ALGEBRAIC DIFFERENTIATION USING A TOP-DOWN SYNTAX ORIENTED TRANSLATOR

A THESIS

PRESENTED TO

THE FACULTY OF GRADUATE STUDIES AND RESEARCH

THE UNIVERSITY OF MANITOBA

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS OF THE DEGREE

MASTER OF SCIENCE

IN THE DEPARTMENT OF COMPUTER SCIENCE

by

Ching Hoi Tse

February 1971

ABSTRACT

This work employs a top-down syntax oriented translator to obtain a derivative of an algebraic expression. Tables of syntax and semantics to achieve this purpose are given. Simplification of the resulting expression is performed eliminating both redundant parentheses and terms consisting of 0 or 1. With a slightly extended version of the translator, partial derivatives can also be obtained, and any resulting expressions evaluated at assigned points.

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to Dr. P.R. King, my thesis supervisor, for his guidance and supervision in the preparation of this thesis.

I would also wish to thank Professor J. Wells and Professor A. Wexler for the reading of this thesis.

Lastly, I would like to thank Miss N. Cheng for typing this thesis.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION AND REVIEW OF LITERATURE

1.1. Introduction.

This work employed a syntax-directed translator [30] to perform algebraic differentiation on a digital computer. Particular efforts are made on the provision of the necessary syntax specifications so that the input algebraic expressions would be recognized. With the necessary semantics the corresponding target string would be produced. This can be in any desired form depending on the interpretation rules (semantics) ; in this case it would be the result of albegraic differentiation.

The translator employed uses a top-down technique [9, 16, 30] as contrast to the bottom-up method used in other syntax-directed translator [26, 27, 31]. Some basic features of the translator are discussed and the precise rules of syntax and semantics are discussed next. This translator was originally written with a purpose of performing works on compiling but it was found that other applications can be done as well [20]. This work is one of the example of these applications.

The advantage of this method is that any changes and modifications necessary in the translation can be made by just changing the syntactic and semantic

specifications without regard to the logic of the translator. This would not be the case if a program were written specifically for analytic differentiation [23, 29, 35].

In contrast to Schorr's [39] work, three syntax and semantic tables are provided for translation. The first set deals with algebraic differentiation using the basic rules of differentiation whereas the target string produced is processed alternately by the other two sets of syntax and semantic tables to obtain a simplified output. With a separately written input routine and 'value' routine for the translator the following can be obtained:

1. the derivatives of higher order;
2. the partial derivatives;
3. the derivatives of implicit functions; and
4. the evaluation of the resulting derivative.

1.2. Formal grammars.

The importance of formal grammars was first apparent when in the report on ALGOL [33, 34] the language is defined by a set of syntactic rules. The studies were further spurred when the development of the syntax-directed and syntax-controlled processors required a formal definition of the language to be processed. The first important original papers on formal studies of

grammars were done by Chomsky [10, 11] and Bar-Hillel et al [2]. Since then formal studies on grammars have become one of the major fields in linguistic studies both in programming and natural languages. It is found that out of the formal grammars the context-free phrase-structure grammar is of particular interest. The definition of this grammar is perhaps best expressed with the notation used by Ginsburg [21]. Suppose $\mathcal{A}$ denotes the set of terminal symbols (e.g. $\underline{T}_1$, $\underline{T}_2$, T_3, etc.), η, the set of nonterminal symbols (denoted by Latin capitals e.g. $\underline{U}$, $\underline{V}$, and $\underline{Z}$) and the vocabulary $\mathcal{V}$ is defined as the union of $\mathcal{A}$ and η. The symbols $\underline{S}$, $\underline{S}_1$, $\underline{S}_2$ etc. are used to denote members of $\mathcal{V}$, while strings of symbols (including the empty string $\wedge$) are denoted by lower case Latin letters $\underline{u}$, $\underline{v}$, $\underline{w}$, ... The context-free phrase-structure grammar $\mathcal{G}$ then can be defined as a finite set of productions of the form $\underline{U}_i \longrightarrow \underline{u}_i$ with the following properties :

1. each $\underline{u}_i$ is a nonempty string whose symbols are in $\mathcal{V}$
2. each $\underline{U}_i$ is a nonterminal symbol i.e. $\underline{U}_i \in \eta$
3. there is a unique $\underline{U}_i$ which occurs in no $\underline{u}_i$.

A language that can be specified in terms of this grammar is called a context-free phrase-structure language. As an example the following is a context-free phrase-structure grammar

S ⟶ E

E ⟶ T + E

E ⟶ T

T ⟶ P * T

T ⟶ P

P ⟶ I

P ⟶ (E)

where S E T P are nonterminal symbols and + * ( ) I are terminal symbols. This grammar specifies restricted arithmetic expressions like I+I*I and (I+I+I)*(I+I) etc. However, in order to express the grammar in a more compact form Backus Normal Form (BNF) [1] is used. In this notation the symbol '⟶' is replaced by '::=' while production rules with the same left side are combined using the symbol '|' with the meaning "or". Hence the above grammar can be rewritten as

S ::= E

E ::= T + E | T

T ::= P * T | P

P ::= I | (E)

The BNF notation is in fact proved to be equivalent to the formal notation by Ginsburg [21] and is the notation employed for the syntax specification in this work. The particulars and details of these syntactic rules are discussed in Chapter II.

1.3. Works on algebraic differentiation and related fields.

There is much literature on the topic of formula manipulation using a digital computer [36, 37]. The most significant systems are perhaps the FORMAC [38] and ALPAK [6, 7, 25]. These systems are general systems that handle algebraic manipulations like multiplication, division, addition and subtraction of polynomials, differentiation and certain simplifications of algebraic expressions. Programs also have been written soly on algebraic differentiation. The earliest work on this was written independently by Nolan [35] and Kahramanian [29] in 1953. In both cases the mathematical expression to be differentiated has to be completely parenthesized and encoded into a three-address code. The derivative was output in tabular form and the resulting analytic expression for the derivative then developed by hand. A more recent work was done by Hanson et al [23] in which the algebraic expression can directly served as input without any modifications, but is transformed automatically into a table of triples by Ershov's algorithm [14]. The required derivative of each triple can then be obtained by applying the basic rules of differentiation on these triples. The parenthesis-free tabular form of the derivative is then transformed into a parenthesized string or expression. Another approach to symbolic differentiation is that of Weissman [42] who makes use of the list process-

ing language LISP 1.5. The input algebraic expression is first transformed to its prefix form, then differentiated, simplified and converted back to its infix form for output.

All these approaches to algebraic differentiation are written in a specific language for symbol-manipulation. In this work an entirely different method is used. A top-down syntax oriented translator is employed with the proper syntax and semantic tables. Schorr [39] also used this method for algebraic differentiation, but the translator used is a bottom-up type developed by E. T. Irons [26, 27] . There has been much controversy about the relative efficiency of the top-down and bottom-up translators [4, 22] , however, it is not intended here to present an analytic judgement on these two types.

CHAPTER II

METHOD OF ANALYSIS

2.1. Introduction.

The translator employed was first designed and written by King [30] in ALGOL 60. A pure top-down syntax analysis technique is used in this translator to analyze the input string. The input string is first read in together with its assumed phrase (syntactic) type, and is stored in an integer array called "source". It is then analysed according to the given syntax. If the search is successful, that is, if the source string conforms to the syntax, the corresponding target string is produced according to the semantics, and is stored in the integer array "targ" ready for output. The translator thus consists of four main subroutines :

(a) READSYNTAX to read and store the syntax.

(b) READSEMANTICS to read and store one or more sets of semantics.

(c) SEARCH to attempt a syntactic analysis of the source string according to the syntax and to assemble a tabular analysis record if the search is successful.

(d) COMPILE to apply the semantics to the source string using the analysis record, in order to produce the target string.

The program also contains a number of subsidiary routines for input output etc.; a complete listing of the program appear in Appendix B.

2.2. Notation for Syntax Specification.

The notation employed is basically Backus Normal Form (BNF). Backus Normal Form has become extremely well known since its first appearance in 1959 [1]; it is perhaps best to explain its principal features with examples.

Example 1

<digit> ::= 0|1|2|3|4|5|6|7|8|9

Here digit is the name of the phrase to be defined and is enclosed by the symbols <and > to distinguish it from the terminal symbols 0, 1, 2, ... 9; the symbol ::= is a separation symbol separating the right and left side of the definition and it can be interpreted as "is defined as"; the symbol | separates the alternatives and has the meaning 'or'.

The above definition means that phrase type <digit> is defined as the symbol 0 or 1 or 2 or 3 ...., or 9.

A phrase type may be defined in terms of other phrase types.

Example 2

<integer> ::= <digit> | <integer> <digit>

Note the recursive nature of the above definition; an integer is defined as one digit or a string of digits of arbitrary length.

A phrase type can be defined in terms of both terminal symbols and phrase types.

Example 3

<u/s integer> ::= <integer> | + <integer> | - <integer>

2.2.1. Modifications to standard BNF notation.

Because of the search mechanism employed in our translator the following two restrictions must be imposed on the phrase definitions.

1. Infinite recursion

A definition of the form

<A> ::= <A> <B>

or <A> ::= <B>

<B> ::= <C>

<C> ::= <A>

will cause the search routine to search in an infinite loop. In order to avoid this the following restrictions are imposed on the grammar :

(i) No definition should start with the phrase type it is defining.

(ii) In a set of definitions (S_1, S_2, S_3, ... S_n), no cycle or loop should exist i.e. none of the alternatives of S_1 may begin with S_2, S_2 with S_3, ..., S_n with S_1.

Thus in view of the above conditions

⟨integer⟩ ::= ⟨integer⟩ ⟨digit⟩ | ⟨digit⟩

is not a permissible definition. It may be rewritten as

⟨integer⟩ ::= ⟨digit⟩ ⟨integer⟩|⟨digit⟩

2. Non-maximum search.

Using the definition

⟨integer⟩ ::= ⟨digit⟩ | ⟨digit⟩ ⟨integer⟩

due to the method employed in the search routine, no integer of more than one digit will be recognized successfully. For example given the integer 456, the search routine will recognize the digit 4 and exit because the first alternative for phrase type ⟨integer⟩ is satisfied. Therefore it is necessary to rearrange all the alternatives so that they are in the order of decreasing length of phrase which may be found. A satisfactory definition of integer would be

⟨integer⟩ ::= ⟨digit⟩ ⟨integer⟩|⟨digit⟩

The above restrictions are in fact quite common for top-down syntax analyzers.

The following fairly trivial additional modifications are made for this particular translator :

1. The metasymbol ::= is replaced by the symbol =.

2. Each phrase type is assigned a unique positive integer.

3. The phrase type name referred on the right side of the metasymbol '=' can be replaced by its corresponding phrase type number.

4. Each phrase type definition is terminated by the metasymbol ';'.

With the above modifications the phrase definitions

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<integer> ::= <digit> <integer> | <digit>

are rewritten as

11 <digit> = 0|1|2|3|4 | 5|6|7|8|9 ;

12 <integer>= <11> <12> | <11> ;

5. In order to increase the efficiency of searching the metasymbols '(' and ')' are introduced. These metasymbols may enclose alternatives that have common subphrases. <E> is used for the null string. Hence phrase type 12 above can be rewritten as

12 <integer>= <11> (<12> | <E>);

6. Some of the most common phrase definitions such as <digit>, <integer>, <number>, <letter>, <identifier> and <string> that are frequently referred to are built into the translator. The phrase type digit

may be referred to as <D> , letter as <L> , identifier as <I> , integer as <N> and character string as <S> . Hence if we want to define a phrase type <decimal number> , with the above built-in phrase definitions we can define it as

31 <decimal number> = <N> . <N> | <N> . | <N> | . <N> ;

or 31 <decimal number> = <N> (. < N> | . | <E>) | . <N> ;

In fact automatically assigned phrase type numbers are entered whenever the above built-in phrase type (bips) are used; 1 is assigned to phrase type letter <L>, 2 to digit <D> , 4 to identifier <I> , 6 to integer <N> , and 8 to character string <S> . Actually, integers from 1 to 10 are reserved for bips.

7. The symbols <> | ; ( ) and = are reserved as metasymbols in phrase type definitions and therefore cannot be used as terminal characters. They are enclosed by cornered brackets if required as terminals. Thus <<> represents the terminal character < , <> > the terminal character > , <|> the terminal character | and so on.

Example of a complete syntax table.

11 <primary> = <12> | <13> | <14> | <(> <16> <) > ;

12 <indept. variable> = X;

13 <dependent variable> = Y;

14 <constant> = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9;

15 <term> = <11> (* <15> | <E>) ;

16 <expression> = <15> (+ <16> | <E>) ;

Some valid phrases of type 16 would be

4 + X * Y

2 * X * (Y + 5)

X + Y

4 + 5 * X

X

0

(3 + 9 + X)

2.3. Entering the syntax table.

The syntax table must be first read and converted into an internal form that can be made use of by the SEARCH routine in the syntax analysis. This is done by the routine READSYNTAX with two integer arrays DSYN and ILFSYN used for storage purposes. The integer array DSYN is used to store the contents on the right of the metasymbol '=' of each phrase type while another integer array ILFSYN is used to store the start location of each phrase type in DSYN. Hence each element of the vector ILFSYN [i] is, in fact, a pointer to the start of phrase type number i in the array DSYN. Note that the intervening material between the phrase type number and the metasymbol '=' is ignored by the READSYNTAX routine. The significant symbols following '=' are entered successively into the array DSYN with the following assumptions.

1. The metasymbols '=', '<'and'>' are not entered into the array DSYN but used as check points to evoke some appropriate instructions.

2. A unique positive input code number is assigned to each different terminal symbol read.

3. The phrase type number enclosed by the metasymbols '<' and '>' is entered into DSYN as -1 followed by the phrase type number. Hence <11> , would be entered as -1 11 in the array DSYN. Negative numbers are also assigned to some of the metasymbols. The input code and internal representation of each metasymbol is shown in the following list.

metasymbol	Corresponding identifier used in readsyntax		input code	internal representation
	for input code	for internal representation		
=	equals	nil	49	nil
<	ob	nil	44	nil
>	cb	nil	45	nil
(	inin	in	46	-10
)	outin	out	47	-12 -11
;	termin	term	50	-12 -13
|	orin	or	51	-12

4. The built-in phrase types would be entered by the corresponding built-in phrase type number with a preceding negative one. Hence

<L>	would be entered as	-1	1
<D>	as	-1	2
<I>	as	-1	4
<N>	as	-1	6
<S>	as	-1	8

For <E> nothing is entered since it represents a null string.

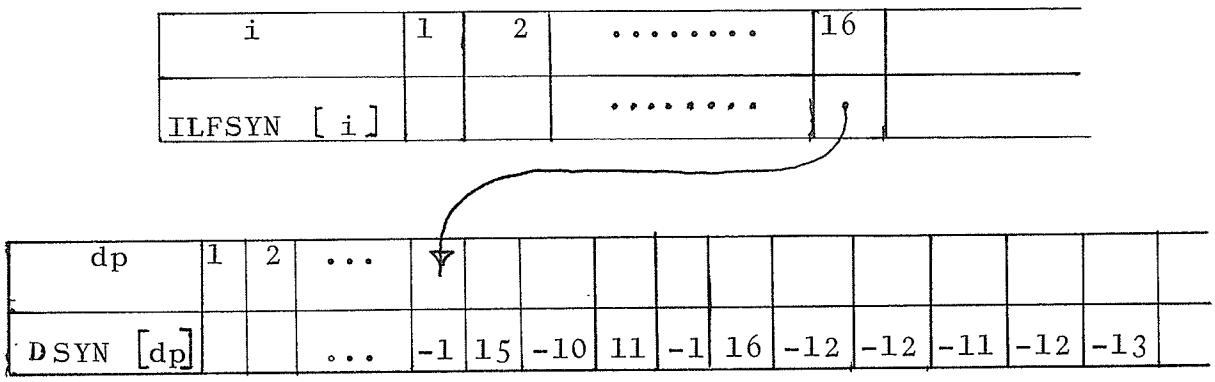
5. When the metasymbol ; is read and the corresponding internal representation is entered, start to read in the next phrase type number as above and perform a similar process. This will be repeated until a suitable terminator for the syntax table is reached such as the occurrence of a phrase of type 0.

Example 5

The locations of ILFSYN and DSYN that contains the phrase type

16 <expression> = <15> (+ <16> | <E>)

is shown schematically below.



where ILFSYN [16] points to the start location in the array DSYN for the phrase type considered. Note that the positive number 11 is the internal code for + and the terminator ; of the phrase type is entered as -12 -13.

2.4. The SEARCH routine and its analysis record.

When the syntax table is stored as above, a source string, given as of a certain phrase type can then be analysed by the SEARCH routine. The first category of the assumed phrase type is searched recursively until a definite answer, either a 'success' or 'failure', is returned. When the result of this search is failure, the next category is tried. The process is repeated until either a successful search is made in one of the alternatives of the assumed phrase type or the terminator of the phrase definition is reached. The details of the development and operation of the procedure SEARCH are fully described in [30]. The

important feature here is the information assembled by the routine in the analysis record. This comprises, for positive integer j = 1, 2, ..., the following integer arrays.

type [j] : the phrase type of entry no. j in the record.

cat [j] : the category recognized in this phrase type.

from [j] : the starting position of the source string analyzed.

to [j] : the ending position of the source string analyzed.

link [i,j] : the position in the record where details of the constituents are stored.

Example 6

The analysis record for the source string 4 + X * Y according to the syntax table on page 12 would be

entry j	type [j]	cat [j]	from [j]	to [j]	link [i,j] i = 1	link [i,j] i = 2
1	16	1	1	5	2	5
2	15	2	1	1	3	0
3	11	3	1	1	4	0
4	14	5	1	1	0	0
5	16	2	3	5	6	0
6	15	1	3	5	7	9
7	11	1	3	3	8	0
8	12	1	1	1	0	0
9	15	2	5	5	10	0
10	11	2	5	5	11	0
11	13	1	5	5	0	0

The first entry of the above analysis record then reads that it is a phrase type 16 of category 1, occupying positions from 1 to 5 of the source string, and the records of its constituent parts are located in entry 2 (for type 15) and 5 (for type 16) respectively. The second entry simply shows that the phrase type 15 that was found is of category 2, occupying only one

position of the source string and further details are located in entry 3. When there is no more detail that can be traced, the value of link $[i,j]$ is automatically zero. The integer i indicates the number of links required and is dependent on the number of phrase types that appears as constituent parts of a definition. Thus the definitions

1 ⟨A⟩ = ⟨2⟩ ⟨3⟩ ⟨3⟩ ;

2 ⟨B⟩ = ⟨3⟩ ⟨2⟩ ;

3 ⟨C⟩ = ⟨2⟩ ;

requires respectively 3, 2 and 1 links. Usually we find that 3 is sufficient.

2.5. Interpretation of the analysis record : Compilation.

2.5.1. Introduction.

In some compiler or translator systems the syntax and semantic analysis is so inextricably entwined that it is impossible to treat them separately and hazardous to change. However this translator has quite separate routines to treat the syntax and semantics. The advantages of this is that either the syntax or semantics or both can be modified without too much difficulty.

There are two possible and useful approaches for providing semantics. The first approach is internal : the semantic rules appear as subroutine in the translator and are written in the language of the translator. The second approach is to design a special purpose metasemantic language in terms of which the semantics can be expressed and which like the syntax form part of the primary data.

2.5.2. Internal semantics.

Here semantic interpretation is performed using a recursive ALGOL procedure (which we called COMPILE for the moment) which has the following procedure heading.

procedure compile (i,j); value i,j; integer i,j;

The procedure interprets the ith link of the phrase detailed by the jth entry of the analysis record. Details of this jth entry can be obtained by the following instructions.

j:=link [i,j] ; i:=type [j] ; k:=cat [j] ;

The variables i and k specify the type and category of the phrase to which the original ith link of the jth entry referred. The procedure has a set of instructions for interpreting every possible type and

category of a phrase, including recursive calls when interpretation of the constituent phrases is required. This procedure also make use of three other additional procedures for the semantic interpretation, namely :

writetext ('string') : to insert the string quoted to the target string.

copyphrase (j) : to copy the phrase given by the j entry of the analysis record to the target string, that is, that portion of source phrase specified by the pointers from [j] and to [j] of the jth entry of the analysis record.

copylink (i,j) : to copy the ith constituent of the phrase defined by the jth entry of the analysis record. This is equivalent to copyphrase (link [i,j]).

With the given syntax and analysis record the algebraic differentiation of the given restricted arithmetic expression 4 + X * Y, can be performed by the procedure.

```
        procedure  diff (i,j)  value i,j; integer i,j;
        begin      integer  k;
                   switch   s := primary, ind variable,
                                 dep variable, constant,
                                 term, expression ;
        j := link [i,j]  ;  i := type [j] ;  k := cat [j] ;
        goto       s [i - 10] ;
primary :     if  k = 4  then
              begin   writetext ('('); diff (1,j); write-
                                                   text (')')
                                        end
        else   diff (1,j) ;
        goto   end ;
ind variable : writetext ('1'); goto end ;
dep variable : copyphrase (j) ; writetext ('''); goto end ;
constant :     writetext ('0'); goto end ;
term     :     if  k = 1  then
              begin  writetext ('('); diff (1,j);
                     writetext ('*'); copylink (2,j);
                                      writetext ('+');
                     copylink (1,j);  writetext ('*');
                     diff (2,j);      writetext (')');
              end
```

```
        else    diff (1,j);
        goto    end ;
expression :    if  k = 1  then
        begin   diff (1,j); writetext ('+'); diff (2,j);
        end

        else    diff (1,j);
end         :   end diff   ;
```

Hence, with information stored in the analysis record and with link [1,0] set to 1, the instruction

```
diff (1,0) ;
```

will cause the output of the derivative of the source string. In our particular example it would be

```
0 + (1 * Y + X * Y')
```

Therefore the output expression is dependent on the procedure which is written for the particular semantics. In this case it is the procedure DIFF .

2.5.3. External semantics.

The semantics described above has a serious drawback in that when the semantics have to be changed or modified for some reason particularly during testing, the entire procedure must be corrected, re-

written and recompiled many times. A metasemantic language has been designed for semantic rules each of which has a form similar to that of the syntax definition. Each rule begins with a phrase type number followed usually by the name of the phrase type enclosed by the cornered brackets then the symbol '=' followed by the semantics. The definition is terminated by the symbol ';'.

The semantics may include five basic instructions described as follows.

(i) W ⟨string⟩ : copy the string into the output. If any of the metasymbols ';', '.', '⟨', '⟩' and '&' are required, they appear in the form '⟨;⟩', '⟨.⟩', '⟨◇', '⟨⟩' and '&' respectively. When space or newline is required, the corresponding instruction would be W⟨⟨S⟩⟩ and W⟨⟨N⟩⟩ respectively.

(ii) P : copy of the phrase of the current entry of the analysis record into the target string.

(iii) L $\langle i \rangle$: copy the ith constituent of the phrase of the current entry of the record into the target string.

(iv) C $\langle i \rangle$: compile, recursively, the ith constituent given by the current entry of the record.

(v) K $\langle i_1, \ldots. i_n \rangle$: 'K' specifies the categories $i_1, \ldots. i_n$ $(n > 0)$. If the current category belongs to the set $\{i_1, \ldots. i_n\}$ then the instructions following K are obeyed until either one of the two terminators . (meaning end) or & (meaning else) is reached; otherwise these instructions are skipped. The terminator & is used only when followed by another K $\langle i'_1, \ldots., i'_{n_1} \rangle$ where the sets $\{i_1, \ldots., i_n\}$, $\{i'_1, \ldots., i'_{n_1}\}$ are disjoint. Otherwise the terminator . should be used instead.

Hence instead of the procedure DIFF (page 22) written for the interpretation of the syntax table of page 12 , the following corresponding external semantics for algebraic differentiation of the restricted arithmetic

expressions (e.g. 4 + X * Y) can be provided instead.

```
11  <primary>              = K <4> W<(> .
                             K <1,2,3,4> C <1> .
                             K <4> W <)> . ;
12  <indept variable>      = W <1> ;
13  <dependent variable>   = PW < ' > ;
14  <constant>             = W < 0 > ;
15  <term>                 = K <1> W <(> C <1>
                             W <*> L <2> W <+>
                             L <1> W <*> C <2>
                             W <)> & K <2> C <1> .;
16  <expression>           = C <1> K <1> W <+>
                             C <2> .;
```

The entire input listing of the syntax and semantics and the analyzed output of 4 + X * Y can be found in Appendix A.

2.6. Entering the external semantics.

The semantic table is read in using the routine READSEMANTICS. Like READSYNTAX two integer arrays ILFSEM and DSEM are assembled and have similar functions to ILFSYN and DSYN in READSYNTAX. After the phrase type number is read, the intervening character

string is again skipped until the metasymbol '=' is reached. The symbols then entered by the READSEMANTICS routine with the following assumptions.

1. The metasymbols '=', '<' '>' and ',' are again not entered into the array DSEM but serves as check points to evoke some appropriate instructions.

