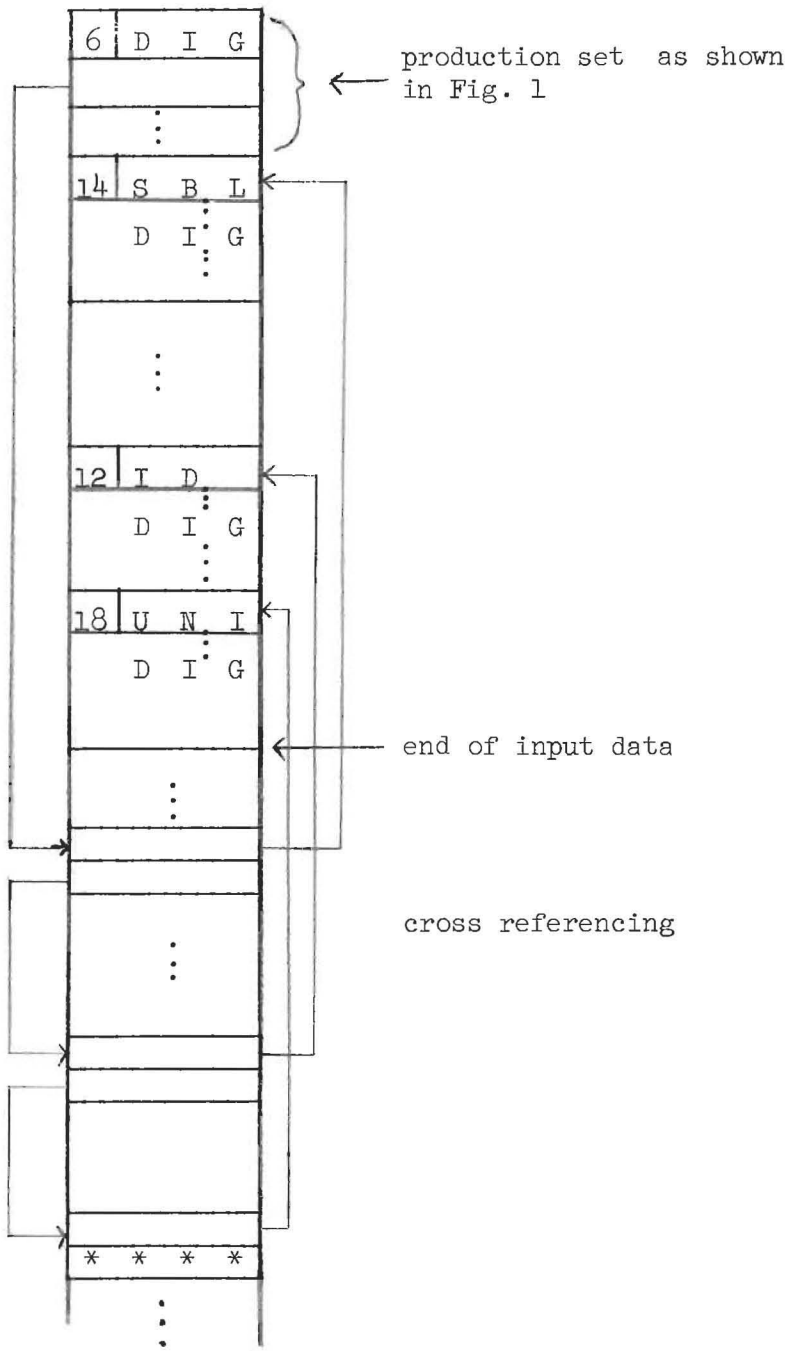

Fig. 3 An illustration of the cross referencing between symbols in the array MAA. This shows that the symbol DIGIT (DIG) appears in the production sets defined by BASIC SYMBOL (SBL), IDENTIFIER (ID), and UNSIGNED INTEGER (UNI).

Underlined words denote reserved words.
Basic symbols appear at the end.