2. A positive input code number is assigned to each different symbol entered.

3. Any character string between any one of the special characters (W, K, C, L) and the metasymbol '<' is ignored and treated as comment. Special characters (W, K, C, L) and the metasymbol '<' is ignored and treated as comment.

4. To facilitate the work of the compile routine later, the input codes of some metasymbols are changed to negative code numbers whenever necessary. Sometimes two internal code numbers are necessary for the representation of a metasymbol. For example K is represented by two internal codes, "kst" (-5) and "kterm" (-9). "kst" signifies the start of the categories entered following by the categories and terminated by kterm. Hence for K <1, 2, 3> , it will be entered internally as

-5	1	2	3	-9

Note that metasymbols '<', ',', and '>' are not entered. The input code and internal representation of each metasymbol is shown below.

metasymbol	Corresponding identifier used in readsemantics		input code	internal representation
	for input code	for internal representation		
=	equals	nil	49	nil
<	ob	nil	44	nil
>	cb	nil	45	nil
,	comma	nil	42	nil
.	endin	endd	43	-3
;	termin	term	50	-13
&	elsein	else	52	-2
*N	nn	nl	28	-1
*S	ss	sp	33	0
C	cc	cc	17	17
L	ll	ll	26	26
P	pp	pp	30	30
W	ww	ww, wterm	37	37, -4
K	kk	kst, kterm	25	-5, -9

* The metasymbol N and S must be always of the form <N> and <S> and when combined with the metasymbol W would give the desired function.

W<<N>> would be interpreted by the compile routine as to skip a line, and W<<S>> would mean to skip a space, whereas W<N> and W<S> would mean print out character N and S respectively.

5. Again, each semantic rule is terminated by the metasymbol ';'. The next semantic rule is entered similarly as above until the terminator '0' for the semantic table is reached.

Also two Boolean variables kf and elf are set so that any violations of the rules as specified on page 25, can be checked and appropriate action taken.

2.7. The COMPILE routine for External Semantics.

The COMPILE routine has the same initial statements as on page 20. It is a recursive procedure which will make use of the analysis record as well as the input semantic table to perform the desired interpretation. Appropriate instructions corresponding to the metasymbols W, P, L, C and K as described on page 24 are executed in this routine and the resulting target string is stored in the array TARG ready for output. The

the maximum length of the target string. The numeric values in card 3 can in fact be changed whenever necessary.

Immediately following the initial data are the arbitrary number of user's documents. A typical user's document is shown below.

```
+++
DIFFERENTIATION OF ALGEBRAIC EXPRESSION ;
16, 80, 160, 1 ;
11 <primary>              =  <12>|<13>|<14>|<(><16><)> ;
12 <indept variable>      =  X;
13 <dependent variable>   =  Y;
14 <constant>             =  0 1|2|3|4|5|6|7|8|9 ;
15 <term>                 =  <11> (*<15>|<E>) ;
16 <expression>           =  <15> (+<16>|<E>) ;
0;
1;
11 <primary>              =  K <4> W <(> .
                             K <1, 2, 3, 4> C <1> .
                             K <4> W <)> .;
12 <indept variable>      =  W <1> ;
13 <dependent variable>   =  PW <'> ;
14 <constant>             =  W <0> ;
15 <term>                 =  K <1> W <(> C <1> W <*> L <2>
                             W <+> L <1> W <*> C <2> W <)> &
                             K <2> C <1> . ;
```

16 <expression> = C <1> K <1> W <+> C <2> . ;

0;

0;

16;

4 + X * Y ;

⋮

followed by any arbitrary number of input strings preceded with an assumed phrase type.

⋮

0;

+++

The first card begins with three consecutive '+' symbols to signify the beginning of a user's document. Following this card is any comment (excluding the character ;) identifying this document terminated with a semi-colon. The third card consists of parameters which provide informations about the maximum phrase type number required in the syntax table, the maximum size of the integer array DSYN, the maximum size of the integer array DSEM, and the number of the semantic sets required. All these parameters are separated by a comma or semi-colon. The numeric values of these parameters can be changed and adjusted as required. Following this card is the complete syntax table which

is ended with a terminator 0. The semantic sets are then entered each begins with its semantic set number followed by the corresponding semantic set which is terminated with a 0. An extra terminator, 0, is added to signify the end of the semantic sets. This is immediately followed by the source statements to be analyzed each of which is preceded with its assumed phrase type. The complete set of the source statements is similarly terminated with 0. The document is then completed with the three consecutive '+' characters. Any number of user's documents can be run at the same time. That is, different or the same problems can be solved by various documents while any mistakes that occurs in one document do not have any effect on any other documents. This is particularly desirable in testing runs or when the translator is used by many persons.

The complete set of user's documents is finally terminated with three terminal cards (terminal data).

Card 1	+++
Card 2	END OF RUN ;
Card 3	0;

Card 1 consists of three consecutive '+' characters and card 2 contains any comment concerning about the documents and ended with a semi-colon. Card 3 consists of the terminator 0. The complete input data listing for algebraic differentiation of restricted arithmetic expressions can be seen in Appendix A.

CHAPTER III

ALGEBRAIC DIFFERENTIATION

3.1. Syntax and Semantics for Differentiation.

The syntax employed for algebraic differentiation is shown on the opposite page.

This syntax for algebraic expressions observes the following conventions.

1. X is the independent variable.
   Y is the dependent variable.
2. No definition of ⟨identifier⟩ is given and it is assumed that all non-numeric constants and variables are represented by single letters.
3. The derivative of Y is indicated by a ' symbol. Successive uses of ' indicate higher order derivatives.
4. All the following trigonometric functions may be used in the algebraic expressions :
   exp, ln, sinh, cosh, tanh, coth, sech, csch, arcsin, arccos, arctan, arccot, arcsec, arccsc, sin, cos, tan, cot, sec, csc, arsinh, artanh, arcoth, arsech, arcsch.

   The argument of the function is enclosed in parenthesis as in normal programming practice.

SYNTAX FOR DIFFERENTIATION

```
11   <INDEPENDENT VARIABLE>=X;
12   <DEPENDENT VARIABLE>    =Y;
13   <CONSTANT>      =A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|Z;
16   <QUOTE>         ='(<16>|<E>);
17   <DIGITS>        =0|1|2|3|4|5|6|7|8|9;
18   <INTEGERS>      =<17>(<18>|<E>);
19   <FRACTION>      =<17>(<19>|<E>);
20   <UNSIGNED DECIMAL NUMBER> =<18>.<19>|.<19>|<18>;
21   <PRIMARY>       =<(><26><)>|<28>|<20>|<13>|<11>|<12>(<16>|<E>);
22   <SIGNED PRIMARY>=(+|-|<E>)<21>;
23   <FACTOR>        =<22>(@<23>|<E>);
24   <TERM1>         =<23>(/<24>|<E>);
25   <TERM2>         =<24>(*<25>|<E>);
26   <S.A.E.>        =<25>(+<26>|-<26>|<E>);
27   <STATEMENT>     =<26>(<=><26>|<E>);
28   <FUNCTION>      =(EXP|LN|SINH|COSH|TANH|COTH|SECH|CSCH|
                      ARCSIN|ARCCOS|ARCTAN|ARCCOT|ARCSEC|ARCCSC|
                      SIN|COS|TAN|COT|SEC|CSC|
                      ARSINH|ARCOSH|ARTANH|ARCOTH|ARSECH|ARCSCH)<(><26><)>;
```

5. There are five binary operators in the following (descending) order of precedence :

@	is used for exponentiation
/	is used for division
*	is used for multiplication
+, -	are used respectively for addition and subtraction
+ -	can also be used as unary operators.

A problem arises in connection with the associativity of the operators - , / and @. In ordinary mathematical convention operators of equal precedence are assumed to work from left to right (with the exception of @ which is from right to left), so that

A / B / C / D

is interpreted as

((A / B) / C) / D

The above syntax interprets the expression as

A / (B / (C /D))

since a left to right scanning syntax analyser is used. Parentheses are used in such cases to achieve the required associativity, and it is recommended that parentheses should always be inserted in all cases of this type.

With the syntax table so formed, all the following arithmetic expressions would be valid phrase types of 27.

```
Y = (1-X)/(1+X)
Y = (X@3+2*X@2-3*X-2)/(X@2-1)
Y = LN (2*TAN (X) +1)/ (TAN (X) +2))
Y = (SIN (2*X))@B
X*Y+X@2-1
Y*SIN(X)-X
Y'*SIN(X)+Y*(1)*COS(X))-1
X@3+Y@3-6*X*Y=0
Y' + COSH(X)/(1+SINH(X)@2)
Y = 5*X/3
A + B + C/D
45 - 3.6 - E
```

As an example take Y = 5*X/3 as a phrase type 27. The analysis record after 'search' would be as shown on the opposite page.

The target string is produced by application of the semantics to the analysis record.

The semantic rules in turn make use of the mathematical rules for algebraic differentiation. The table that follows indicates the mathematical rules

```
NEW SOURCE STATEMENT
TYPE         27
LENGTH        7

SOURCE  STATEMENT :
Y=5*X/3
ANALYSED SUCCESSFULLY AS TYPE         27
TABULAR ANALYSIS RECORD :
```

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	7	2	9
2	26	3	1	1	3	0
3	25	2	1	1	4	0
4	24	2	1	1	5	0
5	23	2	1	1	6	0
6	22	3	1	1	7	0
7	21	7	1	1	8	9
8	12	1	1	1	9	0
9	26	3	3	7	10	0
10	25	1	3	7	11	18
11	24	2	3	3	12	0
12	23	2	3	3	13	0
13	22	3	3	3	14	0
14	21	3	3	3	15	0
15	20	3	3	3	16	0
16	18	2	3	3	17	18
17	17	6	3	3	0	0
18	25	2	5	7	19	0
19	24	1	5	7	20	24
20	23	2	5	5	21	0
21	22	3	5	5	22	0
22	21	5	5	5	23	0
23	11	1	5	5	24	0
24	24	2	7	7	25	0
25	23	2	7	7	26	0
26	22	3	7	7	27	0
27	21	3	7	7	28	0
28	20	3	7	7	29	0
29	18	2	7	7	30	31
30	17	4	7	7	0	0

```
TARGET STRING PRODUCED :

Y'=(0*X/3+5*(1*3-X*0)/3@2)

****END OUTPUT THIS DOCUMENT
```

required. The symbol p is used to represent <simple arithmetic expression> (i.e. <S.A.E.>) as defined in the syntax table for <S.A.E.> . The expression 'a' prior to differentiation is indicated by p_a and the derivative of p_a is indicated by $p_{a.b}$ (a and b are integers).

Original form of the Expression	Differentiation rule for the Expression
$p_1 \pm p_2$	$p_{1.2} + p_{2.2}$
$p_1 * p_2$	$p_{1.2} * p_2 + p_1 * p_{2.2}$
p_2/p_1	$(p_{2.1}*p_1 - p_2*p_{1.2})/p_1@2$
$p_2@p_1$	$p_2@p_1*((p_1*p_{2.2})/p_2+p_{1.2}*\ln(p_2))$
$\ln(p_1)$	$p_{1.2}/p_1$
$\exp(p_1)$	$p_{1.2} * \exp(p_1)$
$\sin(p_1)$	$p_{1.2}*\cos(p_1)$
$\cos(p_1)$	$-(p_{1.2}*\sin(p_1))$
$\tan(p_1)$	$p_{1.2}*\sec(p_1)@2$
$\cot(p_1)$	$-p_{1.2}*\csc(p_1)@2$
$\sec(p_1)$	$p_{1.2}*\sec(p_1)*\tan(p_1)$
$\csc(p_1)$	$-p_{1.2}*\csc(p_1)*\cot(p_1)$
$\arcsin(p_1)$	$p_{1.2}/(1-p_1@2)@(1/2)$
$\arccos(p_1)$	$-p_{1.2}/(1-p_1@2)@(1/2)$
$\arctan(p_1)$	$p_{1.2}/(1+p_1@2)$

Original form of the Expression	Differentiation rule for the Expression
arccot (p_1)	$-p_{1.2}/(1+p_1@2)$
arcsec (p_1)	$p_{1.2}/(p_1*(p_1@2-1)@(1/2))$
arccsc (p_1)	$-p_{1.2}/(p_1*(p_1@2-1)@(1/2))$
sinh (p_1)	$p_{1.2}*\cosh(p_1)$
cosh (p_1)	$p_{1.2}*\sinh(p_1)$
tanh (p_1)	$p_{1.2}*\mathrm{sech}(p_1)@2$
coth (p_1)	$-p_{1.2}*\mathrm{csch}(p_1)@2$
sech (p_1)	$-p_{1.2}*\mathrm{sech}(p_1)*\tanh(p_1)$
csch (p_1)	$-p_{1.2}*\mathrm{csch}(p_1)*\coth(p_1)$
arsinh (p_1)	$p_{1.2}/(1+p_1@2)@(1/2)$
arcosh (p_1)	$p_{1.2}/(p_1@2-1)@(1/2)$
artanh (p_1)	$p_{1.2}/(1-p_1@2)$
arcoth (p_1)	$p_{1.2}/(1-p_1@2)$
arsech (p_1)	$-p_{1.2}/(p_1*(1-p_1@2)@(1/2))$
arcsh (p_1)	$-p_{1.2}/(p_1*(1+p_1@2)@(1/2))$

In addition to the list of rules above, it is understood that the following additional rules must be observed:

1. The derivative of independent variable is 1.
2. The first derivative of dependent variable (Y) is indicated by the symbol ' after the variable (Y'). Succeeding derivatives of Y would be indicated by additional ' on left of Y.

3. The derivatives of all constants and numbers are zero.

The semantics corresponding to the syntax table on page 36, can then be shown on the opposite page.

With these semantic rules the generation of the target string from Y = 5*X/3 according to the analysis record on page 39 would be Y'= (0*X/3+5*(1*3-X*0)/3@2). The mechanism that the compile routine has gone through the analysis record can be illustrated on the following table on page 44.

SEMANTIC TABLE FOR DIFFERENTIATION

```
11   <INDEPENDENT VARIABLE>=W<1>;
12   <DEPENDENT VARIABLE>=PW<'>;
13   <CONSTANT>      =W<0>;
16   <QUOTE>         =P;
20   <UNSIGNED DECIMAL NUMBER>=W<0>;
21   <PRIMARY>       =K<1>W<(>C<1>W<)>.
                     K<2,3,4,5,6,7>C<1>.
                      K<6>C<2>.;
22   <SIGNED PRIMARY>=K<2>W<->.
                       K<1,2,3>C<1>.;
23   <FACTOR>        =K<1>L<1>W<@>L<2>W<*((>L<2>W<*>C<1>W<)/>L<1>W<+>C<2>
                      W<*LN(>L<1>W<))>&K<2>C<1>.;
24   <TERM1>        =K<1>W<(>C<1>W<*>L<2>W<->L<1>W<*>C<2>W<)/>L<2>W<@2>&
                     K<2>C<1>.;
25   <TERM2>         =K<1>W<(>C<1>W<*>L<2>W<+>L<1>W<*>C<2>W<)>&
                     K<2>C<1>.;
26   <S.A.E.>      =K<1,2,3>C<1>.
                    K<1>W<+>.
                    K<2>W<->.
                    K<1,2>C<2>.;
27   <STATEMENT>     =C<1>K<1>W<=>C<2>.;
28   <FUNCTION>      =K<1>W<(>C<1>W<)*EXP(>L<1>W<)>&
                      K<2>W<(>C<1>W<)/(>L<1>W<)>&
                       K<3>W<(>C<1>W<)*COSH(>L<1>W<)>&
                       K<4>W<(>C<1>W<)*SINH(>L<1>W<)>&
                       K<5>W<(>C<1>W<)*(SECH(>L<1>W<))@2>&
                       K<6>W<(-(>C<1>W<))*(CSCH(>L<1>W<))@2>&
                       K<7>W<(-(>C<1>W<))*SECH(>L<1>W<)*TANH(>L<1>W<)>&
                       K<8>W<(-(>C<1>W<))*CSCH(>L<1>W<)*COTH(>L<1>W<)>&
                      K<9>W<(>C<1>W<)/(1-(>L<1>W<)@2)@(1/2)>&
                      K<10>W<(-(>C<1>W<))/(1-(>L<1>W<)@2)@(1/2)>&
                      K<11>W<(>C<1>W<)/(1+(>L<1>W<)@2)>&
                      K<12>W<(-(>C<1>W<))/(1+(>L<1>W<)@2)>&
                      K<13>W<(>C<1>W<)/((>L<1>W<)*((>L<1>W<)@2-1)@(1/2))>&
                      K<14>W<(-(>C<1>W<))/((>L<1>W<)*((>L<1>W<)@2-1)@(1/2))>&
                     K<15>W<(>C<1>W<)*COS(>L<1>W<)>&
                     K<16>W<(-(>C<1>W<))*SIN(>L<1>W<)>&
                     K<17>W<(>C<1>W<)*(SEC(>L<1>W<))@2>&
                     K<18>W<(-(>C<1>W<))*((CSC(>L<1>W<))@2>&
                     K<19>W<(>C<1>W<)*SEC(>L<1>W<)*TAN(>L<1>W<)>&
                     K<20>W<(-(>C<1>W<))*CSC(>L<1>W<)*COT(>L<1>W<)>&
                      K<21>W<(>C<1>W<)/(1+(>L<1>W<)@2)@(1/2)>&
                      K<22>W<(>C<1>W<)/((>L<1>W<)@2-1)@(1/2)>&
                      K<23>W<(>C<1>W<)/(1-(>L<1>W<)@2)>&
                      K<24>W<(>C<1>W<)/(1-(>L<1>W<)@2)>&
                      K<25>W<(-(>C<1>W<))/((>L<1>W<)*(1-(>L<1>W<)@2)@(1/2))>&
                      K<26>W<(-(>C<1>W<))/((>L<1>W<)*(1+(>L<1>W<)@2)@(1/2))>.;
```

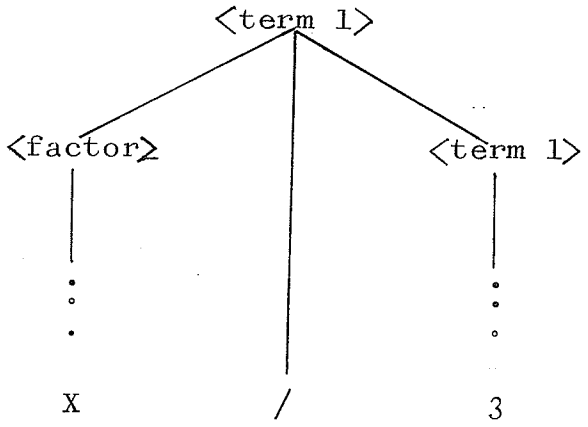
entry j	Source string interpreted	target string produced	Instructions executed	Compile stack
1			C <1>	0
2			C <1>	1
3			C <1>	2,1
4			C <1>	3,2,1
5			C <1>	4,3,2,1
6			C <1>	5,4,3,2,1
7			C <1>	6,5,4,3,2,1
8	Y	Y'	PW <'>	7,6,5,4,3,2,1
1	Y=	Y'=	W <=> C <2>	0
9	Y=	Y'=	C <1>	1
10	Y=	Y'=(	W <(> C <1>	9,1
11	Y=	Y'=(	C <1>	10,9,1
12	Y=	Y'=(	C <1>	11,10,9,1
13	Y=	Y'=(	C <1>	12,11,10,9,1
14	Y=	Y'=(	C <1>	13,12,11,10,9,1
15	Y=	Y'=(0	W <0>	14,13,12,11,10,9,1

entry j	Source string interpreted	target string produced	Instructions executed	Compile stack
*16	Y=	Y=(0	-	-
*17	Y=5	Y'=(0	-	-
10	Y=5*	Y'=(0*X/3+5*	W<*>L<2>W<+>L<1>W<*>C<2>	9,1
18	Y=5*	Y'=(0*X/3+5*	C <1>	10,9,1
19	Y=5*	Y'=(0*X/3+5*(	W<(>C<1>	18,10,9,1
20	Y=5*	Y'=(0*X/3+5*(	C<1>	19,18,10,9,1
21	Y=5*	Y'=(0*X/3+5*(	C<1>	20,19,18,10,9,1
22	Y=5*	Y'=(0*X/3+5*(	C<1>	21,20,19,18,10,9,1
23	Y=5*X	Y'=(0*X/3+5*(1	W<1>	22,21,20,19,18,10,9,1
19	Y=5*X/	Y'=(0*X/3+5*(1*3-X*	W<*>L<2>W<->L<1>W<*>C<2>	18,10,9,1
24	Y=5*X/	Y'=(0*X/3+5*(1*3-X*	C<1>	19,18,10,9,1
25	Y=5*X/	Y'=(0*X/3+5*(1*3-X*	C<1>	24,19,18,10,9,1
26	Y=5*X/	Y'=(0*X/3+5*(1*3-X*	C<1>	25,24,19,18,10,9,1
27	Y=5*X/	Y'=(0*X/3+5*(1*3-X*	C<1>	26,25,24,19,18,10,9,1
28	Y=5*X/	Y'=(0*X/3+5*(1*3-X*0	W<0>	27,26,25,24,19,18,10,9,1
*29	Y=5*X/	Y'=(0*X/3+5*(1*3-X*0	-	-

entry j	Source string interpreted	target string produced	Instructions executed	Compile stack
*30	Y=5*X/3	Y'=(0*X/3+5*(1*3-X*0	-	-
19	Y=5*X/3	Y'=(0*X/3+5*(1*3-X*0)/3@2	W<)/>L<2>W<@2>	18,10,9,1
10	Y=5*X/3	Y'=(0*X/3+5*(1*3-X*0)/3@2)	W<)>	9,1
1	Y=5*X/3	Y'=(0*X/3+5*(1*3-X*0)/3@2)	-	0

* These entries do not have corresponding semantic interpretations, however, this is no handicap since the interpretation is already handled by some preceding entries.

The sub-tree formed by entry 19 and its succeeding entries are responsible for the formation of the portion of the target string (1*3-X*0)/3@2 from the substring X/3 of the source string.



The complete instructions for <term 1> at entry 19 is

W<(>C<1>W<*>L<2>W<->L<1>W<*>C<2>W<)/>L<2>W<@ 2>

where C <1> produces a 1.

C <2> produces a 0.

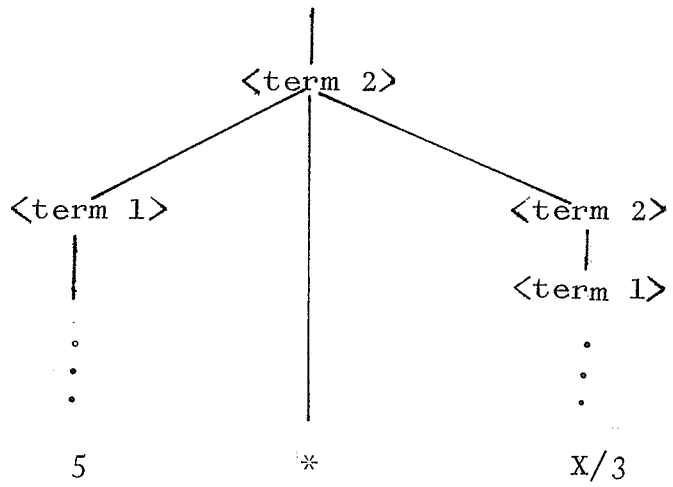
L <1> gives the first constituent of entry 19 i.e. X.

L <2> gives 3.

W <literal> gives the literal inside the cornered brackets.

Therefore the resulting output from entry 19 is (1*3-X*0)/3@2 which is, in fact, the derivative of X/3 .

Similarly the <term 2> at entry 10 corresponding to the formation of (0*X/3+5*(1*3-X*0)/3@2) from the substring 5*X/3.



The complete instructions for <term 2> at this entry 10 is

W<(>C<1>W<*>L<2>W<+>L<1>W<*>C<2>W<)>

where C <1> will finally produce a 5

C <2> compile recursively and finally goes down to entry 19 which gives the resulting differentiation on X/3 i.e. (1*3-X*0)/3@2

L <1> and L <2> gives respectively 5 and X/3.

The complete differentiation of 5*X/3 is (0*X/3+5*(1*3-X*0)/3@2). Note the derivative of a quotient rule takes precedence over the product rule which is valid mathematically. This is due to the fact that though the source string is searched from left to right, the order for operators of equal precedence works from right to left in our approach in contrast to the ordinary convention (Appendix C). However, this is no handicap in differentiation, since the order of working on strings with series of multipulation or addition and subtraction (e.g. A*B*C*D or A+B-C-D+E) would produce the same result though it may be in a different but equivalent form. The equal precedence for * and / assumed in conventional mathematics has been changed so that / takes precedence over *.

For operator / parentheses can always be inserted to ensure the proper order of operation. The derivative of 5*X/3 is in fact trivial, and is solely for illustrative purpose. In fact, the syntax and semantic table we used above can work on quite complicate expressions including implicit functions. A few solutions have been worked out from the algebraic expressions as shown. Note that the last three solutions give examples on implicit functions.

OUTPUT LISTING ON DIFFERENTIATION

```
SOURCE  STATEMENT :
Y=SIN(2*X)@B

TARGET STRING PRODUCED :
Y'=SIN(2*X)@B*((B*((0*X+2*1))*COS(2*X))/SIN(2*X)+0*LN(SIN(2*X)))


SOURCE  STATEMENT :
Y=ARCOSH(SEC(X))

TARGET STRING PRODUCED :
Y'=((1)*SEC(X)*TAN(X))/((SEC(X))@2-1)@(1/2)


SOURCE  STATEMENT :
Y=X@2*EXP(A/X)

TARGET STRING PRODUCED :
Y'=(X@2*((2*1)/X+0*LN(X))*EXP(A/X)+X@2*((0*X-A*1)/X@2)*EXP(A/X))


SOURCE  STATEMENT :
Y=(X@N-X@(-N))/X

TARGET STRING PRODUCED :
Y'=((X@N*((N*1)/X+0*LN(X))-X@(-N)*(((-N)*1)/X+(-0)*LN(X)))*X-(X@N-X@(-N))*1)/X@2


SOURCE  STATEMENT :
X*Y+X@2-1

TARGET STRING PRODUCED :
(1*Y+X*Y')+X@2*((2*1)/X+0*LN(X))-0


SOURCE  STATEMENT :
Y*SIN(X)-X

TARGET STRING PRODUCED :
(Y'*SIN(X)+Y*(1)*COS(X))-1


SOURCE  STATEMENT :
X@3+Y@3-6*X*Y=0

TARGET STRING PRODUCED :
X@3*((3*1)/X+0*LN(X))+Y@3*((3*Y')/Y+0*LN(Y))-(0*X*Y+6*(1*Y+X*Y'))=0



****END OUTPUT THIS DOCUMENT
```

3.2. Simplification of the Form of the Derivative.

It is obvious, from the examples given, that the derivatives obtained are mathematically inelegant and consist of some superfluous terms and redundant parentheses. In order to avoid these, two alternatives can be considered.

1. Redefine the syntax and semantic tables with provisions to handle constants or constant expressions.
2. Supply another syntax and semantic tables which will simplify the target string.

The second alternative is chosen for the reasons that it can demonstrate the otherwise obscure basic principles of differentiation and simplification.

In this simplification, no attempt is made to combine like terms or factors or to simplify trigonometric expressions, so that the expressions

```
A + B - 4 * A
A * C - A * D * SIN (X)
(X+1) * (X-1)/(X+1)
```

and `X * (1+TAN (X) @2)`

would remain unchanged and not simplify to

```
B - 3 * A
A * (C-D * SIN  (X))
X - 1
X * SEC (X) @2
```

since this would be in the more involved field of symbolic or polynomial manipulation [6, 7, 25, 36, 38], however, appropriate simplifications on terms with 1's and/or 0's are made. That is, the following basic transformations are always made.

A + 0 or 0 + A	⟶	A
A * 0 or 0 * A	⟶	0
1 * A or A * 1	⟶	A
A @ 0	⟶	1
A @ 1	⟶	A
1 @ A	⟶	1
0 @ A	⟶	0

Where A may be in any form from a single non-zero symbol A to an algebraic expression enclosed by parentheses, whereas 0(1) may be a digit 0(1) or any expression that after simplification becomes a 0(1).

In the syntax for simplification, we must define separately the non-zero expression, the zero expression, the various form of 1 and the various form of -1. In the corresponding semantic rules for the class 1 and -1 would be respectively W<1> and W<-1>. The various

combinations of the non-zero and zero expressions are also taken care of in the syntax definitions with the appropriate interpretation in the corresponding semantic rules.

The complete simplification syntax and semantic table designed for this purpose is on pages 54 and 55.

With this syntax and semantics the output strings of the differentiation on the expressions on page 50 can be fed in as source strings and produce simplified target strings as shown on page 56.

The appearance of some parentheses on some of these terms is in fact due to the omission of some of the zero and one terms. A complete omission of the parentheses is not intended for this syntax and semantic table (let's call it SIM1), but handled by a second syntax and semantic tables (call SIM2). This SIM2 will simplify the general expressions like

	A/((X)@2)	A/X@2
	A*(B/C)	A*B/C
or	A*((A+B*(C)))+C	A*(A+B*C)+C

The resulting expressions from SIM1 on page 54, when passed through SIM2 (page 57) would give outputs as shown on page 58.

SIM1

```
11   <ADD OP>                  =+|-;
12   <MULP OP>                 =*;
13   <DIV OP>                  =/;
14   <ZERO>                    =(<11>|<E>)0;
15   <ZERO TYPE>               =<11><(><15><)>|<(><15><)>|<14>(*<15>|<E>);
16   <ONE>                     =1;
17   <DIGITS>                  =2|3|4|5|6|7|8|9;
18   <NONZERO INT>             =<17>(<N>|<E>)|<16><N>|<14><18>;
19   <NONZERO UNSIGNED DEC NO.>=(<N>.|.|<E>)<18>;
20   <NONZERO DEC NO>          =(<11>|<E>)<19>;
21   <CONSTANT>                =<L>(<34>|<E>)|<20>;
22   <FUNCTION>     =<44><(><27><)>;
23   <PRIMARY>                 =<22>|<21>|<(><37><)>|<(><27><)>;
24   <FACTOR>                  =1@<24>|<23>(@<24>|@<15>|@<41>|<E>);
25   <TERM2>                   =1/<25>|<24>(/<25>|/<41>|<E>);
26   <TERM1>                   =(<41>|<43>)*<26>|
                                <41>*<41>|
                                <25>(*<26>|*<41>|*<43>|<E>);
27   <S.A.E>                   =<41><11><27>|<26>(<11><27>|<E>)|<41>|<42>;
28   <ZERO FACTOR>             =<31>(@<24>|<E>)|<15>;
29   <ZERO TERM2>              =<28>(/<26>|<E>);
30   <ZERO TERM1>              =<29>*<26>|(<24>*|<E>)<29>;
31   <ZERO EXP>                =<(><32><)>;
32   <ZERO ARITH EXP>          =<30>(<11><32>|<E>);
33   <EXPRESSION>              =<32>(+<27>|-<27>|<E>)|<26>(<11><32>|
                                <11><27>|<E>);
34   <QUOTE>                   ='(<34>|<E>);
36   <STATEMENT>               =<37>(<=><37>|<E>);
37   <SIGNED EXPRESSION>       =(<11>|<E>)<39>;
39   <COMBINE EXP.>            =<33>(<11><39>|<E>);
40   <POSITIVE ONE>            =0+1|1*1|1<11>0|1|+<40>|<(><40><)>;
41   <BRACKETED POS. ONE>      =<(><40><)>|<(>-<43><)>|-<43>|1;
42   <NEGATIVE ONE>            =0-1|-<41>;
43   <BRACKETED NEG ONE>       =<(><43><)>|<(><42><)>;
44   <FUNCTION NAME> =EXP|LN|SINH|COSH|TANH|COTH|SECH|CSCH|
                      ARCSIN|ARCCOS|ARCTAN|ARCCOT|ARCSEC|ARCCSC|
                      SIN|COS|TAN|COT|SEC|CSC|
                      ARSINH|ARCOSH|ARTANH|ARCOTH|ARSECH|ARCSCH;
0;
1;
11   <ADD OP>                  =P;
12   <MULP OP>                 =P;
13   <DIV OP>                  =P;
14   <ZERO>                    =W<0>;
15   <ZERO>                    =W<0>;
16   <ONE>                     =P;
17   <DIGITS>                  =P;
18   <NONZERO INT>             =K<1,2,3,4>C<1>.
                                K<1,3>L<2>.
                                K<4>C<2>.;
19   <NONZERO UNSIGNED DEC NO.>=K<1>L<1>W<.>C<2>&
                                  K<2>W<.>C<1>&
                                  K<3>C<1>.;
20   <NONZERO DEC NO>          =C<1>K<1>C<2>.;
21   <CONSTANT>                =K<1,2>L<1>.
                                K<1>C<2>.
                                K<3>C<1>.;
22   <FUNCTION>     =C<1>W<(>C<2>W<)>;
23   <PRIMARY>                 =K<3,4>W<(>.
```

```
                                  K<1,2,3,4>C<1>.
                                  K<3,4>W<)>.;
24   <FACTOR>                    =K<1,3>W<1>.
                                  K<2,4,5>C<1>.
                                  K<2>W<@>C<2>.;
25   <TERM2>                     =K<1>W<1/>C<1>.
                                  K<2,3,4>C<1>.
                                  K<2>W</>C<2>.;
26   <TERM1>                     =K<2,6>W<(->.
                                  K<4,5,6,7>C<1>.
                                  K<4>W<*>.
                                  K<1,2,4>C<2>.
                                  K<2,6>W<)>.
                                  K<3>W<1>.;
27   <S.A.E.>                    =K<1,2,3,4,5>C<1>.
                                  K<1,2>C<2>C<3>.;
33   <EXPRESSION>                =K<4,5,6>C<1>.
                                  K<2>W<->.
                                  K<3>W<0>.
                                  K<5>C<2>C<3>.
                                  K<1,2>C<2>.;
34   <QUOTE>                     =P;
36   <STATEMENT>                 =C<1>K<1>W<=>C<2>.;
37   <SIGNED EXPRESSION>         =C<1>K<1>C<2>.;
39   <COMBINE EXP.>              =C<1>;
41   <BRACKETED POS. ONE>        =W<1>;
42   <NEGATIVE ONE>              =W<-1>;
44   <FUNCTION NAME> =P;
```

OUTPUT LISTING AFTER SIM1

```
SOURCE  STATEMENT :
Y'=SIN(2*X)@B*((B*((0*X+2*1))*COS(2*X))/SIN(2*X)+0*LN(SIN(2*X)))

TARGET STRING PRODUCED :
Y'=SIN(2*X)@B*((B*((2))*COS(2*X))/SIN(2*X))


SOURCE  STATEMENT :
Y'=((1)*SEC(X)*TAN(X))/((SEC(X))@2-1)@(1/2)

TARGET STRING PRODUCED :
Y'=(SEC(X)*TAN(X))/((SEC(X))@2-1)@(1/2)


SOURCE  STATEMENT :
Y'=(X@2*((2*1)/X+0*LN(X))*EXP(A/X)+X@2*((0*X-A*1)/X@2)*EXP(A/X))

TARGET STRING PRODUCED :
Y'=(X@2*((2)/X)*EXP(A/X)+X@2*((-A)/X@2)*EXP(A/X))


SOURCE  STATEMENT :
Y'=((X@N*((N*1)/X+0*LN(X))-X@(-N)*(((-N)*1)/X+(-0)*LN(X)))*X-(X@N-X@(-N))*1)/X@2

TARGET STRING PRODUCED :
Y'=((X@N*((N)/X)-X@(-N)*(((-N))/X))*X-(X@N-X@(-N)))/X@2


SOURCE  STATEMENT :
(1*Y+X*Y')+X@2*((2*1)/X+0*LN(X))-0

TARGET STRING PRODUCED :
(Y+X*Y')+X@2*((2)/X)


SOURCE  STATEMENT :
(Y'*SIN(X)+Y*(1)*COS(X))-1

TARGET STRING PRODUCED :
(Y'*SIN(X)+Y*COS(X))-1


SOURCE  STATEMENT :
X@3*((3*1)/X+0*LN(X))+Y@3*((3*Y')/Y+0*LN(Y))-(0*X*Y+6*(1*Y+X*Y'))=0

TARGET STRING PRODUCED :
X@3*((3)/X)+Y@3*((3*Y')/Y)-(6*(Y+X*Y'))=0



****END OUTPUT THIS DOCUMENT
```

SIM2

```
11   <ADD OP>                  =+|-;
13   <QUOTE>                   ='(<13>|<E>);
14   <DECIMAL NUMBER>          =<N>.<N>|.<N>|<N>;
15   <CONSTANT>                =<L>(<13>|<E>)|<14>;
16   <POSITIVE CONSTANT>       =<15>|<(><16><)>;
17   <NEGATIVE CONSTANT>       =<(>-<15><)>|<(><17><)>;
18   <SYMBOL>                  =<16>|<17>;
19   <PRIMARY>                 =<31>|<29>|<30>|<18>|<(><(><24><)><)>|<(><24><)>;
20   <SIGNED PRIMARY>          =(+|-|<E>)<19>;
21   <FACTOR>                  =<(><20>@<20><)>|
                                <20>(@<30>|@<20>|<E>);
22   <TERM1>                   =<21>(/<22>|<E>);
23   <TERM2>                   =<22>(*<23>|<E>);
24   <S.A.E.>          =<23>(<11><24>|<E>);
25   <FUNCTION NAME> =EXP|LN|SINH|COSH|TANH|COTH|SECH|CSCH|
                      ARCSIN|ARCCOS|ARCTAN|ARCCOT|ARCSEC|ARCCSC|
                      SIN|COS|TAN|COT|SEC|CSC|
                      ARSINH|ARCOSH|ARTANH|ARCOTH|ARSECH|ARCSCH;
27   <STATEMENT>               =<24>(<=><24>|<E>);
28   <PROGRAM>=<27>;
29   <FUNCTION>                =<25><(><24><)>;
30   <BRAC CONS>               =<(><18>/<18><)>;
31   <BRACKETED TERM>          =<(><29><)>;
0;
1;
11   <ADD OP>                  =P;
13   <QUOTE>                   =P;
14   <DECIMAL NUMBER>          =K<1>L<1>W<.>L<2>&
                                K<2>W<.>L<1>&
                                K<3>L<1>.;
15   <CONSTANT>                =K<1>L<1>C<2>&
                                K<2>L<1>&
                                K<3>C<1>.;
16   <POSITIVE CONSTANT>       =C<1>;
17   <NEGATIVE CONSTANT>       =K<1>W<(->C<1>W<)>.
                                K<2>C<1>.;
18   <SYMBOL>                  =C<1>;
19   <PRIMARY>                 =K<1,2,3,4>C<1>.
                                K<5,6>W<(>C<1>W<)>.;
20   <SIGNED PRIMARY>          =K<2>W<->.
                                K<1,2,3>C<1>.;
21   <FACTOR>                  =K<1,2,3,4>C<1>.
                                K<1,3>W<@>C<2>.
                                K<2>W<@(>C<2>W<)>.;
22   <TERM1>                   =C<1>K<1>W</>C<2>.;
23   <TERM2>         =C<1>K<1>W<*>C<2>.;
24   <S.A.E.>        =C<1>K<1>C<2>C<3>.;
25   <FUNCTION NAME>           =P;
27   <STATEMENT>               =C<1>K<1>W<=>C<2>.;
28   <PROGRAM>=C<1>;
29   <FUNCTION>                =C<1>W<(>C<2>W<)>;
30   <BRAC CONS>               =C<1>W</>C<2>;
31   <BRACKETED TERM>          =C<1>;
```

OUTPUT LISTING AFTER SIM2

```
SOURCE  STATEMENT :
Y'=SIN(2*X)@B*((B*((2))*COS(2*X))/SIN(2*X))

TARGET STRING PRODUCED :
Y'=SIN(2*X)@B*((B*2*COS(2*X))/SIN(2*X))


SOURCE  STATEMENT :
Y'=(SEC(X)*TAN(X))/((SEC(X))@2-1)@(1/2)

TARGET STRING PRODUCED :
Y'=(SEC(X)*TAN(X))/(SEC(X)@2-1)@(1/2)


SOURCE  STATEMENT :
Y'=(X@2*((2)/X)*EXP(A/X)+X@2*((-A)/X@2)*EXP(A/X))

TARGET STRING PRODUCED :
Y'=(X@2*2/X*EXP(A/X)+X@2*((-A)/X@2)*EXP(A/X))


SOURCE  STATEMENT :
Y'=((X@N*((N)/X)-X@(-N)*(((-N))/X))*X-(X@N-X@(-N)))/X@2

TARGET STRING PRODUCED :
Y'=((X@N*N/X-X@(-N)*(-N)/X)*X-(X@N-X@(-N)))/X@2


SOURCE  STATEMENT :
(Y+X*Y')+X@2*((2)/X)

TARGET STRING PRODUCED :
(Y+X*Y')+X@2*2/X


SOURCE  STATEMENT :
(Y'*SIN(X)+Y*COS(X))-1

TARGET STRING PRODUCED :
(Y'*SIN(X)+Y*COS(X))-1


SOURCE  STATEMENT :
X@3*((3)/X)+Y@3*((3*Y')/Y)-(6*(Y+X*Y'))=0

TARGET STRING PRODUCED :
X@3*3/X+Y@3*((3*Y')/Y)-(6*(Y+X*Y'))=0



****END OUTPUT THIS DOCUMENT
```

In some long algebraic expressions a single pass through SIM1 and SIM2 may not result in its simplies form. For example, the target string

```
W''=((0*COS(X-Y)+0*(-(1-0))*SIN(X-Y))+(1*(-(0-1))*
    SIN(X-Y)+X*((-(0-0))*SIN(X-Y)+
    (-(0-1))*(1-0)*COS(X-Y))));
```

on passing through SIM1 and SIM2 would give

After SIM1

```
W''=((SIN(X-Y)+X*((-0)*SIN(X-Y)+COS(X-Y))));
```

After SIM2

```
W''=(SIN(X-Y)+X*((-0)*SIN(X-Y)+COS(X-Y)));
```

which is not the simplest result and if again passed through SIM1 and SIM2 the immediate and final results are

After SIM1

```
W''=(SIN(X-Y)+X*(COS(X-Y)))
```

After SIM2

```
W''=(SIN(X-Y)+X*COS(X-Y));
```

which is in fact the simplest form that can be achieved, since there is no more zero or one terms or presence of any superfluous parentheses. Actually we can pass the output on differentiation through SIM1 and SIM2 alternately until the target string that produced is of constant length.

3.3. Evaluating the Derivative at Assigned Points.

The above description is concerned with finding the derivative of a function and presenting the result as an algebraic expression. It may be required to calculate the numerical value of such a derivative. The system allows for this by rewriting COMPILE as a type procedure in which the resulting value of a component can be assigned. This process, however, cannot be described in the external semantics.

A type procedure is written to handle the mathematical calculations. The analysis record from the SEARCH routine can be made use of by this procedure via

j := link [i,j] ; i := type [j] ; k := cat [j] ;

as has been discussed on section 2.5.2. The syntax table used to define the source string is exactly the same as that for algebraic differentiation, but the interpretation of the syntax rules (semantics) is entirely different. For operators +, -, *, / they carry the ordinary mathematical function. Thus, if A = 5, B = 3, the expressions

C = A + B would set C = 8 after compilation

C = A - B would set C = 2 after compilation

C = A * B would set C = 15 after compilation

C = A / B would set C = 1.666667 after compilation

however C = A @ B is interpreted as C = A ** B where **

is the linary operator of exponentiation in the ALGOL programming language, and would give the result C = 125 after compilation. Floating point mathematics is assumed in all cases.

On the evaluation of functions we take advantages of the built-in functions that are provided in the ALGOL programming language.

EXP(X)

SIN(X)

COS(X)

ABS(X)

SQRT(X)

ARCTAN(X)

LN(X)

where X is an argument that could be a simple variable, arithmetic expression or another function. With these given built-in functions evaluation of other functions can be done with the following given formulas.

$$\tan x = \frac{\sin x}{\cos x}$$

$$\cot x = \frac{\cos x}{\sin x}$$

$$\sec x = \frac{1}{\cos x}$$

$$\csc x = \frac{1}{\sin x}$$

$$\arcsin x = \arctan\left(\frac{x}{\sqrt{1 - x^2}}\right)$$

$$\arccos x = \frac{\pi}{2} - \arctan\left(\frac{x}{\sqrt{1 - x^2}}\right)$$

$$\operatorname{arccot} x = \frac{\pi}{2} - \arctan x$$

$$\operatorname{arcsec} x = \arctan\left(\sqrt{x^2 - 1}\right) \quad \text{for } x \geqslant 0$$

$$\operatorname{arcsec} x = \arctan\left(\sqrt{x^2 - 1} - \pi\right) \quad \text{for } x < 0$$

$$\operatorname{arccsc} x = \arctan\left(\frac{1}{\sqrt{x^2 - 1}}\right) \quad \text{for } x \geqslant 0$$

$$\operatorname{arccsc} x = \arctan\left(\frac{1}{\sqrt{x^2 - 1}}\right) - \pi \quad \text{for } x < 0$$

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

$$\cosh x = \frac{e^x + e^{-x}}{2}$$

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\coth x = \frac{e^x + e^{-x}}{e^x - e^{-x}}$$

$$\operatorname{sech} x = \frac{2}{e^x + e^{-x}}$$

$$\operatorname{csch} x = \frac{2}{e^x - e^{-x}}$$

$$\sinh^{-1}\frac{x}{a} = \ln\left(\frac{x + \sqrt{x^2 + a^2}}{a}\right), \quad a > 0$$

$$\cosh^{-1}\frac{x}{a} = \ln\left(\frac{x + \sqrt{x^2 - a^2}}{a}\right), \quad a > 0$$

$$\tanh^{-1}\frac{x}{a} = \frac{1}{2}\ln\left|\frac{a+x}{a-x}\right|, \quad |x| < |a|$$

$$\coth^{-1}\frac{x}{a} = \frac{1}{2}\ln\left|\frac{a+x}{a-x}\right|, \quad |x| > |a|$$

$$\operatorname{sech}^{-1}\frac{x}{a} = \ln\left(\frac{a}{x} + \sqrt{\frac{a^2}{x^2} - 1}\right), \quad a > 0$$

$$\operatorname{csch}^{-1}\frac{x}{a} = \ln\left(\frac{a}{x} + \sqrt{\frac{a^2}{x^2} + 1}\right), \quad a > 0$$

The <u>value</u> routine that handles all the evaluations can then be shown on pages 64, 65 and 66.

The derivative of Y = ARCOSH (SEC(X)) would be Y' = ((1)*SEC(X)*TAN(X))/((SEC(X))@2-1)@(1/2) and if X = 1.000 radians, can be evaluated through this 'value' routine to give the numeric result of 1.851. The complete analysed result is shown on pages 67, 68 and 69.

In the syntax the operator / remains to take the higher precedence than * since it always provides the correct result as if they are of equal precedence and works from left to right. Hence the expression 3*5/4*3 would give the same result in both cases as illustrated below.