ACTUAL PARAMETER	APA
ACTUAL PARAMETER LIST	APL
ACTUAL PARAMETER PART	APP
ADDING OPERATOR	ADO
AND	AND
ARITHMETIC EXPRESSION	AE
ARITHMETIC OPERATOR	AOP
ARRAY	ARY
ARRAY DECLARATION	ADC
ARRAY IDENTIFIER	AID
ARRAY LIST	ALT
ARRAY SEGMENT	ASM
ASSIGNMENT STATEMENT	AST
BASIC STATEMENT	BST
BASIC SYMBOL	SBL
BEGIN	BGN
BLOCK	BLK
BLOCK HEAD	BLH
BOOLEAN	BOL
BOOLEAN EXPRESSION	BE
BOOLEAN FACTOR	BFT
BOOLEAN PRIMARY	BPM
BOOLEAN SECONDARY	BSC
BOOLEAN TERM	BTM
BOUND PAIR	BPR
BOUND PAIR LIST	BPL
BRACKET	BKT
CODE	CDE
COMMENT	COM
COMPOUND STATEMENT	CPS
COMPOUND TAIL	CTL
CONDITIONAL STATEMENT	CST
DECIMAL FRACTION	DFR
DECIMAL NUMBER	DNO
DECLARATION	DEC
DECLARATOR	DCL
DELIMITER	DLM
DESIGNATIONAL EXPRESSION	DE
DIGIT	DIG
DIV	DIV
DO	DO
DUMMY STATEMENT	DST
ELSE	ELS
EMPTY	MTY
END	END
EQV	EQV
EXPONENT PART	EPT
EXPRESSION	EXP
FACTOR	FCT
FALSE	FAL
FOR	FOR
FOR CLAUSE	FCL
FOR LIST	FL
FOR LIST ELEMENT	FLE
FOR STATEMENT	FST
FORMAL PARAMETER	FPA
FORMAL PARAMETER LIST	FPL
FORMAL PARAMETER PART	FPP
FUNCTION DESIGNATOR	FDG
GO TO	GO
GO TO STATEMENT	GST
IDENTIFIER	ID
IDENTIFIER LIST	IDL
IF	IF
IF CLAUSE	IFC
IF STATEMENT	IST
IMP	IMP
IMPLICATION	IPC
INTEGER	INT
INTEGER	ITG
LABEL	LAB
LABEL	LBL
LEFT PART	LPT
LEFT PART LIST	LPL
LETTER	LET
LETTER STRING	LSG
LOCAL OR OWN TYPE	LOT
LOGICAL OPERATOR	LOP
LOGICAL VALUE	LVA
LOWER BOUND	LBC
MULTIPLYING OPERATOR	MOP
NOT	NOT
NUMBER	NO
OPEN STRING	OSG
OPERATOR	OPR
OR	OR
OWN	OWN
PARAMETER DELIMITER	PDL
PRIMARY	PMY
PROCEDURE	PRO
PROCEDURE BODY	PRB
PROCEDURE DECLARATION	PDC
PROCEDURE HEADING	PRH
PROCEDURE IDENTIFIER	PID
PROCEDURE STATEMENT	PST
PROGRAM	PGM
PROPER STRING	PSG

Fig. 4 A Dictionary of symbols and descriptors for ALGOL 60. Note that certain ALGOL 60 characters must be represented differently because of the hardware.

REAL	RL
RELATION	REL
RELATIONAL OPERATOR	ROP
RESERVED WORD	RW
SEPARATOR	SEP
SEQUENTIAL OPERATOR	SOP
SIMPLE ARITHMETIC EXPRESSION	SAE
SIMPLE BOOLEAN	SBO
SIMPLE DESIGNATIONAL EXPRESSION	SDE
SIMPLE VARIABLE	SMV
SPECIFICATION PART	SPT
SPECIFICATOR	SPC
SPECIFIER	SPF
STATEMENT	ST
STEP	STP
STRING	STR
STRING	STG
SUBSCRIPT EXPRESSION	SEX
SUBSCRIPT LIST	SLT
SUBSCRIPTED VARIABLE	SBV
SWITCH	SWT
SWITCH DECLARATION	SDC
SWITCH DESIGNATOR	SDG
SWITCH IDENTIFIER	SID
SWITCH LIST	SWL
TERM	TRM
THEN	THN
TRUE	TRU
TYPE	TYP
TYPE DECLARATION	TDC
TYPE LIST	TLT
UNCONDITIONAL STATEMENT	UST
UNLABELED BASIS STATEMENT	UBS
UNLABELED BLOCK	UBL
UNLABELED COMPOUND	UCP
UNSIGNED INTEGER	UIN
UNSIGNED NUMBER	UNO
UNTIL	UTL
UPPER BOUND	UBD
VALUE	VAL
VALUE PART	VPT
VARIABLE	V
VARIABLE IDENTIFIER	VID
WHILE	WHL

BASIC SYMBOLS	
-	-
+	+
(	(
)	)
[	(
]	)
<	<
≤	<=
=	=
>	>
≥	>=
×	*
/	/
*	**
:=	:=
≠	¬=
@	@
;	;
,	,
"	"
:	:
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

(All upper and lower case letters are also basic symbols)

Fig. 4 (page 2)

To find all productions defined by descriptor XXX type

R XXX.

To find all productions in which XXX appears in the right part type

L XXX.

To delete the entire display type

D.

To return to this page of instructions type

H.

To display page 1 of the dictionary to locate the descriptors type

H 1.

For the remaining pages of the dictionary type

H 2. H 3. etc.

To delete a production use the light pen.

Fig. 5 INSTRUCTIONS FOR USING SRS