3*5/4*3	3*5/4*3
T_1	T_1
T_2	T_2
T_3	T_3

```
'REAL' 'PROCEDURE'  VALUE(I,J); 'VALUE' I,J;  'INTEGER' I,J;
  'BEGIN'  'INTEGER' K;  'REAL'  L,Y;
  'SWITCH' S:=INDEPENDENT VARIABLE,DEP VARIABLE,CONSTANT,ALPHABETS,
     DUMMY1,QUOTE,DIGITS,INTEGERS,FRACTION PART,UNSIGNED DECIMAL NO,
     PRIMARY,SIG PRIMARY,FACTOR,TERM1,TERM2,SAE;
  'SWITCH' SS:=STATEMENT,FUNCTION NAME;
  'BOOLEAN' LTAO,LTMO;
LTAO:=TAO;
LTMO:=TMO;
J:=LINK(/I,J/); I:=TYPE(/J/);  K:=CAT(/J/);
'IF' I>26 'THEN' 'GOTO' SS(/I-26/);       'GOTO' S(/I-10/);
INDEPENDENT VARIABLE:
        VALUE:=IX; 'GOTO' END;
DEP VARIABLE:
        PRINTCH(VAR(/2/)); 'GOTO' END;
CONSTANT: VALUE:=A(/K/); 'GOTO' END;
ALPHABETS: 'GOTO' END;
DUMMY1:  'GOTO' END;
QUOTE:
       OUTSYMBOL(1,'('''')',1);
       'IF' K=1 'THEN' VALUE(1,J);
       'GOTO' END;
DIGITS:    VALUE:=SA(/K/);
        'GOTO' END;
INTEGERS:'IF'K=1 'THEN' VALUE:=10*VALUE(1,J)+VALUE(2,J)
        'ELSE' VALUE:=VALUE(1,J);
        'GOTO' END;
FRACTION PART:
        'IF' K=1 'THEN' VALUE:=VALUE(1,J)/10+VALUE(2,J)/100
        'ELSE' VALUE:=VALUE(1,J)/10;
        'GOTO' END;
UNSIGNED DECIMAL NO:
        'IF' K=1 'THEN' VALUE:=VALUE(1,J)+VALUE(2,J)
        'ELSE' VALUE:=VALUE(1,J);
        'GOTO' END;
PRIMARY: 'IF' K=1 'THEN' 'BEGIN'
        TAO:='TRUE';
        TMO:='TRUE';
        VALUE:=VALUE(1,J);
        TAO:=LTAO;  TMO:=LTMO
                        'END'
  'ELSE' VALUE:=VALUE(1,J);
        'IF' K=6 'THEN' VALUE(2,J);
        'GOTO' END;
SIG PRIMARY:
        'IF' K=2 'THEN' VALUE:=-VALUE(1,J)
        'ELSE' VALUE:=VALUE(1,J);
        'GOTO' END;
FACTOR:  'IF' K=1 'THEN' 'BEGIN' L:=VALUE(2,J);
                        VALUE:=VALUE(1,J)**L
                        'END'
        'ELSE' VALUE:=VALUE(1,J);
        'GOTO' END;
TERM1:
        'IF' K=1 'THEN' 'BEGIN'
                 'IF' LTMO 'THEN' 'BEGIN'
                 Y:=VALUE(1,J);
                 TMO:='FALSE';
                 VALUE:=Y/VALUE(2,J)
```

```
                                        'END'
                'ELSE' VALUE:=VALUE(1,J)*VALUE(2,J)
                                'END'
     'ELSE' VALUE:=VALUE(1,J);
          'GOTO' END;
TERM2:
          'IF' K=1 'THEN' 'BEGIN'
          Y:=VALUE(1,J);
          TMO:='TRUE';
          VALUE:=Y*VALUE(2,J)
                                'END'
     'ELSE' VALUE:=VALUE(1,J);
         'GOTO' END;
SAE:      'IF' K=1 'THEN'
      'BEGIN'
      TMO:='TRUE';
      Y:=SIGN('IF' LTAO 'THEN' 1 'ELSE' -1)*VALUE(1,J);
      TAO:='TRUE'  ;
      TMO:='TRUE';
      VALUE:=Y+VALUE(2,J);
      TMO:=LTMO
      'END'
      'ELSE' 'IF' K=2 'THEN'
      'BEGIN'
      TMO:='TRUE';
      Y:=SIGN('IF' LTAO 'THEN' 1 'ELSE' -1)*VALUE(1,J);
      TAO:='FALSE';
      TMO:='TRUE';
      VALUE:=Y+VALUE(2,J);
      TMO:=LTMO
      'END'
          'ELSE' VALUE:=SIGN('IF' LTAO 'THEN' 1 'ELSE'-1)*VALUE(1,J);
          'GOTO' END;
STATEMENT:
          VALUE(1,J); OUTST('('='); VALUE:=VALUE(2,J);
          'GOTO' END;
FUNCTION NAME:
      'BEGIN'
'REAL' 'PROCEDURE'  EXP1(X); 'REAL' X;
     EXP1:=EXP(X)-EXP(-X);
'REAL' 'PROCEDURE' EXP2(X) ; 'REAL' X ;
     EXP2:=EXP(X)+EXP(-X);
            'SWITCH' S:=AEXP,ALN,SINH,COSH,TANH,COTH,SECH,CSCH,
ARCSIN,ARCCOS,AARCTAN,ARCCOT,ARCSEC,ARCCSC,ASIN,ACOS;
     'SWITCH' SS:=TAN,COT,SEC,CSC,ARSINH,ARCOSH,ARTANH,ARCOTH,ARSECH,
      ARCSCH;
'REAL' X,PI;
TAO:='TRUE'; TMO:='TRUE'  ;
X:=VALUE(1,J);  PI:=3.141593;
'IF' K<=16 'THEN' 'GOTO' S(/K/) 'ELSE' 'GOTO' SS(/K-16/);
AEXP:  VALUE:=EXP(X);  'GOTO' END;
ALN:  VALUE:=LN(X); 'GOTO' END;
SINH:  VALUE:=EXP1(X)/2; 'GOTO' END;
COSH:  VALUE:=EXP2(X)/2;  'GOTO' END;
TANH:  VALUE:=EXP1(X)/EXP2(X) ;  'GOTO' END;
COTH: 'IF' EXP1(X)=0 'THEN' 'GOTO' OVERFLO;
               VALUE:=EXP2(X)/EXP1(X);
     'GOTO' END;
SECH:  VALUE:=2/EXP2(X) ;  'GOTO' END;
CSCH: 'IF' EXP1(X)=0 'THEN' 'GOTO' OVERFLO;
```

```
                                VALUE:=2/EXP1(X);  'GOTO' END;
ARCSIN:  'IF' X=1 'THEN' 'GOTO' OVERFLO;
                VALUE:=ARCTAN(X/SQRT(1-X**2));  'GOTO' END;
ARCCOS:  'IF' X=1 'THEN' 'GOTO' OVERFLO;
                VALUE:=PI/2-ARCTAN(X/SQRT(1-X**2));  'GOTO' END;
AARCTAN:  VALUE:=ARCTAN(X);  'GOTO' END;
ARCCOT:  VALUE:=PI/2-ARCTAN(X);  'GOTO' END;
ARCSEC:  'IF' X>=0 'THEN' VALUE:=ARCTAN(SQRT(X**2-1))
       'ELSE' VALUE:=ARCTAN(SQRT(X**2-1)-PI);  'GOTO' END;
ARCCSC:  'IF' X>=0 'THEN' VALUE:=ARCTAN(1/SQRT(X**2-1))
       'ELSE' VALUE:=ARCTAN(1/SQRT(X**2-1))-PI;  'GOTO' END;
ASIN:  VALUE:=SIN(X);  'GOTO' END;
ACOS:  VALUE:=COS(X);  'GOTO' END;
TAN:  'IF' COS(X)=0 'THEN' 'GOTO' OVERFLO;
                VALUE:=SIN(X)/COS(X);  'GOTO' END;
COT:   'IF' SIN(X)=0 'THEN' 'GOTO' OVERFLO;
                VALUE:=COS(X)/SIN(X);  'GOTO' END;
SEC:  'IF' COS(X)=0 'THEN' 'GOTO' OVERFLO;
                VALUE:=1/COS(X);  'GOTO' END;
CSC:  'IF' SIN(X)=0 'THEN' 'GOTO' OVERFLO;
                VALUE:=1/SIN(X);  'GOTO' END;
ARSINH:  VALUE:=LN(X+SQRT(X**2+1));  'GOTO' END;
ARCOSH:  VALUE:=LN(X+SQRT(X**2-1)); 'GOTO' END;
ARTANH:  'IF' ABS(X)>=1 'THEN' 'GOTO' OVERFLO;
                VALUE:=0.5*LN(ABS((1+X)/(1-X)));  'GOTO' END;
ARCOTH:  'IF' ABS(X)<=1 'THEN' 'GOTO' OVERFLO;
                VALUE:=0.5*LN(ABS((1+X)/(X-1)));  'GOTO' END;
ARSECH:  VALUE:=LN(1/X+SQRT(1/X**2)-1);  'GOTO' END;
ARCSCH:  VALUE:=LN(1/X+SQRT(1/X**2)+1);
  END:  TAO:=LTAO;  TMO:=LTMO;
     'END'   ;
END:  'END' VALUE;
```

```
TYPE         27
LENGTH         16
HIGHIEST ORDER OF DIFFERENTIATION ACHIEVED          1
CODE FOR EVALUATION IS 1 ELSE 0,    CODE=           1

SOURCE  STATEMENT :
Y=ARCOSH(SEC(X))
 INDEPENDENT VARIABLE IS X DEPENDENT VARIABLE IS Y
ANALYSED SUCCESSFULLY AS TYPE          27
TABULAR ANALYSIS RECORD :
```

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	16	2	9
2	26	3	1	1	3	0
3	25	2	1	1	4	0
4	24	2	1	1	5	0
5	23	2	1	1	6	0
6	22	3	1	1	7	0
7	21	7	1	1	8	9
8	12	1	1	1	9	0
9	26	3	3	16	10	0
10	25	2	3	16	11	0
11	24	2	3	16	12	0
12	23	2	3	16	13	0
13	22	3	3	16	14	0
14	21	2	3	16	15	0
15	28	22	3	16	16	0
16	26	3	10	15	17	0
17	25	2	10	15	18	0
18	24	2	10	15	19	0
19	23	2	10	15	20	0
20	22	3	10	15	21	0
21	21	2	10	15	22	0
22	28	19	10	15	23	0
23	26	3	14	14	24	0
24	25	2	14	14	25	0
25	24	2	14	14	26	0
26	23	2	14	14	27	0
27	22	3	14	14	28	0
28	21	5	14	14	29	0
29	11	1	14	14	30	0

```
TARGET STRING PRODUCED :

Y'=((1)*SEC(X)*TAN(X))/((SEC(X))@2-1)@(1/2)
X =       1.000
ANALYSED SUCCESSFULLY AS TYPE          27
TABULAR ANALYSIS RECORD :
```

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	43	2	10
2	26	3	1	2	3	0
3	25	2	1	2	4	0
4	24	2	1	2	5	0
5	23	2	1	2	6	0
6	22	3	1	2	7	0
7	21	6	1	2	8	9
8	12	1	1	1	9	0
9	16	2	2	2	10	0
10	26	3	4	43	11	0
11	25	2	4	43	12	0

12	24	1	4	43	13	57
13	23	2	4	22	14	0
14	22	3	4	22	15	0
15	21	1	4	22	16	0
16	26	3	5	21	17	0
17	25	1	5	21	18	31
18	24	2	5	7	19	0
19	23	2	5	7	20	0
20	22	3	5	7	21	0
21	21	1	5	7	22	0
22	26	3	6	6	23	0
23	25	2	6	6	24	0
24	24	2	6	6	25	0
25	23	2	6	6	26	0
26	22	3	6	6	27	0
27	21	3	6	6	28	0
28	20	3	6	6	29	0
29	18	2	6	6	30	31
30	17	2	6	6	0	0
31	25	1	9	21	32	44
32	24	2	9	14	33	0
33	23	2	9	14	34	0
34	22	3	9	14	35	0
35	21	2	9	14	36	0
36	28	19	9	14	37	0
37	26	3	13	13	38	0
38	25	2	13	13	39	0
39	24	2	13	13	40	0
40	23	2	13	13	41	0
41	22	3	13	13	42	0
42	21	5	13	13	43	0
43	11	1	13	13	44	0
44	25	2	16	21	45	0
45	24	2	16	21	46	0
46	23	2	16	21	47	0
47	22	3	16	21	48	0
48	21	2	16	21	49	0
49	28	17	16	21	50	0
50	26	3	20	20	51	0
51	25	2	20	20	52	0
52	24	2	20	20	53	0
53	23	2	20	20	54	0
54	22	3	20	20	55	0
55	21	5	20	20	56	0
56	11	1	20	20	57	0
57	24	2	24	43	58	0
58	23	1	24	43	59	96
59	22	3	24	37	60	0
60	21	1	24	37	61	0
61	26	2	25	36	62	87
62	25	2	25	34	63	0
63	24	2	25	34	64	0
64	23	1	25	34	65	81
65	22	3	25	32	66	0
66	21	1	25	32	67	0
67	26	3	26	31	68	0
68	25	2	26	31	69	0
69	24	2	26	31	70	0
70	23	2	26	31	71	0
71	22	3	26	31	72	0
72	21	2	26	31	73	0
73	28	19	26	31	74	0
74	26	3	30	30	75	0
75	25	2	30	30	76	0

76	24	2	30	30	77	0
77	23	2	30	30	78	0
78	22	3	30	30	79	0
79	21	5	30	30	80	0
80	11	1	30	30	81	0
81	23	2	34	34	82	0
82	22	3	34	34	83	0
83	21	3	34	34	84	0
84	20	3	34	34	85	0
85	18	2	34	34	86	87
86	17	3	34	34	0	0
87	26	3	36	36	88	0
88	25	2	36	36	89	0
89	24	2	36	36	90	0
90	23	2	36	36	91	0
91	22	3	36	36	92	0
92	21	3	36	36	93	0
93	20	3	36	36	94	0
94	18	2	36	36	95	96
95	17	2	36	36	0	0
96	23	2	39	43	97	0
97	22	3	39	43	98	0
98	21	1	39	43	99	0
99	26	3	40	42	100	0
100	25	2	40	42	101	0
101	24	1	40	42	102	108
102	23	2	40	40	103	0
103	22	3	40	40	104	0
104	21	3	40	40	105	0
105	20	3	40	40	106	0
106	18	2	40	40	107	108
107	17	2	40	40	0	0
108	24	2	42	42	109	0
109	23	2	42	42	110	0
110	22	3	42	42	111	0
111	21	3	42	42	112	0
112	20	3	42	42	113	0
113	18	2	42	42	114	115
114	17	3	42	42	0	0

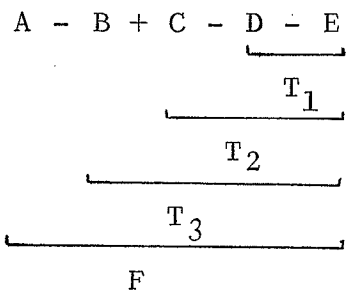
TARGET STRING PRODUCED :
Y'= 1.851

****END OUTPUT THIS DOCUMENT

or	$T_1/4*3$	$T_1 = 15$		$3*T_1*3$	$T_1 = 1.25$
	T_2*3	$T_2 = 3.75$		$3*T_2$	$T_2 = 3.75$
	T_3	$T_3 = 11.25$		T_3	$T_3 = 11.25$

where T_i indicates the immediate result obtained at step i.

Note that for the operators of equal precedence, the operation procedes from right to left according to the translator, flags must therefore be set on or off whenever necessary in order to carry the conventional order of operations. For example the expression A-B+C-D-E if no control flags appended in the 'value' routine the order of operation would be as shown.



It is obvious that such procedure of operation is wrong if it is assumed to work according to the conventional mathematics. Given A=3, B=4, C=2, D=3 and E=5 the intermediate values of T_1, T_2, T_3 are -2, 4, 8 respectively and the final value of the complete expression is -5 (i.e. F= -5) which is wrong. To achieve the correct result the above expression is interpreted as

ltao T F T F F

A + B +C + D + E

with the information for the sign of each variable stored in a logical flag ltao (true (T) status of which means positive whereas false (F) status means negative). The detection of the add and subtract operators is done by the global flag tao and the immediate information is passed to ltao to evaluate the sign of the current variable considered, while the status of tao may set to true or false depending on the subsequent add or subtract operators encountered.

The order of operation would be the same as before, but the result would be correct. The execution procedure is shown below.

ltao

A + B + C + D + E	
A + B + C + T_1	T_1 = -D-E = -3-5 = -8
A + B + T_2	T_2 = +2-8 = -6
A + T_3	T_3 = -4-6 = -10
F	F = +3-10= -7

Expressions with parentheses can be handled relatively easily with the following controls.

1. While the current status of tao is passed to ltao as before, the status of tao is set to

true status before the analysis of the bracketed expression.

2. After a similar analysis of the expression as before, the status of tao is restored by the instruction tao := ltao.

A similar control has been imposed on the division operator / for the reason that expressions of the form

A/B/C/D

will not be evaluated correctly without the control. Two flags, the global flag tmo and the local ltmo are similarly set up for the purpose. The flag tmo is initially set to true status whenever the operator / is encountered. Each time the information is passed to ltmo which determines the appropriate execution to be taken. Expressions with +, - and/or parentheses are similarly handled with the current status of tmo stored and restored via the local flag ltmo.

CHAPTER IV

MODIFICATIONS MADE ON THE TRANSLATOR

4.1. Introduction.

So far syntax and semantics tables have been given which will perform first order algebraic differentiation on an input expression with X, Y assumed to be independent and dependent variables respectively. This chapter discusses how, with little modification to the syntax and the translator, the syntax and semantics tables can be used to produce algebraic derivatives of any order (provided the length of the target string does not exceed the limit specified) and either the result can be in symbolic form or the numerical value of the result at assigned points can be evaluated through a routine VALUE. In the latter case the numeric values of all variables in a given input string must be supplied and will be handled by the input routine. The dependent and independent variables can be specified together with the input string and this information can be handled by the input routine which recognizes the dependent and independent variable.

4.2. Additional parameters for evaluation and order of differentiation.

In order to supply the information for the order of differentiation and numerical evaluation, two

parameters must be supplied.

The parameter for order of differentiation is an integer ORDER which can be any positive integer. A second parameter EVAL is used to indicate whether a numerical result (EVAL = 1) or simply the algebraic form of the result (EVAL = 0) is required. When a numerical result is required the values of the unknown variables should be given and the source string must be such that only the dependent variable is on the left side of the '=' symbol. These parameters are put to follow the phrase type number of each source string. Hence '27;1;0;' would mean the source string to be processed is of type 27, differentiation of first order is necessary and no evaluation for the result. The complete set of source statements is then terminated with '0;0;0;' instead of '0;' as discussed in section 2.8..

4.3. Internal Defined Syntax for dependent, independent variables and constants.

Since the independent and dependent variables are not necessarily X and Y, the phrase types <independent variable> , <dependent variable> , and <constant> in syntax table on page 36 must be deleted and handled by a special routine GETVARIABLE, while the phrase type <alphabets> defined to be any one of the upper case English letters is added to the syntax table. The variables chosen to be independent and dependent variables are provided immediately following the source string to be

differentiated. These variables are preceded by a '&' symbol which signifies that they are employed respectively as independent and dependent variables and the input routine will only take the character immediately following the mark '&' as the variable employed; and other characters preceding '&' or following the variable are ignored. An expression like $f(r,s) = e^{rs} \sin r \cos s$ when differentiation with respect to r is required is written in the input data as

```
27; 1; 0;
Y=EXP(R*S)*SIN(R)*COS(S);
INDEPENDENT VARIABLE IS &R DEPENDENT VARIABLE IS &Y;
```

The routine GETVARIABLE provides the internal codes for phrase types 11, 12, 13 which are equivalent to those specified by the syntax table on page 36, except that the independent and dependent variables may not be X and Y and the <constant> will consist of all the alphabets except the letters specified for independent and dependent variables. The GETVARIABLE routine listing is shown below.

```
procedure getvariable;
    begin integer  i,j, indx ;
  boolean    testor ;
  for  i := 1 step 1 until 2 do
    begin  for  j := readch  while  j ¬= 52 do ;
    var  [i]  := readch
    end ;
```

```
        sysact (1, 14, 1) ;
        outst ('( independent variable is ')');
        printch (var  [1] ) ;
        outst ('(' dependent variable is ')') ;
        printch (var  [2] ) ;
testor  := true
        indx  := ilfsyn  [13]  :=  7 ;
for  i  := 60    step 1  until  112  do
        begin  if dsyn [ i ] = var  [1]  then
           begin  ilfsyn  [11]  :=  1 ;
           dsyn [1]  := var [ 1 ]  ; dsyn [2 ]  := -12;
           dsyn [3]  := -13;
           testor  :=  false ;
           goto        endloop
           end ;
        if dsyn  [i] = var  [2]  then
           begin  ilfsyn  [12] := 4 ;
           dsyn [4] := var  [2] ;  dsyn [5 ]:= -12;
           dsyn [6] := -13;
           testor  :=  false ;
           goto        endloop
        end ;
        if testor then  begin
           dsyn [indx]  :=  dsyn  [i] ;
           indx  :=  indx + 1 ;
                       end
```

```
            else testor   := true ;
endloop :   end ;
            end getvariable ;
```

The starting points for the phrase types are indicated by the vector ILFSYN. The starting points for the phrase types concerned are shown as following

phrase type number i	phrase type	ILFSYN[i]
11	independent variable	1
12	dependent variable	4
13	constant	7
14	alphabets	60

4.4. Input routine for numeric data.

A routine VALUE has been designed to handle evaluation of derivatives and is discussed in section 3.3.. Here this input routine GETVALUES is discussed. The purpose of this routine is to read the numeric values for the necessary variables (including the numeric value for the independent variable). An array A [1 : 24] is reserved for storage of numeric values for the constants and is set to 0 initially. The real variable IX is also reserved for the numeric value of the independent variable. The input data for this routine is immediately following that specified for independent and dependent variables, and

it has the form

&variable 1 = numeric value ; &variable 2 = numeric value ; ..... &variable n = numeric value ; #

where variable 1, ... variable n are variables consisting of one letter only. The mark & precedes the variable has the same purpose as discussed in the last section. The mark # following the last semi-colon indicates the end of input information for the numerical values of the variables in the input string. All the numeric values are assumed to be real floating point numbers and will be automatically assigned to the corresponding constants by the routine GETVALUES. The listing of the routine is shown below.

```
integer  procedure readvalue ;
     begin  integer  i ;
start  : insymbol (0,'('ABCDEFGHIJKLMNOPQRSTUVWXYZ
                    &# *)',i);
     if  i = 29  then goto  start ;
     readvalue := i
     end ;
procedure getvalues ;
     begin integer  index, min, max ;
     if var  [1] <var  [2]  then begin
          max  := var  [2]  ;  min  :=  var  [1]
                                   end
```

```
else  begin
      max  := var [1] ;   min  :=  var [2]
      end ;
  min  :=  min - 14 ;
  max  :=  max  -14 ;
reread :  index  := readvalue ;
      if index  := 27  then goto readconst ;
      if index  = 28  then goto end ;
        goto  reread ;
readconst  :  index  :=  readvalue ;
              r  :=  read r ;
      sysact  (1, 14, 1) ;
      outsymbol (1,'('ABCDEFGHIJKLMNOPQRSTUVWXYZ
                &# ')', INDEX);
      oust ('(' = ')'); out r (R, true, 10, 3);
      if  index = var  [1] -14 then ix := r else
        A  [if index < min then index
            else if index < max then index -1
            else index -2 ]   := r ;
        goto  reread ;
end : end getvalues ;
```

4.5. Modification on the main program for co-ordination of the various subroutines.

In order to implement the above subroutines, the main program of the translator must be slightly modified. As has been mentioned, ORDER and EVAL are parameters set up respectively for the order of differentiation and the code for evaluation. The variables LORDER and LEVAL are set up for similar purposes but will work logically inside the block structure of the for statement. A Boolean variable TESTEVAL is also set up and has initially the value false. It is set true when the EVAL has the value of 1. When the variable is true the subroutine GETVALUES is executed in addition to the routine VALUE. When the value of the parameter ORDER is greater than 1 or the code EVAL is 1 the previous target string produced is to be used as the source string for the next higher algebraic differentiation or evaluation, so that entire target string is copied back as a source string for further processing. This has also been done in the main program. The portion of the main program that has been modified is shown below.

```
                    .
                    .
                    .
    readsyntax ;
    readsemantics;
    for  i := 1 step  1  until  10  do
                        sa  [1] :=  i -1 ;
```

```
newss : readss (pt);
        testeval := false; lorder := order; leval := eval;
findval : for  m := lorder  step  -1  until  1  do
        begin
          for  i := 1 step  1  until  26  do  A [ i ]  := 0;
        for i  := 1 step  1  until  length  do
        begin
            type  [i] :=  cat  [i] :=  from [i] := to [i] := 0;
        for j  := 1  step  1  until  nlinks  do link [j,i]  :=0
        end ;
        link [1, 0] := 1;  rpoint  := spoint  := 1 ;
        nextout  := 0 ;
        if testeval then
              begin
          tao  := true ; tmo  := true ;
          getvalues ;
              end
              else  getvariable;
       if search (pt) then
              begin
              mi('(' analysed successfully as type ')', pt);
                  sysact (1, 14, 1);
              outst ('(' tabular analysis record : ')');
                  sysact (1,  14, 2);
              outst (
      '('   entry  type  cat  from  to  links   ')');
              sysact (1, 14, 2);
```

```
for  i := 1  step  1  until  rpoint -1  do
    begin   outi (i, 10); outi (type [i] , 10);
outi (cat  [i] , 10); outi (from  [i] , 10);
outi (to  [i] , 10);
for  j := 1  step 1  until  nlinks  do
            outi (link [j, i] , 10);
    sysact (1, 14, 1)
    end ;
sysact (1, 14, 2); outst ('(' target string
    produced :')');
sysact (1, 14, 1);
if  testeval then begin
  result  := value (1, 0);
  out r  (result, true, 15, 3);
                  end
else              begin
  compile (1, 0, 1); sysact (1, 14, 1);
target print ;
if   nextout  >  =  lss then
       begin
         fault (20, false); goto newss
       end ;
for  i := 1  step 1  until nextout  do
  source  [i] :=  targ [i] ;
  nextin  :=  nextout;
  insource  (term);
                  end
end
```

```
    else
        begin
        mi ('(' absent as type ')', pt); sysact (1, 14, 1);
        outst ('(' no attempt at compilation possible ')');
       goto newss
       end ;
end ;
   if leval = 1  then begin testeval := true ;
                            lorder := 1; leval := 0;
                            goto findval
                      end ;
   goto newss ;
ssfail : mi('('**** overlength ss of type ')', pt);
             sysact (1, 14, 1);
```

CHAPTER V

CONCLUSION

5.1. Introduction.

This work shows how a syntax-directed translator can be used for algebraic differentiation instead of its usual task of translating a high-level programming language into an assembly language. We now discuss advantages and disadvantages of this system and some suggestions of future works on it.

5.2. Advantages of this system.

The main advantage of using a syntax-directed translator for symbol manipulation is versatility. The same translator can be used for different symbol manipulation problems (e.g. the translation of an algebraic expression into its reverse polish notation form, to obtain a derivative of an algebraic expression, a programming language from another programming language). The only data to be supplied is the necessary syntax and semantic tables and the specimen source statement to be translated. Since the translator is already a tested program, there is no need to recompile or modify the logic of the program each time when a different translation is required, instead, effort is spent on designing a efficient syntax and semantic tables for a

particular translation. The fact that the writing and altering the syntax and semantic specification is a relatively easy task as compared to the approach when a program is written specifically for a particular translation makes this approach a feasible one. Furthermore the error-checking and self-recovering facilities built in the translator offer great assistance in providing the correct syntax and semantic specifications especially during test runs.

5.3. Disadvantages.

The main disadvantage for employing a top-down syntax-directed translator on symbolic manipulation is that it is relatively slow. This is especially so when the translator is written in ALGOL which is slow in IBM/360 in general. The main reason that it is slow as compared to some other methods is the time spent on searching. No matter how clever the syntax designer is, the analyzer will spend a certain amount of time in exploring the blind alleys. In fact it is found that the time spent on analyzing an expression and get the output increase exponentially with the length of the source string.

A second disadvantage is that the relatively large amount of memory space required for the translator. The translator employed here together with data areas

requires a memory space of approximately 180K. The reasons for this are that the translator itself is already a large program and requires a fairly amount of core. Furthermore, large amount of space must also be allocated to the arrays like DSYN, ILFSYN, DSEM, and ILFSEM to store internally the syntax and semantic tables. Memory space must also be given to the array SOURCE for the storage of the input string, the arrays TYPE, CAT, FROM, TO, LINK to store the analysis record and the array TARG for the resulting target string. Some of the main memory space could be save if some secondary storage devices is used for some of the informations mentioned above.

5.4. Suggestions for future work.

In this work, this method has been proved quite satisfactory as far as flexibility is concerned. The additional syntax rules and the corresponding semantic rules can be added quite easily whenever required. Also any alphabet can be used to stand for any constant or variable desired by simply supplying the information together with the source string in the input data. It can also do partial derivatives, derivatives of higher order and the derivatives of implicit functions. The evaluation of the resulting output can also be possible. Perhaps the algebraic differentiation can be made in the form of

a specific command (say DIFF). For example the third derivative of $y^3 - 4xy + 5$ with respect to x can be written as

DIFF (Y@3 - 4*X*Y + 5, X, 3)

Partial derivatives can also be found similarly

e.g. $\dfrac{\partial^3(x \sin (x-y))}{\partial x \, \partial y^2}$

can be expressed as

DIFF (X*SIN (X-Y), X, 1, Y, 2)

The command for simplification can also be made similarly. For example, the above differentiation with simplification can be written as

SIM (DIFF (X*SIN (X-Y), X, 1, Y,2)).

Similarly specific command can be made on evaluation (say EVAL). The value of y cos(x@3) can be found by this command when the values of x and y are given.

e.g. EVAL (Y*COS(X@3), X=4, Y=7)

The numeric result of the third derivative of $y^3 - 4xy + 3$ can be found similarly with specified values of x and y.

EVAL ((DIFF(Y@3 - 4*X*Y+3), X, 3), X=2, Y=5)

All the above specific commands are suggestions but there seem to be no reasons why they cannot be implemented.

This command language can be further extended so that they can be used in an on-line system. This is very desirable especially under a time-sharing environment which allows many users to have easy access to the computer simultaneously. When a user has gain access to the system the word 'MESSAGE;' will be displayed on his terminal. He can reply with any one of the following commands.

(a) NEW.

(b) SAME.

(c) CONTINUE.

(d) Any one of the following

DIFF.

EVAL.

SIM.

(e) FINISH.

Each command in (d) corresponding to a set of syntax and semantics which has been previously feed in.

The functions of these commands can be explains as follows :

(a) When the command NEW. is typed in, the system will responds with another

MESSAGE ;

and await a source statement.

The necessary source statement is then feed in accordingly.

(b) Command SAME. is used when the current source statement is to be process the second time. Again the system will responds with

MESSAGE ;

and await next command instruction, which would, in fact, accept any of the above commands but presumably the user will reply with a command of type (d).

(c) If the user wishes to process the compiled version of the current source statement, he can type in CONTINUE. , and the program will copy this target string to the source statement array and reply with

COPY BACK ;

MESSAGE ;

The user can then feed in the command DIFF. if he want to obtain a higher derivative, or SIM. , if he want to simplify the source statement, or simply type in EVAL. to obtain the numerical value of the source statement. As in (c), the user can in fact feed in any of the commands but the above three are the

one most like to be used.

(d) When any one of these commands are fed in, the program will analyze the current source statement, and if the search is successful the message

SEARCH OK ;

is printed and start to assemble the target string via the COMPILE routine. The resulting target string is then printed out followed by the prompt

MESSAGE ;

If the search fails the message

SEARCH FAILS ;

is printed and followed by

MESSAGE ;

(e) When the user wishes to terminate the program he can simply feeds in the command

FINISH.

This on-line system has been tried in the University of Nottingham and has proved quite successful. Although the terminal equipment they used was the KDF9 monitor flexowriter which is, in fact, slow and unsuitable for this purpose. In fact, instead of the normal keypunch as input and high-speed printer as output, if we can devise some good display equipment

for the purpose it would be highly desirable. The expression like $y = x^2 \exp\left(\frac{A}{x}\right)$ would be keypunched in the linear form like

```
Y = X@2*EXP(A/X)
```

and the derivative of this expression printed on the line-printer would be

```
Y'= (X@2*((2*1)/X+0*LN(X))*EXP(A/X)
   +X@2*((0*X-A*1)/X@2)*EXP(A/X)
```

which is almost incomprensible without a lot of practice. If the special display device can accept the input as well as the resulting output expression in two-dimensional form like

$$Y = X^2 * EXP\left(\frac{A}{X}\right)$$

and

$$Y' = \left(X^2 * \left(\frac{2*1}{X} + 0 * LN(X)\right) * EXP\left(\frac{A}{X}\right) + X^2 * \left(\frac{0*X - A*1}{X^2}\right) * EXP\left(\frac{A}{X}\right)\right)$$

the advantage of this is obvious.

APPENDIX A

LISTING FOR DIFFERENTIATION OF RESTRICTED ARITHMETIC EXPRESSION

The following is a complete listing of the input data for the algebraic differentiation of restricted arithmetic expressions (e.g. 4 + X * Y) and the complete output listing for 4 + X * Y is also shown.

```
09
<>()|E;&.+=QU,WKLCPSNAZID
300;2;200;400;
+++
TEST DATA FOR ALGEBRAIC DIFFERENTION OF RESTRICTED ARITHMETIC EXPRESSIONS;
16,80,160,1;
11  <PRIMARY>       =<12>|<13>|<14>|<(><16><)>;
12  <INDEPT VARIABLE>=X;
13  <DEPENDENT VARIABLE>=Y;
14  <CONSTANT>      =0|1|2|3|4|5|6|7|8|9;
15  <TERM>          =<11>(*<15>|<E>);
16  <EXPRESSION>    =<15>(+<16>|<E>);
0;
1;
11  <PRIMARY>       =K<4>W<(>.
                     K<1,2,3,4>C<1>.
                     K<4>W<)>.;
12  <INDEPT VARIABLE>=W<1>;
13  <DEPENDENT VARIABLE>=PW<'>;
14  <CONSTANT>      =W<0>;
15  <TERM>          =K<1>W<(>C<1>W<*>L<2>
                     W<+>L<1>W<*>C<2>W<)>&
                     K<2>C<1>.;
16  <EXPRESSION>    =C<1>K<1>W<+>C<2>.;
0;
0;
16;
4+X*Y;
0;
+++
+++
END OF RUN;
0;
```

```
CODE FOR ZERO          1
CODE FOR NINE         10
CODE FOR OB           44
CODE FOR CB           45
CODE FOR ININ         46
CODE FOR IN          -10
CODE FOR OUTIN        47
CODE FOR OUT         -11
CODE FOR ORIN         51
CODE FOR OR          -12
CODE FOR EE           19
CODE FOR TERMIN       50
CODE FOR TERM        -13
CODE FOR ELSEIN       52
CODE FOR ELSE         -2
CODE FOR ENDDIN       43
CODE FOR ENDD         -3
CODE FOR STAR         11
CODE FOR EQUALS       49
CODE FOR OQ           31
CODE FOR CQ           35
CODE FOR COMMA        42
CODE FOR WW           37
CODE FOR KK           25
CODE FOR LL           26
CODE FOR CC           17
CODE FOR PP           30
CODE FOR SS           33
CODE FOR SP            0
CODE FOR NN           28
CODE FOR NL           -1
CODE FOR AA           15
CODE FOR ZZ           40
CODE FOR II           23
CODE FOR DD           18
CODE FOR WTERM        -4
CODE FOR KTERM        -9
CODE FOR KST          -5
MAX LENGTH RECORD          300
MAX NUMBER LINKS             2
MAX LENGTH SOURCE STRING          200
MAX LENGTH TARGET STRING          400
```

```
TEST DATA FOR ALGEBRAIC DIFFERENTION OF RESTRICTED ARITHMETIC EXPRESSIONS;

USER PARAMETERS FOR THIS DOCUMENT
LENGTH OF ILIFFE VECTORS          16
LENGTH OF SYNTAX VECTOR           80
LENGTH OF SEMANTICS VECTOR        160
NUMBER OF SEMANTIC SETS           1
READSYNTAX ENTERED

SYNTAX OK TYPE         11
SYNTAX OK TYPE         12
SYNTAX OK TYPE         13
SYNTAX OK TYPE         14
SYNTAX OK TYPE         15
SYNTAX OK TYPE         16


SYNTAX ENTERED   STORES OCCUPIED          64
POINTERS FOR SYNTAX
         0         0         0         0         0         0         0         0         0         0         1        16
        19        22        43        54
CONTENTS OF SYNTAX
        -1        12       -12        -1        13       -12        -1        14       -12        46        -1        16
        47       -12       -13        38       -12       -13        39       -12       -13         1       -12         2
       -12         3       -12         4       -12         5       -12         6       -12         7       -12         8
       -12         9       -12        10       -12       -13        -1        11       -10        13        -1        15
       -12       -12       -11       -12       -13        -1        15       -10        11        -1        16       -12
       -12       -11       -12       -13         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0


STARTED READING SEMANTIC SET         1
SEMANTICS OK TYPE         11
SEMANTICS OK TYPE         12
SEMANTICS OK TYPE         13
SEMANTICS OK TYPE         14
SEMANTICS OK TYPE         15
SEMANTICS OK TYPE         16


SEMANTICS ENTERED FOR SET NUMBER         1

STORES USED         90
POINTERS FOR THIS SET:
         0         0         0         0         0         0         0         0         0         0         1        26
        30        35        39        76
CONTENTS OF THIS SEMANTIC SET
        -5         4        -9        37        46        -4        -3        -5         1         2         3         4
        -9        17         1         1        -3        -5         4        -9        37        47        -4        -3
       -13        37         2        -4       -13        30        37        41        -4       -13        37         1
        -4       -13        -5         1        -9        37        46        -4        17         1         1        37
        13        -4        26         2        37        11        -4        26         1        37        13        -4
        17         2         1        37        47        -4        -2        -5         2        -9        17         1
         1        -3       -13        17         1         1        -5         1        -9        37        11        -4
        17         2         1        -3       -13


READSEMANTICS COMPLETE     TOTAL STORES OCCUPIED         89


NEW SOURCE STATEMENT
TYPE         16
```

```
LENGTH          5

SOURCE  STATEMENT :
4+X*Y
ANALYSED SUCCESSFULLY AS TYPE          16
TABULAR ANALYSIS RECORD :
```

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	16	1	1	5	2	5
2	15	2	1	1	3	0
3	11	3	1	1	4	0
4	14	5	1	1	0	0
5	16	2	3	5	6	0
6	15	1	3	5	7	9
7	11	1	3	3	8	0
8	12	1	3	3	0	0
9	15	2	5	5	10	0
10	11	2	5	5	11	0
11	13	1	5	5	0	0

```
TARGET STRING PRODUCED :

0+(1*Y+X*Y')

****END OUTPUT THIS DOCUMENT


END OF RUN;
****END OUTPUT FOR THIS SET OF DOCUMENTS
```

APPENDIX B

A COMPLETE LISTING OF THE TRANSLATOR

A slightly modified version of King's translator is shown. This translator can perform algebraic differentiation of higher order, partial derivatives, derivatives of implicit functions and evaluation of the result.

```
'BEGIN'
'COMMENT' THE FOLLOWING INPUT/OUTPUT ROUTINES HAVE BEEN ADDED TO THIS PROGRAM :

                    **  INPUT/OUTPUT TRANSFER ROUTINES  **

    INPUT FUNCTION PROCEDURES : READR, READI, READB

    OUTPUT ROUTINES :
        BASIC PROCEDURES :
                    REAL FORMAT(BOOL, FW, D)      OUTREAL(CHANNEL, R)
                    INTEGER FORMAT(FW)            OUTINTEGER(CHANNEL, I)
                    BOOLEAN FORMAT(FW)            OUTBOOLEAN(CHANNEL, B)

        FUNCTION PROCEDURES :
                    IOR(R,'('VAR NAME')', BOOL, FW, D)
                    IOI(I, '('VAR NAME')', FW)
                    IOB(B, '('VAR NAME')', FW)

        PROCEDURES :  IOA(A, L, U, '('ARRAY NAME')', BOOL, FW, D)

                    OTR(R, '('VAR NAME')', BOOL, FW, D)
                    OTI(I, '('VAR NAME')', FW)
                    OTB(B, '('VAR NAME')', FW)
                    OTA(A, L, U, '('ARRAY NAME')', BOOL, FW, D)

                    OUTR(R, BOOL, FW, D),   OUTI(I, FW),   OUTB(B, FW)
                    OUTA(A, L, U, BOOL, FW, D)
                    OUTIA(IA, L, U, FW)
                    OUTBA(BA, L, U, FW)

                    OUTST( STRING )

        ECHO TEXT FROM INPUT TO OUTPUT DEVICE :
                    COPYTEXT


                  **  INPUT/OUTPUT CONTROL ROUTINES  **

    DATA SET CNTRL :  OPEN(CHANNEL)   CLOSE(CHANNEL)
                      LINESIZE(N)     PAGESIZE(N)

    INPUT CNTRL :     NEXTCARD

    OUTPUT CNTRL :    SPACE(N)  TAB(N)  NEWLINES(N)  NEWPAGE  ;
```

```
00339   'BEGIN'
00339   'INTEGER'OB,CB,IN,OUT,ININ,OUTIN,OR,ORIN,TERM,TERMIN,
00339        ELSE,ELSEIN,ENDD,ENDDIN,STAR,EQUALS,OQ,CQ,COMMA,
00339        WW,KK,LL,CC,PP,SS,SP,NN,NL,AA,ZZ,ZERO,NINE,I,J,
00339        LILF,LDSYN,LDSEM,NS,NLINKS,LENGTH,RPOINT,SPOINT,
00339        WTERM,KTERM,KST,LSS,LTARG,NEXTIN,NEXTOUT,II,DD,EE,PT,EVAL,M,ORDER,
00339        LEVAL,LORDER;
00340   'REAL' RESULT,IX,R ;
00341   'BOOLEAN' TESTEVAL;
00342   'INTEGER' 'PROCEDURE' READVALUE;
00343        'BEGIN' 'INTEGER' I;
00344   START: INSYMBOL(0,'('ABCDEFGHIJKLMNOPQRSTUVWXYZ&# ')',I);
00345        'IF' I=29 'THEN' 'GOTO' START;
00346        READVALUE:=I
00346        'END';
00347   'INTEGER''PROCEDURE'READCH;
00348        'BEGIN'
00348        'INTEGER' I;
00349   START:
00349   INSYMBOL(0,'('0123456789+-*/ABCDEFGHIJKLMNOPQRSTUVWXYZ',.<>():=;|&¬@ ')'
00349         ,I);
00350        'IF' I = 55 'THEN' 'GOTO'  START;
00351        READCH:=I
00351        'END';
00352   'INTEGER''PROCEDURE' READINT;
00353        'BEGIN' 'INTEGER'I,J;
00354        'FOR' I:=READCH 'WHILE' 'NOT'  DIGIT(I) 'DO';
00355        I:=I-ZERO;
00356        'FOR' J:=READCH 'WHILE' DIGIT(J) 'DO'
00356             I:=I*10 + J-ZERO;
00357        READINT:=I
00357        'END';
00358   'PROCEDURE' PRINTCH(I); 'VALUE' I; 'INTEGER' I;
00361        'IF'I=-1 'THEN' SYSACT(1,14,1) 'ELSE'
00361         OUTSYMBOL(1,'('0123456789+-*/ABCDEFGHIJKLMNOPQRSTUVWXYZ',.<>():=;|
00361   &¬@')',I);
00362   'PROCEDURE' MI(S,I); 'VALUE'I; 'INTEGER'I; 'STRING' S;
00366        'BEGIN'
00366        SYSACT(1,14,1); OUTST(S);  OUTI(I,10)
00368        'END';
00369   'BOOLEAN''PROCEDURE' LETTER(I); 'VALUE' I; 'INTEGER' I;
00372        LETTER:= AA<=I 'AND' I<=ZZ;
00373   'BOOLEAN''PROCEDURE'DIGIT(I);  'VALUE' I; 'INTEGER' I;
00376        DIGIT:=ZERO<=I 'AND' I<=NINE ;
00377   OPEN(OUTCHANNEL);
00378   ZERO:=READCH;  NINE:=READCH;
00380   OB:=READCH;
00381   CB:=READCH;
```

```
 SC     SOURCE STATEMENT

00382   ININ:=READCH;  IN:=-10;
00384   OUTIN:=READCH;  OUT:=-11;
00386   ORIN:=READCH;  OR:=-12;
00388   EE:=READCH;
00389   TERMIN:=READCH;  TERM:=-13;
00391   ELSEIN:=READCH;  ELSE:=-2;
00393   ENDDIN:=READCH;  ENDD:=-3;
00395   STAR:=READCH;
00396   EQUALS:=READCH;
00397   OQ:=READCH;
00398   CQ:=READCH;
00399   COMMA:=READCH;
00400   WW:=READCH;
00401   KK:=READCH;
00402   LL:=READCH;
00403   CC:=READCH;
00404   PP:=READCH;
00405   SS:=READCH;  SP:=0;
00407   NN:=READCH;  NL:=-1;
00409   AA:=READCH;
00410   ZZ:=READCH;
00411   II:=READCH;
00412   DD:=READCH;
00413   WTERM:=-4;
00414   KTERM:=-9;
00415   KST:=-5;
00416   LENGTH:=READINT;
00417   NLINKS:=READINT;
00418   LSS:=READINT;
00419   LTARG:=READINT;
00420   MI('('MAX LENGTH RECORD ')',LENGTH);
00421   MI('('MAX NUMBER LINKS ')',NLINKS);
00422   MI('('MAX LENGTH SOURCE STRING ')',LSS);
00423   MI('('MAX LENGTH TARGET STRING ')',LTARG);
00424   NEWDOC:
00424   NEWPAGE;
00425   'FOR' I:=READCH 'WHILE' I=STAR 'DO';
00426   PRINTCH(I);
00427   COPYTEXT;
00428   LILF:=READINT; 'IF' LILF=0 'THEN' 'GOTO' END;
00430   LDSYN:=READINT;
00431   LDSEM:=READINT;
00432   NS:=READINT;
00433   SYSACT(1,14,2);
00434   OUTST('('USER PARAMETERS FOR THIS DOCUMENT')');
00435   MI('('LENGTH OF ILIFFE VECTORS ')',LILF);
00436   MI('('LENGTH OF SYNTAX VECTOR  ')',LDSYN);
00437   MI('('LENGTH OF SEMANTICS VECTOR ')',LDSEM);
```

```
SC      SOURCE STATEMENT

00438   MI('('NUMBER OF SEMANTIC SETS ')',NS);
00439        'BEGIN'
00439        'INTEGER''ARRAY'
00439             ILFSYN(/1:LILF/), DSYN(/1:LDSYN/),
00439             ILFSEM(/1:NS,1:LILF/), DSEM(/1:LDSEM/),
00439             SOURCE(/1:LSS/), TARG(/1:LTARG/),
00439             TYPE,CAT,FROM,TO(/1:LENGTH/),
00439             LINK(/1:NLINKS,0:LENGTH/),SA(/1:10/),VAR(/1:2/);
00440        'REAL' 'ARRAY'
00440            A(/1:26/);
00441        'BOOLEAN' TAO,TMO;
00442   'PROCEDURE' GETVARIABLE;
00443         'BEGIN' 'INTEGER' I,J,INDX;
00444        'BOOLEAN' TESTOR;
00445        'FOR' I:=1 'STEP' 1 'UNTIL' 2 'DO'
00445             'BEGIN' 'FOR' J:=READCH 'WHILE' J¬=52 'DO';
00446             VAR(/I/):=READCH
00446             'END';
00447            SYSACT(1,14,1);
00448             OUTST('(' INDEPENDENT VARIABLE IS ')');
00449             PRINTCH(VAR(/1/));
00450             OUTST('(' DEPENDENT VARIABLE IS ')' );
00451             PRINTCH(VAR(/2/));
00452   TESTOR:='TRUE';
00453       INDX:=ILFSYN(/13/):=7;
00454     'FOR' I:=60 'STEP' 1 'UNTIL' 112 'DO'
00454        'BEGIN' 'IF' DSYN(/I/)=VAR(/1/) 'THEN'
00454             'BEGIN' ILFSYN(/11/):=1;
00455   DSYN(/1/):=VAR(/1/);  DSYN(/2/):=-12;  DSYN(/3/):=-13;
00458             TESTOR:='FALSE';
00459             'GOTO' ENDLOOP
00459             'END';
00460        'IF' DSYN(/I/)=VAR(/2/) 'THEN'
00460        'BEGIN' ILFSYN(/12/):=4;
00461   DSYN(/4/):=VAR(/2/);  DSYN(/5/):=-12;  DSYN(/6/):=-13;
00464             TESTOR:='FALSE';
00465             'GOTO' ENDLOOP
00465        'END';
00466        'IF' TESTOR 'THEN' 'BEGIN'
00466       DSYN(/INDX/):=DSYN(/I/);
00467        INDX:=INDX+1;
00468                            'END'
00468        'ELSE' TESTOR:='TRUE';
00469   ENDLOOP:  'END';
00470        'END' GETVARIABLE;
00471   'PROCEDURE' GETVALUES;
00472        'BEGIN' 'INTEGER' INDEX,MIN,MAX;
00473        'IF' VAR(/1/)<VAR(/2/) 'THEN' 'BEGIN'
```

```
SC      SOURCE STATEMENT
00473             MAX:=VAR(/2/); MIN:=VAR(/1/)
00474             'END'
00474        'ELSE' 'BEGIN'
00474             MAX:=VAR(/1/); MIN:=VAR(/2/)
00475               'END';
00476        MIN:=MIN-14;
00477        MAX:=MAX-14;
00478   REREAD: INDEX:=READVALUE;
00479        'IF' INDEX=27 'THEN' 'GOTO' READCONST;
00480        'IF' INDEX=28 'THEN' 'GOTO' END;
00481        'GOTO' REREAD;
00482   READCONST:  INDEX:=READVALUE;
00483         R:=READ R;
00484        SYSACT(1,14,1);
00485        OUTSYMBOL(1,'('ABCDEFGHIJKLMNOPQRSTUVWXYZ&# ')',INDEX);
00486        OUTST('(' = ')') ;  OUT R(R,'TRUE',10,3);
00488        'IF' INDEX=VAR(/1/)-14 'THEN'IX:=R 'ELSE'
00488             A(/'IF' INDEX<MIN 'THEN' INDEX
00488             'ELSE' 'IF' INDEX<MAX 'THEN' INDEX-1
00488             'ELSE' INDEX-2/):=R;
00489        'GOTO' REREAD;
00490   END: 'END' GETVALUES;
00491        'PROCEDURE' FAULT(J,CAT); 'VALUE'J,CAT; 'BOOLEAN'CAT; 'INTEGER'J;
00495             'BEGIN'
00495             'IF' CAT 'THEN' SYSACT(1,14,4);
00496             MI('('****FAILURE NUMBER ')',J);
00497             SYSACT(1,14,1);
00498             'IF' CAT 'THEN'
00498                  'BEGIN'
00498                  OUTST('('****CATASTROPHIC FAILURE')');
00499                  SYSACT(1,14,1);
00500                  OUTST('('****REMAINDER OF DOCUMENT IGNORED')');
00501                  'FOR' I:=READCH 'WHILE'
00501                       'NOT' (I=STAR 'AND' READCH=STAR) 'DO' ;
00502                  'GOTO' NEWDOC
00502                  'END';
00503         OUTST('('****WARNING MESSAGE      DOCUMENT RESUMED')');
00504             SYSACT(1,14,1)
00504             'END';
00505        'PROCEDURE' READSYNTAX;
00506             'BEGIN' 'INTEGER' I,DP,TYPE,J,JJ;
00507             'PROCEDURE' ENTER(I); 'VALUE' I; 'INTEGER' I;
00510                  'BEGIN'
00510                  'IF' DP>LDSYN 'THEN' FAULT(1,'TRUE');
00511                  DSYN(/DP/):=I;  DP:=DP+1
00512                  'END';
00513             'FOR'I:=1 'STEP' 1 'UNTIL' LILF 'DO' ILFSYN(/I/):=0;
00514            'FOR' I:=1 'STEP' 1 'UNTIL' LDSYN 'DO'
```

```
SC     SOURCE STATEMENT
00514                          DSYN(/I/):=0;
00515            DP:=60;
00516            SYSACT(1,14,1);
00517            OUTST('('READSYNTAX ENTERED')');
00518            SYSACT(1,14,1);
00519            'FOR' TYPE:=READINT 'WHILE' TYPE¬=0 'DO'
00519              'IF' TYPE<=10  'OR' TYPE>LILF 'THEN' FAULT(2,'TRUE')
00519              'ELSE'
00519                 'BEGIN'
00519                 'IF' ILFSYN(/TYPE/) ¬= 0 'THEN' FAULT(3,'FALSE');
00520                 ILFSYN(/TYPE/):=DP;
00521                 'FOR' I:=READCH 'WHILE' I¬=EQUALS 'DO';
00522                 'FOR' I:=READCH 'WHILE' I¬=TERMIN 'DO'
00522                      'BEGIN'
00522                      'IF' I=ORIN 'THEN' ENTER(OR)
00522                      'ELSE' 'IF' I=ININ 'THEN' ENTER(IN)
00522                      'ELSE' 'IF' I=OUTIN 'THEN'
00522                           'BEGIN'
00522                           ENTER(OR); ENTER(OUT)
00523                           'END'
00523                      'ELSE' 'IF' I=OB 'THEN'
00523                           'BEGIN'
00523                           I:=READCH;
00524                           'IF' DIGIT(I) 'THEN'
00524                                'BEGIN'
00524                                ENTER(-1); I:=I-ZERO;
00526                                'FOR' J:=READCH 'WHILE' DIGIT(J) 'DO'
00526                                         I:=I * 10 + J - ZERO;
00527                                ENTER(I);
00528                                'IF' J ¬= CB 'THEN' FAULT(4,'TRUE')
00528                                'END'
00528                         'ELSE' 'BEGIN'
00528                                'IF' READCH ¬= CB 'THEN' FAULT(5,'TRUE');
00529                                'IF' I=LL 'OR' I=DD 'OR' I=II 'OR' I=NN
00529                                     'OR' I=SS 'THEN' ENTER(-1);
00530                                'IF' I ¬= EE 'THEN'
00530                                    ENTER('IF' I=LL 'THEN' 1
00530                                    'ELSE' 'IF' I=DD 'THEN' 2
00530                                   'ELSE' 'IF' I=II 'THEN' 4
00530                                   'ELSE' 'IF' I=NN 'THEN' 6
00530                                   'ELSE' 'IF' I=SS 'THEN' 8
00530                                   'ELSE'  I)
00530                                'END'
00530                           'END'
00530                      'ELSE' ENTER(I)
00530            'END';
00531               ENTER(OR);    ENTER(TERM);
00533            MI('('SYNTAX OK TYPE')',TYPE);
```

```
SC      SOURCE STATEMENT
00534                'END';
00535           SYSACT(1,14,2);
00536           MI('('SYNTAX ENTERED   STORES OCCUPIED ')',DP-1);
00537           SYSACT(1,14,1);
00538           OUTST('('POINTERS FOR SYNTAX')');
00539           SYSACT(1,14,1);
00540           'FOR' I:=1 'STEP' 1 'UNTIL' LILF 'DO' OUTI(ILFSYN(/I/),10);
00541           SYSACT(1,14,1);
00542           OUTST('('CONTENTS OF SYNTAX')');
00543           SYSACT(1,14,1);
00544           'FOR' I:=1 'STEP' 1 'UNTIL' LDSYN 'DO' OUTI(DSYN(/I/),10);
00545           SYSACT(1,14,2);
00546           'END' READSYNTAX ;
00547      'BOOLEAN''PROCEDURE' SEARCH(I); 'VALUE' I; 'INTEGER' I;
00550           'BEGIN'
00550           'INTEGER' THIS,DP,D,CATEG,DCATEG,T,SSTART,
00550                     LP,DRSTART,DSSTART,DLPOINT;
00551           'BOOLEAN' DEEP ;
00552           DEEP:='FALSE';
00553           SSTART:=SPOINT;  THIS:=RPOINT;
00555           'IF' I<=10 'THEN' 'GOTO' BIPS;
00556           'IF' I>LILF 'THEN' FAULT(6,'TRUE');
00557           RPOINT:=RPOINT+1;  DP:=ILFSYN(/I/);
00559           'IF' DP=0 'THEN'
00559                'BEGIN'
00559                SEARCH:='FALSE';
00560                FAULT(7,'FALSE');
00561                'GOTO' END
00561                'END';
00562           LP:=DCATEG:=CATEG:=1;
00563      NEXT:
00563           D:=DSYN(/DP/);
00564           'IF' D=OR 'THEN'
00564                'BEGIN'
00564                'IF' DEEP 'THEN'
00564                     'BEGIN'
00564                     'FOR' D:=DSYN(/DP/) 'WHILE' D¬=OUT 'DO'
00564                          'BEGIN'
00564                          'IF' D=OR 'THEN' DCATEG:=DCATEG+1;
00565                          DP:=DP+1
00565                          'END';
00566                     DCATEG:=DCATEG-1;
00567                     DP:=DP+1;
00568                     DEEP:='FALSE';
00569                     'GOTO' NEXT
00569                     'END';
00570      SUCCESS: SEARCH:='TRUE';
00571                TYPE(/THIS/):=I;      CAT(/THIS/):=CATEG;
```

```
SC      SOURCE STATEMENT
00573                   FROM(/THIS/):=SSTART;  TO(/THIS/):=SPOINT-1;
00575                   'GOTO' END
00575                   'END';
00576             'IF' D=TERM 'THEN'
00576                   'BEGIN'
00576             SEARCH:='FALSE';
00577                   RPOINT:=THIS;
00578                   'GOTO' END
00578                   'END';
00579             'IF' D=IN 'THEN'
00579                   'BEGIN'
00579                   DEEP:='TRUE'; DSSTART:=SPOINT;
00581                   DRSTART:=RPOINT; DP:=DP+1;
00583             DLPOINT:=LP; 'GOTO' NEXT
00584                   'END';
00585             'IF' D=OUT 'THEN'
00585                   'BEGIN'
00585                   CATEG:=CATEG-1;
00586                   DEEP:='FALSE';
00587                   'GOTO' TRYNEXTCATEGORY
00587                   'END';
00588             'IF' D=-1 'THEN'
00588                   'BEGIN'
00588                   DP:=DP+1;  T:=DSYN(/DP/);
00590                   LINK(/LP,THIS/):=RPOINT;
00591                   'IF' SEARCH(T) 'THEN'
00591                         'BEGIN'
00591                         LP:=LP+1; DP:=DP+1;
00593                         'GOTO' NEXT
00593                         'END';
00594                   'GOTO' TRYNEXTCATEGORY
00594                   'END';
00595             'IF' D=OUTSOURCE(SPOINT) 'THEN'
00595                   'BEGIN'
00595                   SPOINT:=SPOINT+1; DP:=DP+1;
00597                   'GOTO' NEXT
00597                   'END';
00598   TRYNEXTCATEGORY:
00598             'IF' DEEP 'THEN'
00598                   'BEGIN'
00598                   'FOR' D:=DSYN(/DP/) 'WHILE' D¬=OR 'DO' DP:=DP+1;
00599                   DP:=DP+1; CATEG:=CATEG+1;
00601                   LP:=DLPOINT;
00602             SPOINT:=DSSTART;
00603                   RPOINT:=DRSTART;
00604                   'GOTO' NEXT
00604                   'END';
00605             'FOR' DP:=DP+1 'WHILE' DSYN(/DP/)¬=OR 'DO'
```

```
SC      SOURCE STATEMENT
00605                'IF' DSYN(/DP/)=IN 'THEN'
00605                   'BEGIN'
00605                   'FOR' DP:=DP+1 'WHILE' DSYN(/DP/) ¬= OUT 'DO'
00605                     'IF' DSYN(/DP/)=OR 'THEN' CATEG:=CATEG+1;
00606                   CATEG:=CATEG-1
00606                   'END';
00607              DP:=DP+1;
00608              CATEG:=('IF' CATEG>DCATEG 'THEN' CATEG 'ELSE' DCATEG)+1;
00609              RPOINT:=THIS+1;   SPOINT:=SSTART;
00611              LP:=1;  'GOTO' NEXT;
00613         BIPS:
00613                   'BEGIN'
00613                   'SWITCH' S:=LETTR,DIGITT,L,IDENT,L,INTEGER,L,STRING,L,L;
00614                   D:=OUTSOURCE(SPOINT);
00615                   'GOTO'  S(/I/);
00616                L: FAULT(22,'FALSE');
00617                   SEARCH:='FALSE';
00618                   'GOTO' END;
00619            LETTR: 'IF' LETTER(D) 'THEN' 'GOTO' BOTH;
00620                   SEARCH:='FALSE';  'GOTO' END;
00622           DIGITT: 'IF' DIGIT(D) 'THEN'
00622                   BOTH: 'BEGIN'
00622                        SPOINT:=SPOINT+1;
00623                        CATEG:=D-('IF' I=2 'THEN' ZERO 'ELSE' AA);
00624                        RPOINT:=RPOINT+1;
00625                        'GOTO' SUCCESS
00625                        'END';
00626                   SEARCH:='FALSE';  'GOTO' END;
00628            IDENT: 'IF' LETTER(D) 'THEN'
00628                        'BEGIN'  'INTEGER' I;
00629                        SPOINT:=SPOINT+1;
00630                        CATEG   :=D;
00631                        I:=1;
00632                        'FOR' D:=OUTSOURCE(SPOINT) 'WHILE' LETTER(D)
00632                                                   'OR' DIGIT(D) 'DO'
00632                             'BEGIN'
00632                             SPOINT:=SPOINT+1;
00633                             'IF' I<=6 'THEN' CATEG   :=CATEG   *ZZ+D;
00634                             I:=I+1
00634                   'END';
00635                        RPOINT:=RPOINT+1;
00636                        'GOTO' SUCCESS
00636                        'END';
00637                   SEARCH:='FALSE';  'GOTO' END;
00639         INTEGER: 'IF' DIGIT(D) 'THEN'
00639                        'BEGIN'
00639                        SPOINT:=SPOINT+1;  CATEG   :=D-ZERO;
00641                        'FOR' D:=OUTSOURCE(SPOINT) 'WHILE' DIGIT(D) 'DO'
```

```
SC      SOURCE STATEMENT

00641                              'BEGIN'
00641                              SPOINT:=SPOINT+1;
00642                              CATEG:=10*CATEG + D - ZERO
00642                              'END';
00643                         RPOINT:=RPOINT+1;  'GOTO' SUCCESS
00644                         'END';
00645                    SEARCH:='FALSE';  'GOTO END;
00647          STRING: 'IF' D=OQ 'THEN'
00647                         'BEGIN'   'INTEGER' I,J;
00648                         I:=1; SPOINT:=SPOINT+1;  CATEG:=OUTSOURCE(SPOINT);
00651                         'FOR' SPOINT:=SPOINT,SPOINT+1'WHILE' I>0 'DO'
00651                              'BEGIN'   J:=OUTSOURCE(SPOINT);
00652                              'IF' J=OQ 'THEN' I:=I+1 'ELSE''IF' J=CQ'THEN'
00652                                   I:=I-1; J:=SPOINT
00653                              'END';
00654                         SPOINT:=J;
00655                         RPOINT:=RPOINT+1;
00656                         'GOTO'  SUCCESS
00656                         'END';
00657                    SEARCH:='FALSE';
00658                    'GOTO' END
00658                    'END'  BIPS;
00659          END: 'END'  SEARCH   ;
00660          'PROCEDURE'  READSEMANTICS ;
00661             'BEGIN'
00661             'INTEGER' N,DP,T,SSN,I,J,X;
00662             'BOOLEAN' KF,ELF;
00663             'PROCEDURE' IGNORE TO(Z); 'VALUE' Z; 'INTEGER' Z;
00666                   'BEGIN'  'INTEGER' I;
00667                   'FOR' I:=READCH 'WHILE' I¬=Z 'DO'
00667                   'END';
00668             'PROCEDURE' ENTER(I);  'VALUE' I; 'INTEGER' I;
00671                   'BEGIN'
00671                   'IF' DP>LDSEM 'THEN' FAULT(8,'TRUE');
00672                   DSEM(/DP/):=I;  DP:=DP+1
00673                   'END';
00674             'PROCEDURE' NUMBER(I,J); 'INTEGER' I,J;
00676                   'BEGIN'
00676                   I:=READCH - ZERO;
00677                   'FOR' J:=READCH 'WHILE' DIGIT(J) 'DO' I:=I*10 + J - ZERO
00677                   'END';
00678             'FOR' I:=1 'STEP' 1 'UNTIL' NS 'DO'
00678                 'FOR' J:=1 'STEP' 1 'UNTIL' LILF 'DO' ILFSEM(/I,J/):=0;
00679               'FOR' I:=1 'STEP' 1 'UNTIL' LDSEM 'DO'
00679                  DSEM(/I/):=0;
00680             DP:=1;
00681       START: SSN:=READINT;  N:=DP-1;
00683              'IF' SSN=0 'THEN' 'GOTO' END;
```

```
00684           MI('('STARTED READING SEMANTIC SET')',SSN);
00685           'FOR' T:=READINT 'WHILE' T¬=0 'DO'
00685                'BEGIN'
00685                'IF' T>LILF  'THEN' FAULT(23,'TRUE');
00686                'IF' ILFSEM(/SSN,T/)¬=0 'THEN' FAULT(24,'FALSE');
00687                ILFSEM(/SSN,T/):=DP;
00688                KF:=ELF:='FALSE';
00689                IGNORETO(EQUALS);
00690                'FOR' X:=READCH 'WHILE' X¬=TERMIN 'DO'
00690                     'BEGIN'
00690                     'IF' X=KK 'THEN'
00690                          'BEGIN'
00690                          'IF' KF 'THEN' FAULT(9,'TRUE');
00691                          KF:='TRUE';  IGNORETO(OB);
00693                          ENTER(KST);  ELF:='FALSE';
00695                     EKP: NUMBER(I,J);  ENTER(I);
00697                          'IF' J=COMMA 'THEN' 'GOTO' EKP;
00698                          'IF' J¬=CB  'THEN' FAULT(10,'TRUE');
00699                          ENTER(KTERM)
00699                          'END'
00699                'ELSE'   'BEGIN'
00699                          'IF' ELF 'THEN' FAULT(11,'TRUE');
00700                          'IF' X=ELSEIN 'THEN'
00700                               'BEGIN'
00700                               'IF' 'NOT' KF 'THEN' FAULT(12,'TRUE');
00701                               KF:='FALSE'; ELF:='TRUE';
00703                               ENTER(ELSE)
00703                               'END'
00703                'ELSE' 'IF' X=ENDDIN 'THEN'
00703                               'BEGIN'
00703                               'IF' 'NOT' KF 'THEN' FAULT(13,'TRUE');
00704                               KF:='FALSE'; ENTER(ENDD)
00705                               'END'
00705                'ELSE' 'IF' X=CC     'THEN'
00705                               'BEGIN'
00705                               IGNORETO(OB); ENTER(CC);
00707                               NUMBER(I,J); ENTER(I);
00709                               'IF' J=COMMA 'THEN' NUMBER(I,J)
00709                                          'ELSE' I:=SSN;
00710                               'IF' J¬=CB 'THEN' FAULT(14,'TRUE');
00711                               ENTER(I)
00711                               'END'
00711                'ELSE' 'IF' X=LL     'THEN'
00711                               'BEGIN'
00711                               IGNORETO(OB);  ENTER(LL);
00713                               NUMBER(I,J); ENTER(I);
00715                               'IF' J¬=CB 'THEN' FAULT(15,'TRUE')
00715                               'END'
```

```
SC      SOURCE STATEMENT
00715                   'ELSE' 'IF' X=PP      'THEN' ENTER(X)
00715                   'ELSE' 'IF' X=WW      'THEN'
00715                               'BEGIN'
00715                               IGNORETO(OB); ENTER(WW);
00717                               'FOR' X:=READCH 'WHILE' X¬=CB 'DO'
00717                                    'BEGIN'
00717                                    'IF' X=OB 'THEN'
00717                                         'BEGIN'
00717                                         X:=READCH;
00718                                         'IF' X=NN 'THEN' X:=NL
00718                                         'ELSE' 'IF' X=SS 'THEN' X:=SP;
00719                                         'IF' READCH¬=CB 'THEN'
00719                                                   FAULT(16,'TRUE')
00719                                         'END';
00720                                    ENTER(X)
00720                                    'END';
00721                               ENTER(WTERM)
00721                               'END'
00721                          'ELSE' FAULT(17,'TRUE')
00721                           'END'
00721                     'END';
00722                'IF' KF 'THEN'
00722                     'BEGIN'
00722                     ENTER(ENDD);  FAULT(18,'FALSE')
00723                     'END';
00724           ENTER(TERM);
00725           MI('('SEMANTICS OK TYPE')',T);
00726               'END';
00727           SYSACT(1,14,2);
00728           MI('('SEMANTICS ENTERED FOR SET NUMBER')',SSN);
00729                  SYSACT(1,14,1);
00730            MI('('STORES USED')',DP-N);
00731                  SYSACT(1,14,1);
00732         OUTST('('POINTERS FOR THIS SET: ')');
00733                  SYSACT(1,14,1);
00734             'FOR' I:=1 'STEP' 1 'UNTIL' LILF 'DO'
00734           OUTI(ILFSEM(/SSN,I/),10);
00735                  SYSACT(1,14,1);
00736           OUTST('('CONTENTS OF THIS SEMANTIC SET')');
00737                  SYSACT(1,14,1);
00738       'FOR' I:=N+1 'STEP' 1 'UNTIL' DP-1 'DO'
00738            OUTI(DSEM(/I/),10);
00739          'GOTO' START;
00740           END:   SYSACT(1,14,2);
00741     MI('('READSEMANTICS COMPLETE     TOTAL STORES OCCUPIED')',DP-1);
00742                  SYSACT(1,14,2);
00743          'END' READSEMANTICS ;
00744      'PROCEDURE' COPY(I);  'VALUE' I; 'INTEGER' I;
```

```
SC      SOURCE STATEMENT
00747           'BEGIN'  'INTEGER' P,Q,R;
00748           P:=FROM(/I/);   Q:=TO(/I/);
00750           'FOR' R:=P 'STEP' 1 'UNTIL' Q 'DO' INTARG(OUTSOURCE(R))
00750           'END';
00751         'PROCEDURE' COPYLINK(I,J);  'VALUE'I,J;  'INTEGER' I,J;
00754           COPY(LINK(/I,J/));
00755         'PROCEDURE' COMPILE(I,J,P);  'VALUE' I,J,P;  'INTEGER' I,J,P;
00758           'BEGIN'
00758           'INTEGER' K,DP,D;
00759           'BOOLEAN' TC;
00760           J:=LINK(/I,J/);  I:=TYPE(/J/);  K:=CAT(/J/);
00763           TC:='TRUE';
00764           DP:=ILFSEM(/P,I/);
00765           'IF' DP=0 'THEN'
00765                 'BEGIN'
00765                 FAULT(19,'FALSE'); 'GOTO' END
00766                 'END';
00767       NEXT: 'IF' 'NOT' TC 'THEN'
00767                 'BEGIN'
00767            'FOR' D:=DSEM(/DP/) 'WHILE' D¬=ELSE
00767                 & D¬=ENDD 'DO' DP:=DP+1;
00768                 'END';
00769           D:=DSEM(/DP/); 'IF' D=TERM 'THEN' 'GOTO' END;
00771           'IF' D=ELSE 'THEN'
00771                 'BEGIN'
00771                 DP:=DP+1;
00772                 'IF' TC 'THEN'
00772                       'BEGIN'
00772                       'FOR' D:=DSEM(/DP/) 'WHILE' D¬=ENDD 'DO' DP:=DP+1
00772                       'END'
00772            'ELSE' TC:='TRUE'
00772                 'END'
00772            'ELSE' 'IF' D=ENDD 'THEN'
00772                 'BEGIN'
00772                 TC:='TRUE';  DP:=DP+1
00773                 'END'
00773            'ELSE' 'IF' D=KST  'THEN'
00773                 'BEGIN'
00773                 TC:='FALSE'; DP:=DP+1;
00775                 'FOR' D:=DSEM(/DP/) 'WHILE' D¬=KTERM  'DO'
00775                       'BEGIN'
00775                       'IF' D=K 'THEN'  TC:='TRUE';
00776                       DP:=DP+1
00776                       'END';
00777                 DP:=DP+1
00777                 'END'
00777            'ELSE' 'IF' D=PP 'THEN'
00777                 'BEGIN'
```

```
SC      SOURCE STATEMENT
00777                   DP:=DP+1;  COPY(J)
00778                   'END'
00778              'ELSE' 'IF' D=LL 'THEN'
00778                   'BEGIN'
00778                   COPYLINK(DSEM(/DP+1/),J);
00779                   DP:=DP+2
00779                   'END'
00779              'ELSE' 'IF' D=CC 'THEN'
00779                   'BEGIN'
00779                   COMPILE(DSEM(/DP+1/),J,DSEM(/DP+2/));
00780                   DP:=DP+3
00780                   'END'
00780              'ELSE' 'IF' D=WW 'THEN'
00780                   'BEGIN'
00780                   DP:=DP+1;
00781                   'FOR' D:=DSEM(/DP/) 'WHILE' D¬=WTERM 'DO'
00781                        'BEGIN'
00781                        INTARG(D);
00782                        DP:=DP+1
00782                        'END';
00783                   DP:=DP+1
00783                   'END';
00784              'GOTO' NEXT;
00785         END: 'END' COMPILE  ;
00786   'REAL' 'PROCEDURE'  VALUE(I,J); 'VALUE' I,J;  'INTEGER' I,J;
00789     'BEGIN'  'INTEGER' K;  'REAL'  L,Y;
00791     'SWITCH' S:=INDEPENDENT VARIABLE,DEP VARIABLE,CONSTANT,ALPHABETS,
00791        DUMMY1,QUOTE,DIGITS,INTEGERS,FRACTION PART,UNSIGNED DECIMAL NO,
00791        PRIMARY,SIG PRIMARY,FACTOR,TERM1,TERM2,SAE;
00792     'SWITCH' SS:=STATEMENT,FUNCTION NAME;
00793     'BOOLEAN' LTAO,LTMO;
00794   LTAO:=TAO;
00795   LTMO:=TMO;
00796   J:=LINK(/I,J/); I:=TYPE(/J/);  K:=CAT(/J/);
00799   'IF' I>26 'THEN' 'GOTO' SS(/I-26/);       'GOTO' S(/I-10/);
00801   INDEPENDENT VARIABLE:
00801           VALUE:=IX; 'GOTO' END;
00803   DEP VARIABLE:
00803           PRINTCH(VAR(/2/)); 'GOTO' END;
00805   CONSTANT: VALUE:=A(/K/); 'GOTO' END;
00807   ALPHABETS: 'GOTO' END;
00808   DUMMY1:  'GOTO' END;
00809   QUOTE:
00809          OUTSYMBOL(1,'('''')',1);
00810          'IF' K=1 'THEN' VALUE(1,J);
00811          'GOTO' END;
00812   DIGITS:    VALUE:=SA(/K/);
00813           'GOTO' END;
```

```
SC      SOURCE STATEMENT
00814   INTEGERS:'IF'K=1 'THEN' VALUE:=10*VALUE(1,J)+VALUE(2,J)
00814            'ELSE' VALUE:=VALUE(1,J);
00815            'GOTO' END;
00816   FRACTION PART:
00816            'IF' K=1 'THEN' VALUE:=VALUE(1,J)/10+VALUE(2,J)/100
00816            'ELSE' VALUE:=VALUE(1,J)/10;
00817            'GOTO' END;
00818   UNSIGNED DECIMAL NO:
00818            'IF' K=1 'THEN' VALUE:=VALUE(1,J)+VALUE(2,J)
00818            'ELSE' VALUE:=VALUE(1,J);
00819            'GOTO' END;
00820   PRIMARY: 'IF' K=1 'THEN' 'BEGIN'
00820            TAO:='TRUE';
00821            TMO:='TRUE';
00822            VALUE:=VALUE(1,J);
00823            TAO:=LTAO;  TMO:=LTMO
00824                                'END'
00824      'ELSE' VALUE:=VALUE(1,J);
00825            'IF' K=6 'THEN' VALUE(2,J);
00826            'GOTO' END;
00827   SIG PRIMARY:
00827            'IF' K=2 'THEN' VALUE:=-VALUE(1,J)
00827            'ELSE' VALUE:=VALUE(1,J);
00828            'GOTO' END;
00829   FACTOR:  'IF' K=1 'THEN' 'BEGIN' L:=VALUE(2,J);
00830                             VALUE:=VALUE(1,J)**L
00830                             'END'
00830            'ELSE' VALUE:=VALUE(1,J);
00831            'GOTO' END;
00832   TERM1:
00832            'IF' K=1 'THEN' 'BEGIN'
00832                      'IF' LTMO 'THEN' 'BEGIN'
00832                      Y:=VALUE(1,J);
00833                      TMO:='FALSE';
00834                      VALUE:=Y/VALUE(2,J)
00834                                          'END'
00834                 'ELSE' VALUE:=VALUE(1,J)*VALUE(2,J)
00834                             'END'
00834        'ELSE' VALUE:=VALUE(1,J);
00835            'GOTO' END;
00836   TERM2:
00836            'IF' K=1 'THEN' 'BEGIN'
00836            Y:=VALUE(1,J);
00837            TMO:='TRUE';
00838            VALUE:=Y*VALUE(2,J)
00838                             'END'
00838        'ELSE' VALUE:=VALUE(1,J);
00839            'GOTO' END;
```

```
SC      SOURCE STATEMENT

00840   SAE:      'IF' K=1 'THEN'
00840         'BEGIN'
00840         TMO:='TRUE';
00841         Y:=SIGN('IF' LTAO 'THEN' 1 'ELSE' -1)*VALUE(1,J);
00842         TAO:='TRUE'  ;
00843         TMO:='TRUE';
00844         VALUE:=Y+VALUE(2,J);
00845         TMO:=LTMO
00845         'END'
00845         'ELSE' 'IF' K=2 'THEN'
00845         'BEGIN'
00845         TMO:='TRUE';
00846         Y:=SIGN('IF' LTAO 'THEN' 1 'ELSE' -1)*VALUE(1,J);
00847         TAO:='FALSE';
00848         TMO:='TRUE';
00849         VALUE:=Y+VALUE(2,J);
00850         TMO:=LTMO
00850         'END'
00850             'ELSE' VALUE:=SIGN('IF' LTAO 'THEN' 1 'ELSE'-1)*VALUE(1,J);
00851             'GOTO' END;
00852   STATEMENT:
00852             VALUE(1,J); OUTST('('=')'); VALUE:=VALUE(2,J);
00855             'GOTO' END;
00856   FUNCTION NAME:
00856         'BEGIN'
00856   'REAL' 'PROCEDURE'  EXP1(X); 'REAL' X;
00858        EXP1:=EXP(X)-EXP(-X);
00859   'REAL' 'PROCEDURE' EXP2(X) ; 'REAL' X ;
00861        EXP2:=EXP(X)+EXP(-X);
00862               'SWITCH' S:=AEXP,ALN,SINH,COSH,TANH,COTH,SECH,CSCH,
00862   ARCSIN,ARCCOS,AARCTAN,ARCCOT,ARCSEC,ARCCSC,ASIN,ACOS;
00863        'SWITCH' SS:=TAN,COT,SEC,CSC,ARSINH,ARCOSH,ARTANH,ARCOTH,ARSECH,
00863         ARCSCH;
00864   'REAL' X,PI;
00865   TAO:='TRUE'; TMO:='TRUE'  ;
00867   X:=VALUE(1,J);  PI:=3.141593;
00869   'IF' K<=16 'THEN' 'GOTO' S(/K/) 'ELSE' 'GOTO' SS(/K-16/);
00870   AEXP:  VALUE:=EXP(X);  'GOTO' END;
00872   ALN:  VALUE:=LN(X); 'GOTO' END;
00874   SINH:  VALUE:=EXP1(X)/2; 'GOTO' END;
00876   COSH:  VALUE:=EXP2(X)/2;  'GOTO' END;
00878   TANH:  VALUE:=EXP1(X)/EXP2(X) ;  'GOTO' END;
00880   COTH: 'IF' EXP1(X)=0 'THEN' 'GOTO' OVERFLO;
00881                  VALUE:=EXP2(X)/EXP1(X);
00882        'GOTO' END;
00883   SECH:  VALUE:=2/EXP2(X) ;  'GOTO' END;
00885   CSCH: 'IF' EXP1(X)=0 'THEN' 'GOTO' OVERFLO;
00886                                VALUE:=2/EXP1(X);  'GOTO' END;
```

```
SC     SOURCE STATEMENT
00888  ARCSIN:  'IF' X=1 'THEN' 'GOTO' OVERFLO;
00889                 VALUE:=ARCTAN(X/SQRT(1-X**2));  'GOTO' END;
00891  ARCCOS:  'IF' X=1 'THEN' 'GOTO' OVERFLO;
00892                 VALUE:=PI/2-ARCTAN(X/SQRT(1-X**2));  'GOTO' END;
00894  AARCTAN:  VALUE:=ARCTAN(X);  'GOTO' END;
00896  ARCCOT:  VALUE:=PI/2-ARCTAN(X);  'GOTO' END;
00898  ARCSEC:  'IF' X>=0 'THEN' VALUE:=ARCTAN(SQRT(X**2-1))
00898          'ELSE' VALUE:=ARCTAN(SQRT(X**2-1)-PI);  'GOTO' END;
00900  ARCCSC:  'IF' X>=0 'THEN' VALUE:=ARCTAN(1/SQRT(X**2-1))
00900          'ELSE' VALUE:=ARCTAN(1/SQRT(X**2-1))-PI;  'GOTO' END;
00902  ASIN:  VALUE:=SIN(X);  'GOTO' END;
00904  ACOS:  VALUE:=COS(X);  'GOTO' END;
00906  TAN:  'IF' COS(X)=0 'THEN' 'GOTO' OVERFLO;
00907                 VALUE:=SIN(X)/COS(X);  'GOTO' END;
00909  COT:   'IF' SIN(X)=0 'THEN' 'GOTO' OVERFLO;
00910                 VALUE:=COS(X)/SIN(X);  'GOTO' END;
00912  SEC:  'IF' COS(X)=0 'THEN' 'GOTO' OVERFLO;
00913                 VALUE:=1/COS(X);  'GOTO' END;
00915  CSC:  'IF' SIN(X)=0 'THEN' 'GOTO' OVERFLO;
00916                 VALUE:=1/SIN(X);  'GOTO' END;
00918  ARSINH:  VALUE:=LN(X+SQRT(X**2+1));  'GOTO' END;
00920  ARCOSH:  VALUE:=LN(X+SQRT(X**2-1)); 'GOTO' END;
00922  ARTANH:  'IF' ABS(X)>=1 'THEN' 'GOTO' OVERFLO;
00923                 VALUE:=0.5*LN(ABS((1+X)/(1-X)));  'GOTO' END;
00925  ARCOTH:  'IF' ABS(X)<=1 'THEN' 'GOTO' OVERFLO;
00926                 VALUE:=0.5*LN(ABS((1+X)/(X-1)));  'GOTO' END;
00928  ARSECH:  VALUE:=LN(1/X+SQRT(1/X**2)-1);  'GOTO' END;
00930  ARCSCH:  VALUE:=LN(1/X+SQRT(1/X**2)+1);
00931    END:  TAO:=LTAO;  TMO:=LTMO;
00933        'END'   ;
00934  END:  'END' VALUE;
00935       'PROCEDURE' INSOURCE(I); 'VALUE' I;  'INTEGER' I;
00938            'BEGIN'
00938            'IF' NEXTIN=LSS 'THEN'
00938                 'BEGIN'
00938                 FAULT(20,'FALSE');  'GOTO'  SSFAIL
00939                 'END';
00940            NEXTIN:=NEXTIN+1;  SOURCE(/NEXTIN/):=I
00941            'END';
00942       'INTEGER''PROCEDURE'OUTSOURCE(I); 'VALUE'I; 'INTEGER' I;
00945            OUTSOURCE:=SOURCE(/I/);
00946       'PROCEDURE' INTARG(I);  'VALUE' I; 'INTEGER' I;
00949            'BEGIN'
00949            'IF' NEXTOUT=LTARG 'THEN'
00949                 'BEGIN'
00949                 FAULT(21,'FALSE');   'GOTO'  TARGFAIL
00950                 'END';
00951            NEXTOUT:=NEXTOUT+1;  TARG(/NEXTOUT/):=I
```

```
SC      SOURCE STATEMENT
00952               'END';
00953         'INTEGER''PROCEDURE' OUTTARG(I); 'VALUE' I; 'INTEGER' I;
00956               OUTTARG:=TARG(/I/);
00957         'PROCEDURE' READSS(TYPE);  'INTEGER'  TYPE;
00959               'BEGIN' 'INTEGER' I;
00960               NEXTIN:=0;
00961               TYPE:=READINT; ORDER:=READINT;  EVAL:=READINT;
00964               'IF' TYPE=0 'THEN' 'GOTO' DOCEND;
00965               'FOR' I:=READCH 'WHILE' I¬=TERMIN 'DO' INSOURCE(I);
00966               INSOURCE(TERM);
00967                 NEXTIN:=NEXTIN-1;
00968               SYSACT(1,14,3);
00969               OUTST('('NEW SOURCE STATEMENT')');
00970               MI('('TYPE')',TYPE);
00971               MI('('LENGTH')',NEXTIN);
00972   MI('('HIGHIEST ORDER OF DIFFERENTIATION ACHIEVED')',ORDER);
00973   MI('('CODE FOR EVALUATION IS 1 ELSE 0,    CODE= ')',EVAL);
00974               SYSACT(1,14,2);
00975               OUTST('('SOURCE  STATEMENT :')');  SYSACT(1,14,1);
00977               'FOR' I:=1 'STEP' 1 'UNTIL' NEXTIN 'DO' PRINTCH(OUTSOURCE(I))
00977               'END';
00978         'PROCEDURE'  TARGPRINT;
00979               'FOR' I:=1 'STEP' 1 'UNTIL' NEXTOUT 'DO' PRINTCH(OUTTARG(I));
00980          READSYNTAX;
00981          READSEMANTICS;
00982         'FOR' I:=1 'STEP' 1 'UNTIL' 10 'DO' SA(/I/):=I-1;
00983   NEWSS: READSS(PT);
00984          TESTEVAL:='FALSE'; LORDER:=ORDER; LEVAL:=EVAL;
00987   FINDVAL: 'FOR' M:=LORDER'STEP' -1 'UNTIL' 1 'DO'
00987          'BEGIN'
00987           'FOR' I:=1 'STEP' 1 'UNTIL' 26 'DO' A(/I/):=0;
00988          'FOR' I:=1 'STEP' 1 'UNTIL' LENGTH 'DO'
00988                   'BEGIN'
00988                   TYPE(/I/):=CAT(/I/):=FROM(/I/):=TO(/I/):=0;
00989                   'FOR' J:=1 'STEP' 1 'UNTIL' NLINKS 'DO' LINK(/J,I/):=0
00989                   'END';
00990              LINK(/1,0/):=1; RPOINT:=SPOINT:=1;  NEXTOUT:=0;
00993          'IF' TESTEVAL 'THEN' 'BEGIN'
00993               TAO:='TRUE';  TMO:='TRUE';
00995           GETVALUES;
00996                              'END'
00996                    'ELSE'  GETVARIABLE;
00997         'IF' SEARCH(PT) 'THEN'
00997              'BEGIN'
00997              MI('('ANALYSED SUCCESSFULLY AS TYPE')',PT);  SYSACT(1,14,1);
00999              OUTST('('TABULAR ANALYSIS RECORD :')');   SYSACT(1,14,2);
01001              OUTST(
01001   '('        ENTRY     TYPE       CAT      FROM       TO           LINKS')'
```

```
SC      SOURCE STATEMENT

01001                    );   SYSACT(1,14,2);
01003              'FOR' I:=1 'STEP' 1 'UNTIL' RPOINT-1 'DO'
01003                   'BEGIN'  OUTI(I,10); OUTI(TYPE(/I/),10);
01005                   OUTI(CAT(/I/),10); OUTI(FROM(/I/),10); OUTI(TO(/I/),10);
01008                   'FOR' J:=1 'STEP' 1 'UNTIL' NLINKS 'DO'
01008                         OUTI(LINK(/J,I/),10);
01009                   SYSACT(1,14,1)
01009                   'END';
01010         SYSACT(1,14,2);   OUTST('('TARGET STRING PRODUCED :')');
01012         SYSACT(1,14,1);
01013         'IF' TESTEVAL 'THEN' 'BEGIN'
01013         RESULT:=VALUE(1,0);
01014            OUT R(RESULT,'TRUE',15,3);
01015                              'END'
01015   'ELSE'                     'BEGIN'
01015         COMPILE(1,0,1); SYSACT(1,14,1);
01017         TARGPRINT;
01018         'FOR' I:=1 'STEP' 1 'UNTIL' NEXTOUT 'DO'
01018         SOURCE(/I/):=TARG(/I/);
01019         NEXTIN:= NEXTOUT;
01020         INSOURCE(TERM);
01021                              'END'
01021              'END'
01021        'ELSE'
01021              'BEGIN'
01021              MI('('ABSENT AS TYPE')',PT);  SYSACT(1,14,1);
01023          OUTST('('NO ATTEMPT AT COMPILATION POSSIBLE')'); 'GOTO' NEWSS
01024              'END';
01025         'END';
01026         'IF'LEVAL=1 'THEN' 'BEGIN' TESTEVAL:='TRUE';
01027                                    LORDER:=1;LEVAL:=0; 'GOTO' FINDVAL
01029                               'END';
01030         'GOTO' NEWSS;
01031   SSFAIL: MI('('****OVERLENGTH SS OF TYPE')',PT); SYSACT(1,14,1);
01033        OUTST('('****PORTION OF SS READ :')'); SYSACT(1,14,1);
01035        'FOR' I:=1 'STEP' 1 'UNTIL' NEXTIN 'DO' PRINTCH(OUTSOURCE(I));
01036        SYSACT(1,14,1);
01037        OUTST('('****NO FURTHER OPERATION UPON THIS SS PERFORMED')');
01038        'FOR' I:=READCH 'WHILE' I¬=TERMIN 'DO';  'GOTO' NEWSS;
01040   TARGFAIL: SYSACT(1,14,1); OUTST('('****TARGET STRING OVERLENGTH')');
01042    SYSACT(1,14,1); OUTST('('****TARGET STRING SO FAR PRODUCED')');
01044        SYSACT(1,14,1);  TARGPRINT;  SYSACT(1,14,1);
01047        OUTST('('****NO FURTHER OPERATION UPON THIS SS PERFORMED')');
01048        'GOTO' NEWSS;
01049   OVERFLO:  SYSACT(1,14,1);
01050   OUTST('('**DENOMINATOR CONSISTS OF ZERO OR FUNCTION UNSOLABLE **')');
01051        SYSACT(1,14,1);
01052        OUTST('('***NO FURTHER OPERATION UPON THIS SS PERFORMED ***')');
```

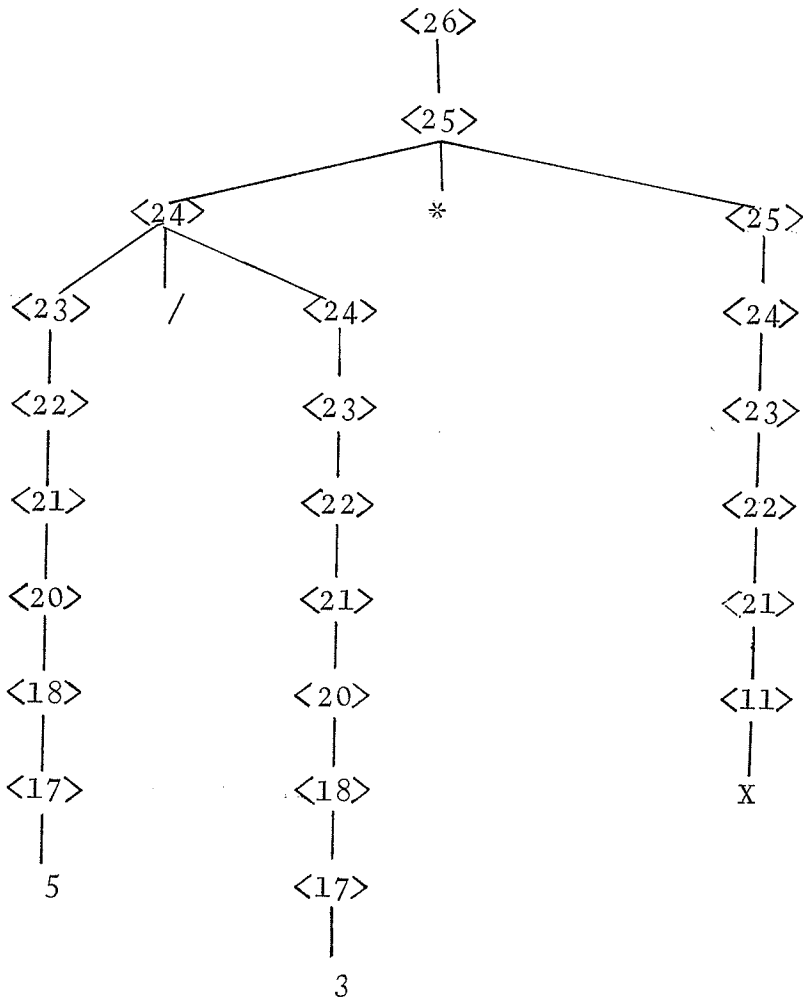
```
SC      SOURCE STATEMENT

01053         'GOTO' NEWSS;
01054   DOCEND: SYSACT(1,14,2); OUTST('('****END OUTPUT THIS DOCUMENT')');
01056         'FOR' I:=READCH 'WHILE' I¬=STAR 'DO'
01056         'END';
01057   'GOTO' NEWDOC;
01058   END: SYSACT(1,14,1);
01059                       OUTST('('****END OUTPUT FOR THIS SET OF DOCUMENTS')')
01059   'END'
01059
01059   'END' OF ALGOL PROGRAM
```

APPENDIX C

PROBLEM ON PRECEDENCE OF / AND *

As we have mentioned the order of precedence for / is assumed to be higher than that of * in order to achieve correct results.

When using the syntax on page 36, the expression 5/3 * X (assumed to be type 26) would have the following syntax tree.



Again assumed the given semantic rules as on page 43, the target string produced would be

((0*3-5*0)/3@2*X+5/3*1)

On simplification the result is 5/3 which is correct.

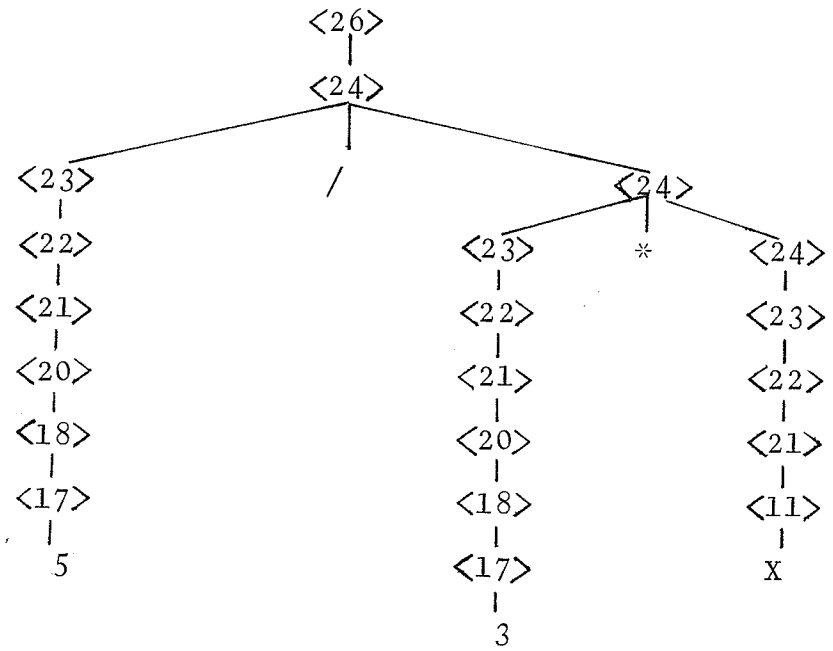
However, if the syntax are specified with * and / of equal precedence, so that the phrase types <term 1> and <term 2> of the syntax table on page 36 are combined as

24 <term> = <23> (* <24> | / <24> | <E>);

and phrase type <S.A.E.> is rewritten as

26 <S.A.E.> = <24> (+ <26> | - <26> | <E>);

the following syntax tree for 5/3*X would be formed.



With the appropriate semantic rules, the target string produced would be

(0*3*X-5*(0*X+3*1))/3*X@2

On simplification it is corresponding to $-5/3X^2$ which is actually the resulting differentiation from 5/(3*X). The reason is that though the operators / and * are specified as of equal precedence, the working order is from right to left in our translator, hence the product rule is applied first before the quotient rule, and the differentiation on 5/3*X would be as if it were 5/(3*X). In order to avoid this either parentheses are inserted as (5/3)*X to achieve the proper order for conventional arithmetic or redefine the syntax so that / takes higher precedence than * as in the given syntax on page 36. The latter approach has been chosen since it has been found that the resulting differentiation on a string with / and * operators always gives the equivalent result when the string is differentiated from left to right and with equal precedence on * and /.

APPENDIX D

COMPLETE OUTPUT LISTING FOR DIFFERENTIATIONS OF AN EXPRESSION

The following output listing showing the immediate symbolic and final numeric results of the expressions

$$\frac{\partial^2 x \cos(x-y)}{\partial x \, \partial y} \quad , \quad \frac{\partial^2 x \cos(x-y)}{\partial y \, \partial x}$$

The complete input data set for this algebraic differentiation is shown on the Appendix E that follows, while program listing of the translator employed is shown in Appendix B.

```
MAX LENGTH RECORD          500
MAX NUMBER LINKS             2
MAX LENGTH SOURCE STRING           400
MAX LENGTH TARGET STRING           400
```

```
DIFFERENTIATION OF ALGEBRIC EXPRESSION;

USER PARAMETERS FOR THIS DOCUMENT
LENGTH OF ILIFFE VECTORS          28
LENGTH OF SYNTAX VECTOR          450
LENGTH OF SEMANTICS VECTOR      1200
NUMBER OF SEMANTIC SETS            1
READSYNTAX ENTERED

SYNTAX OK TYPE          14
SYNTAX OK TYPE          16
SYNTAX OK TYPE          17
SYNTAX OK TYPE          18
SYNTAX OK TYPE          19
SYNTAX OK TYPE          20
SYNTAX OK TYPE          21
SYNTAX OK TYPE          22
SYNTAX OK TYPE          23
SYNTAX OK TYPE          24
SYNTAX OK TYPE          25
SYNTAX OK TYPE          26
SYNTAX OK TYPE          27
SYNTAX OK TYPE          28

SYNTAX ENTERED   STORES OCCUPIED        426
POINTERS FOR SYNTAX
        0       0       0       0       0       0       0       0       0       0       0       0
        0      60       0     113     122     143     153     163     177     204     215     226
      237     248     263     274
CONTENTS OF SYNTAX
        0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0      15
      -12      16     -12      17     -12      18     -12      19     -12      20     -12      21
      -12      22     -12      23     -12      24     -12      25     -12      26     -12      27
      -12      28     -12      29     -12      30     -12      31     -12      32     -12      33
      -12      34     -12      35     -12      36     -12      37     -12      38     -12      39
      -12      40     -12     -13      41     -10      -1      16     -12     -12     -11     -12
      -13       1     -12       2     -12       3     -12       4     -12       5     -12       6
      -12       7     -12       8     -12       9     -12      10     -12     -13      -1      17
      -10      -1      18     -12     -12     -11     -12     -13      -1      17     -10      -1
       19     -12     -12     -11     -12     -13      -1      18      43      -1      19     -12
       43      -1      19     -12      -1      18     -12     -13      46      -1      26      47
      -12      -1      28     -12      -1      20     -12      -1      13     -12      -1      11
      -12      -1      12     -10      -1      16     -12     -12     -11     -12     -13     -10
       11     -12      12     -12     -12     -11      -1      21     -12     -13      -1      22
      -10      54      -1      23     -12     -12     -11     -12     -13      -1      23     -10
       14      -1      24     -12     -12     -11     -12     -13      -1      24     -10      13
       -1      25     -12     -12     -11     -12     -13      -1      25     -10      11      -1
       26     -12      12      -1      26     -12     -12     -11     -12     -13      -1      26
      -10      49      -1      26     -12     -12     -11     -12     -13     -10      19      38
       30     -12      26      28     -12      33      23      28      22     -12      17      29
       33      22     -12      34      15      28      22     -12      17      29      34      22
      -12      33      19      17      22     -12      17      33      17      22     -12      15
       32      17      33      23      28     -12      15      32      17      17      29      33
      -12      15      32      17      34      15      28     -12      15      32      17      17
       29      34     -12      15      32      17      33      19      17     -12      15      32
       17      17      33      17     -12      33      23      28     -12      17      29      33
      -12      34      15      28     -12      17      29      34     -12      33      19      17
      -12      17      33      17     -12      15      32      33      23      28      22     -12
       15      32      17      29      33      22     -12      15      32      34      15      28
```

```
 22   -12    15    32    17    29    34    22   -12    15    32    33
 19    17    22   -12    15    32    17    33    17    22   -12   -11
 46    -1    26    47   -12   -13     0     0     0     0     0     0
  0     0     0     0     0     0     0     0     0     0     0     0
  0     0     0     0     0     0
```

```
STARTED READING SEMANTIC SET        1
SEMANTICS OK TYPE          11
SEMANTICS OK TYPE          12
SEMANTICS OK TYPE          13
SEMANTICS OK TYPE          16
SEMANTICS OK TYPE          20
SEMANTICS OK TYPE          21
SEMANTICS OK TYPE          22
SEMANTICS OK TYPE          23
SEMANTICS OK TYPE          24
SEMANTICS OK TYPE          25
SEMANTICS OK TYPE          26
SEMANTICS OK TYPE          27
SEMANTICS OK TYPE          28

SEMANTICS ENTERED FOR SET NUMBER        1

STORES USED      1067
POINTERS FOR THIS SET:
     0     0     0     0     0     0     0     0     0     0     1     5
    10     0     0    14     0     0     0    16    20    53    70   126
   170   207   239   253
CONTENTS OF THIS SEMANTIC SET
    37     2    -4   -13    30    37    41    -4   -13    37     1    -4
   -13    30   -13    37     1    -4   -13    -5     1    -9    37    46
    -4    17     1     1    37    47    -4    -3    -5     2     3     4
     5     6     7    -9    17     1     1    -3    -5     6    -9    17
     2     1    -3   -13    -5     2    -9    37    12    -4    -3    -5
     1     2     3    -9    17     1     1    -3   -13    -5     1    -9
    26     1    37    54    -4    26     2    37    13    46    46    -4
    26     2    37    13    -4    17     1     1    37    47    14    -4
    26     1    37    11    -4    17     2     1    37    13    26    28
    46    -4    26     1    37    47    47    -4    -2    -5     2    -9
    17     1     1    -3   -13    -5     1    -9    37    46    -4    17
     1     1    37    13    -4    26     2    37    12    -4    26     1
    37    13    -4    17     2     1    37    47    14    -4    26     2
    37    54     3    -4    -2    -5     2    -9    17     1     1    -3
   -13    -5     1    -9    37    46    -4    17     1     1    37    13
    -4    26     2    37    11    -4    26     1    37    13    -4    17
     2     1    37    47    -4    -2    -5     2    -9    17     1     1
    -3   -13    -5     1     2     3    -9    17     1     1    -3    -5
     1    -9    37    11    -4    -3    -5     2    -9    37    12    -4
    -3    -5     1     2    -9    17     2     1    -3   -13    17     1
     1    -5     1    -9    37    49    -4    17     2     1    -3   -13
    -5     1    -9    37    46    -4    17     1     1    37    47    13
    19    38    30    46    -4    26     1    37    47    -4    -2    -5
     2    -9    37    46    -4    17     1     1    37    47    14    46
    -4    26     1    37    47    -4    -2    -5     3    -9    37    46
    -4    17     1     1    37    47    13    17    29    33    22    46
    -4    26     1    37    47    -4    -2    -5     4    -9    37    46
    -4    17     1     1    37    47    13    33    23    28    22    46
    -4    26     1    37    47    -4    -2    -5     5    -9    37    46
    -4    17     1     1    37    47    13    46    33    19    17    22
    46    -4    26     1    37    47    47    54     3    -4    -2    -5
     6    -9    37    46    12    46    -4    17     1     1    37    47
    47    13    46    17    33    17    22    46    -4    26     1    37
```

47	47	54	3	-4	-2	-5	7	-9	37	46	12
46	-4	17	1	1	37	47	47	13	33	19	17
22	46	-4	26	1	37	47	13	34	15	28	22
46	-4	26	1	37	47	-4	-2	-5	8	-9	37
46	12	46	-4	17	1	1	37	47	47	13	17
33	17	22	46	-4	26	1	37	47	13	17	29
34	22	46	-4	26	1	37	47	-4	-2	-5	9
-9	37	46	-4	17	1	1	37	47	14	46	2
12	46	-4	26	1	37	47	54	3	47	54	46
2	14	3	47	-4	-2	-5	10	-9	37	46	12
46	-4	17	1	1	37	47	47	14	46	2	12
46	-4	26	1	37	47	54	3	47	54	46	2
14	3	47	-4	-2	-5	11	-9	37	46	-4	17
1	1	37	47	14	46	2	11	46	-4	26	1
37	47	54	3	47	-4	-2	-5	12	-9	37	46
12	46	-4	17	1	1	37	47	47	14	46	2
11	46	-4	26	1	37	47	54	3	47	-4	-2
-5	13	-9	37	46	-4	17	1	1	37	47	14
46	46	-4	26	1	37	47	13	46	46	-4	26
1	37	47	54	3	12	2	47	54	46	2	14
3	47	47	-4	-2	-5	14	-9	37	46	12	46
-4	17	1	1	37	47	47	14	46	46	-4	26
1	37	47	13	46	46	-4	26	1	37	47	54
3	12	2	47	54	46	2	14	3	47	47	-4
-2	-5	15	-9	37	46	-4	17	1	1	37	47
13	17	29	33	46	-4	26	1	37	47	-4	-2
-5	16	-9	37	46	12	46	-4	17	1	1	37
47	47	13	33	23	28	46	-4	26	1	37	47
-4	-2	-5	17	-9	37	46	-4	17	1	1	37
47	13	46	33	19	17	46	-4	26	1	37	47
47	54	3	-4	-2	-5	18	-9	37	46	12	46
-4	17	1	1	37	47	47	13	46	46	17	33
17	46	-4	26	1	37	47	47	54	3	-4	-2
-5	19	-9	37	46	-4	17	1	1	37	47	13
33	19	17	46	-4	26	1	37	47	13	34	15
28	46	-4	26	1	37	47	-4	-2	-5	20	-9
37	46	12	46	-4	17	1	1	37	47	47	13
17	33	17	46	-4	26	1	37	47	13	17	29
34	46	-4	26	1	37	47	-4	-2	-5	21	-9
37	46	-4	17	1	1	37	47	14	46	2	11
46	-4	26	1	37	47	54	3	47	54	46	2
14	3	47	-4	-2	-5	22	-9	37	46	-4	17
1	1	37	47	14	46	46	-4	26	1	37	47
54	3	12	2	47	54	46	2	14	3	47	-4
-2	-5	23	-9	37	46	-4	17	1	1	37	47
14	46	2	12	46	-4	26	1	37	47	54	3
47	-4	-2	-5	24	-9	37	46	-4	17	1	1
37	47	14	46	2	12	46	-4	26	1	37	47
54	3	47	-4	-2	-5	25	-9	37	46	12	46
-4	17	1	1	37	47	47	14	46	46	-4	26
1	37	47	13	46	2	12	46	-4	26	1	37
47	54	3	47	54	46	2	14	3	47	47	-4
-2	-5	26	-9	37	46	12	46	-4	17	1	1
37	47	47	14	46	46	-4	26	1	37	47	13
46	2	11	46	-4	26	1	37	47	54	3	47
54	46	2	14	3	47	47	-4	-3	-13		

READSEMANTICS COMPLETE TOTAL STORES OCCUPIED 1066

NEW SOURCE STATEMENT

```
TYPE        27
LENGTH        12
HIGHIEST ORDER OF DIFFERENTIATION ACHIEVED          2
CODE FOR EVALUATION IS 1 ELSE 0,    CODE=           1

SOURCE  STATEMENT :
W=X*COS(X-Y)
 INDEPENDENT VARIABLE IS X DEPENDENT VARIABLE IS W
ANALYSED SUCCESSFULLY AS TYPE         27
TABULAR ANALYSIS RECORD :
```

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	12	2	9
2	26	3	1	1	3	0
3	25	2	1	1	4	0
4	24	2	1	1	5	0
5	23	2	1	1	6	0
6	22	3	1	1	7	0
7	21	7	1	1	8	9
8	12	1	1	1	9	0
9	26	3	3	12	10	0
10	25	1	3	12	11	16
11	24	2	3	3	12	0
12	23	2	3	3	13	0
13	22	3	3	3	14	0
14	21	5	3	3	15	0
15	11	1	3	3	16	0
16	25	2	5	12	17	0
17	24	2	5	12	18	0
18	23	2	5	12	19	0
19	22	3	5	12	20	0
20	21	2	5	12	21	0
21	28	16	5	12	22	0
22	26	2	9	11	23	29
23	25	2	9	9	24	0
24	24	2	9	9	25	0
25	23	2	9	9	26	0
26	22	3	9	9	27	0
27	21	5	9	9	28	0
28	11	1	9	9	29	0
29	26	3	11	11	30	0
30	25	2	11	11	31	0
31	24	2	11	11	32	0
32	23	2	11	11	33	0
33	22	3	11	11	34	0
34	21	4	11	11	35	0
35	13	23	11	11	36	0

```
TARGET STRING PRODUCED :

W'=(1*COS(X-Y)+X*(-(1-0))*SIN(X-Y))
 INDEPENDENT VARIABLE IS Y DEPENDENT VARIABLE IS W
ANALYSED SUCCESSFULLY AS TYPE         27
TABULAR ANALYSIS RECORD :
```

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	35	2	10
2	26	3	1	2	3	0
3	25	2	1	2	4	0
4	24	2	1	2	5	0
5	23	2	1	2	6	0

6	22	3	1	2	7	0
7	21	6	1	2	8	9
8	12	1	1	1	9	0
9	16	2	2	2	10	0
10	26	3	4	35	11	0
11	25	2	4	35	12	0
12	24	2	4	35	13	0
13	23	2	4	35	14	0
14	22	3	4	35	15	0
15	21	1	4	35	16	0
16	26	1	5	34	17	45
17	25	1	5	14	18	25
18	24	2	5	5	19	0
19	23	2	5	5	20	0
20	22	3	5	5	21	0
21	21	3	5	5	22	0
22	20	3	5	5	23	0
23	18	2	5	5	24	25
24	17	2	5	5	0	0
25	25	2	7	14	26	0
26	24	2	7	14	27	0
27	23	2	7	14	28	0
28	22	3	7	14	29	0
29	21	2	7	14	30	0
30	28	16	7	14	31	0
31	26	2	11	13	32	38
32	25	2	11	11	33	0
33	24	2	11	11	34	0
34	23	2	11	11	35	0
35	22	3	11	11	36	0
36	21	4	11	11	37	0
37	13	23	11	11	38	0
38	26	3	13	13	39	0
39	25	2	13	13	40	0
40	24	2	13	13	41	0
41	23	2	13	13	42	0
42	22	3	13	13	43	0
43	21	5	13	13	44	0
44	11	1	13	13	45	0
45	26	3	16	34	46	0
46	25	1	16	34	47	52
47	24	2	16	16	48	0
48	23	2	16	16	49	0
49	22	3	16	16	50	0
50	21	4	16	16	51	0
51	13	23	16	16	52	0
52	25	1	18	34	53	81
53	24	2	18	25	54	0
54	23	2	18	25	55	0
55	22	3	18	25	56	0
56	21	1	18	25	57	0
57	26	3	19	24	58	0
58	25	2	19	24	59	0
59	24	2	19	24	60	0
60	23	2	19	24	61	0
61	22	2	19	24	62	0
62	21	1	20	24	63	0
63	26	2	21	23	64	72
64	25	2	21	21	65	0
65	24	2	21	21	66	0
66	23	2	21	21	67	0
67	22	3	21	21	68	0
68	21	3	21	21	69	0
69	20	3	21	21	70	0

70	18	2	21	21	71	72
71	17	2	21	21	0	0
72	26	3	23	23	73	0
73	25	2	23	23	74	0
74	24	2	23	23	75	0
75	23	2	23	23	76	0
76	22	3	23	23	77	0
77	21	3	23	23	78	0
78	20	3	23	23	79	0
79	18	2	23	23	80	81
80	17	1	23	23	0	0
81	25	2	27	34	82	0
82	24	2	27	34	83	0
83	23	2	27	34	84	0
84	22	3	27	34	85	0
85	21	2	27	34	86	0
86	28	15	27	34	87	0
87	26	2	31	33	88	94
88	25	2	31	31	89	0
89	24	2	31	31	90	0
90	23	2	31	31	91	0
91	22	3	31	31	92	0
92	21	4	31	31	93	0
93	13	23	31	31	94	0
94	26	3	33	33	95	0
95	25	2	33	33	96	0
96	24	2	33	33	97	0
97	23	2	33	33	98	0
98	22	3	33	33	99	0
99	21	5	33	33	100	0
100	11	1	33	33	101	0

TARGET STRING PRODUCED :

W''=((0*COS(X-Y)+1*(-(0-1))*SIN(X-Y))+(0*(-(1-0))*SIN(X-Y)+X*((-(0-0))*SIN(X-Y)+(-(1-0))*(0-1)*COS(X-Y))))
X = 2.000
Y = 1.000
ANALYSED SUCCESSFULLY AS TYPE 27
TABULAR ANALYSIS RECORD :

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	106	2	11
2	26	3	1	3	3	0
3	25	2	1	3	4	0
4	24	2	1	3	5	0
5	23	2	1	3	6	0
6	22	3	1	3	7	0
7	21	6	1	3	8	9
8	12	1	1	1	9	0
9	16	1	2	3	10	0
10	16	2	3	3	11	0
11	26	3	5	106	12	0
12	25	2	5	106	13	0
13	24	2	5	106	14	0
14	23	2	5	106	15	0
15	22	3	5	106	16	0
16	21	1	5	106	17	0
17	26	1	6	105	18	110
18	25	2	6	37	19	0
19	24	2	6	37	20	0
20	23	2	6	37	21	0
21	22	3	6	37	22	0

22	21	1	6	37	23	0
23	26	1	7	36	24	52
24	25	1	7	16	25	32
25	24	2	7	7	26	0
26	23	2	7	7	27	0
27	22	3	7	7	28	0
28	21	3	7	7	29	0
29	20	3	7	7	30	0
30	18	2	7	7	31	32
31	17	1	7	7	0	0
32	25	2	9	16	33	0
33	24	2	9	16	34	0
34	23	2	9	16	35	0
35	22	3	9	16	36	0
36	21	2	9	16	37	0
37	28	16	9	16	38	0
38	26	2	13	15	39	45
39	25	2	13	13	40	0
40	24	2	13	13	41	0
41	23	2	13	13	42	0
42	22	3	13	13	43	0
43	21	4	13	13	44	0
44	13	23	13	13	45	0
45	26	3	15	15	46	0
46	25	2	15	15	47	0
47	24	2	15	15	48	0
48	23	2	15	15	49	0
49	22	3	15	15	50	0
50	21	5	15	15	51	0
51	11	1	15	15	52	0
52	26	3	18	36	53	0
53	25	1	18	36	54	61
54	24	2	18	18	55	0
55	23	2	18	18	56	0
56	22	3	18	18	57	0
57	21	3	18	18	58	0
58	20	3	18	18	59	0
59	18	2	18	18	60	61
60	17	2	18	18	0	0
61	25	1	20	36	62	90
62	24	2	20	27	63	0
63	23	2	20	27	64	0
64	22	3	20	27	65	0
65	21	1	20	27	66	0
66	26	3	21	26	67	0
67	25	2	21	26	68	0
68	24	2	21	26	69	0
69	23	2	21	26	70	0
70	22	2	21	26	71	0
71	21	1	22	26	72	0
72	26	2	23	25	73	81
73	25	2	23	23	74	0
74	24	2	23	23	75	0
75	23	2	23	23	76	0
76	22	3	23	23	77	0
77	21	3	23	23	78	0
78	20	3	23	23	79	0
79	18	2	23	23	80	81
80	17	1	23	23	0	0
81	26	3	25	25	82	0
82	25	2	25	25	83	0
83	24	2	25	25	84	0
84	23	2	25	25	85	0
85	22	3	25	25	86	0

86	21	3	25	25	87	0
87	20	3	25	25	88	0
88	18	2	25	25	89	90
89	17	2	25	25	0	0
90	25	2	29	36	91	0
91	24	2	29	36	92	0
92	23	2	29	36	93	0
93	22	3	29	36	94	0
94	21	2	29	36	95	0
95	28	15	29	36	96	0
96	26	2	33	35	97	103
97	25	2	33	33	98	0
98	24	2	33	33	99	0
99	23	2	33	33	100	0
100	22	3	33	33	101	0
101	21	4	33	33	102	0
102	13	23	33	33	103	0
103	26	3	35	35	104	0
104	25	2	35	35	105	0
105	24	2	35	35	106	0
106	23	2	35	35	107	0
107	22	3	35	35	108	0
108	21	5	35	35	109	0
109	11	1	35	35	110	0
110	26	3	39	105	111	0
111	25	2	39	105	112	0
112	24	2	39	105	113	0
113	23	2	39	105	114	0
114	22	3	39	105	115	0
115	21	1	39	105	116	0
116	26	1	40	104	117	174
117	25	1	40	58	118	125
118	24	2	40	40	119	0
119	23	2	40	40	120	0
120	22	3	40	40	121	0
121	21	3	40	40	122	0
122	20	3	40	40	123	0
123	18	2	40	40	124	125
124	17	1	40	40	0	0
125	25	1	42	58	126	154
126	24	2	42	49	127	0
127	23	2	42	49	128	0
128	22	3	42	49	129	0
129	21	1	42	49	130	0
130	26	3	43	48	131	0
131	25	2	43	48	132	0
132	24	2	43	48	133	0
133	23	2	43	48	134	0
134	22	2	43	48	135	0
135	21	1	44	48	136	0
136	26	2	45	47	137	145
137	25	2	45	45	138	0
138	24	2	45	45	139	0
139	23	2	45	45	140	0
140	22	3	45	45	141	0
141	21	3	45	45	142	0
142	20	3	45	45	143	0
143	18	2	45	45	144	145
144	17	2	45	45	0	0
145	26	3	47	47	146	0
146	25	2	47	47	147	0
147	24	2	47	47	148	0
148	23	2	47	47	149	0
149	22	3	47	47	150	0

150	21	3	47	47	151	0
151	20	3	47	47	152	0
152	18	2	47	47	153	154
153	17	1	47	47	0	0
154	25	2	51	58	155	0
155	24	2	51	58	156	0
156	23	2	51	58	157	0
157	22	3	51	58	158	0
158	21	2	51	58	159	0
159	28	15	51	58	160	0
160	26	2	55	57	161	167
161	25	2	55	55	162	0
162	24	2	55	55	163	0
163	23	2	55	55	164	0
164	22	3	55	55	165	0
165	21	4	55	55	166	0
166	13	23	55	55	167	0
167	26	3	57	57	168	0
168	25	2	57	57	169	0
169	24	2	57	57	170	0
170	23	2	57	57	171	0
171	22	3	57	57	172	0
172	21	5	57	57	173	0
173	11	1	57	57	174	0
174	26	3	60	104	175	0
175	25	1	60	104	176	181
176	24	2	60	60	177	0
177	23	2	60	60	178	0
178	22	3	60	60	179	0
179	21	4	60	60	180	0
180	13	23	60	60	181	0
181	25	2	62	104	182	0
182	24	2	62	104	183	0
183	23	2	62	104	184	0
184	22	3	62	104	185	0
185	21	1	62	104	186	0
186	26	1	63	103	187	236
187	25	1	63	79	188	216
188	24	2	63	70	189	0
189	23	2	63	70	190	0
190	22	3	63	70	191	0
191	21	1	63	70	192	0
192	26	3	64	69	193	0
193	25	2	64	69	194	0
194	24	2	64	69	195	0
195	23	2	64	69	196	0
196	22	2	64	69	197	0
197	21	1	65	69	198	0
198	26	2	66	68	199	207
199	25	2	66	66	200	0
200	24	2	66	66	201	0
201	23	2	66	66	202	0
202	22	3	66	66	203	0
203	21	3	66	66	204	0
204	20	3	66	66	205	0
205	18	2	66	66	206	207
206	17	1	66	66	0	0
207	26	3	68	68	208	0
208	25	2	68	68	209	0
209	24	2	68	68	210	0
210	23	2	68	68	211	0
211	22	3	68	68	212	0
212	21	3	68	68	213	0
213	20	3	68	68	214	0

214	18	2	68	68	215	216
215	17	1	68	68	0	0
216	25	2	72	79	217	0
217	24	2	72	79	218	0
218	23	2	72	79	219	0
219	22	3	72	79	220	0
220	21	2	72	79	221	0
221	28	15	72	79	222	0
222	26	2	76	78	223	229
223	25	2	76	76	224	0
224	24	2	76	76	225	0
225	23	2	76	76	226	0
226	22	3	76	76	227	0
227	21	4	76	76	228	0
228	13	23	76	76	229	0
229	26	3	78	78	230	0
230	25	2	78	78	231	0
231	24	2	78	78	232	0
232	23	2	78	78	233	0
233	22	3	78	78	234	0
234	21	5	78	78	235	0
235	11	1	78	78	236	0
236	26	3	81	103	237	0
237	25	1	81	103	238	266
238	24	2	81	88	239	0
239	23	2	81	88	240	0
240	22	3	81	88	241	0
241	21	1	81	88	242	0
242	26	3	82	87	243	0
243	25	2	82	87	244	0
244	24	2	82	87	245	0
245	23	2	82	87	246	0
246	22	2	82	87	247	0
247	21	1	83	87	248	0
248	26	2	84	86	249	257
249	25	2	84	84	250	0
250	24	2	84	84	251	0
251	23	2	84	84	252	0
252	22	3	84	84	253	0
253	21	3	84	84	254	0
254	20	3	84	84	255	0
255	18	2	84	84	256	257
256	17	2	84	84	0	0
257	26	3	86	86	258	0
258	25	2	86	86	259	0
259	24	2	86	86	260	0
260	23	2	86	86	261	0
261	22	3	86	86	262	0
262	21	3	86	86	263	0
263	20	3	86	86	264	0
264	18	2	86	86	265	266
265	17	1	86	86	0	0
266	25	1	90	103	267	289
267	24	2	90	94	268	0
268	23	2	90	94	269	0
269	22	3	90	94	270	0
270	21	1	90	94	271	0
271	26	2	91	93	272	280
272	25	2	91	91	273	0
273	24	2	91	91	274	0
274	23	2	91	91	275	0
275	22	3	91	91	276	0
276	21	3	91	91	277	0
277	20	3	91	91	278	0

278	18	2	91	91	279	280
279	17	1	91	91	0	0
280	26	3	93	93	281	0
281	25	2	93	93	282	0
282	24	2	93	93	283	0
283	23	2	93	93	284	0
284	22	3	93	93	285	0
285	21	3	93	93	286	0
286	20	3	93	93	287	0
287	18	2	93	93	288	289
288	17	2	93	93	0	0
289	25	2	96	103	290	0
290	24	2	96	103	291	0
291	23	2	96	103	292	0
292	22	3	96	103	293	0
293	21	2	96	103	294	0
294	28	16	96	103	295	0
295	26	2	100	102	296	302
296	25	2	100	100	297	0
297	24	2	100	100	298	0
298	23	2	100	100	299	0
299	22	3	100	100	300	0
300	21	4	100	100	301	0
301	13	23	100	100	302	0
302	26	3	102	102	303	0
303	25	2	102	102	304	0
304	24	2	102	102	305	0
305	23	2	102	102	306	0
306	22	3	102	102	307	0
307	21	5	102	102	308	0
308	11	1	102	102	309	0

TARGET STRING PRODUCED :
W''= 1.922

NEW SOURCE STATEMENT
TYPE 27
LENGTH 12
HIGHIEST ORDER OF DIFFERENTIATION ACHIEVED 2
CODE FOR EVALUATION IS 1 ELSE 0, CODE= 1

SOURCE STATEMENT :
W=X*COS(X-Y)
INDEPENDENT VARIABLE IS Y DEPENDENT VARIABLE IS W
ANALYSED SUCCESSFULLY AS TYPE 27
TABULAR ANALYSIS RECORD :

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	12	2	9
2	26	3	1	1	3	0
3	25	2	1	1	4	0
4	24	2	1	1	5	0
5	23	2	1	1	6	0
6	22	3	1	1	7	0
7	21	7	1	1	8	9
8	12	1	1	1	9	0
9	26	3	3	12	10	0
10	25	1	3	12	11	16
11	24	2	3	3	12	0
12	23	2	3	3	13	0
13	22	3	3	3	14	0

14	21	4	3	3	15	0
15	13	23	3	3	16	0
16	25	2	5	12	17	0
17	24	2	5	12	18	0
18	23	2	5	12	19	0
19	22	3	5	12	20	0
20	21	2	5	12	21	0
21	28	16	5	12	22	0
22	26	2	9	11	23	29
23	25	2	9	9	24	0
24	24	2	9	9	25	0
25	23	2	9	9	26	0
26	22	3	9	9	27	0
27	21	4	9	9	28	0
28	13	23	9	9	29	0
29	26	3	11	11	30	0
30	25	2	11	11	31	0
31	24	2	11	11	32	0
32	23	2	11	11	33	0
33	22	3	11	11	34	0
34	21	5	11	11	35	0
35	11	1	11	11	36	0

TARGET STRING PRODUCED :

W*=(0*COS(X-Y)+X*(-(0-1))*SIN(X-Y))
 INDEPENDENT VARIABLE IS X DEPENDENT VARIABLE IS W
ANALYSED SUCCESSFULLY AS TYPE 27
TABULAR ANALYSIS RECORD :

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	35	2	10
2	26	3	1	2	3	0
3	25	2	1	2	4	0
4	24	2	1	2	5	0
5	23	2	1	2	6	0
6	22	3	1	2	7	0
7	21	6	1	2	8	9
8	12	1	1	1	9	0
9	16	2	2	2	10	0
10	26	3	4	35	11	0
11	25	2	4	35	12	0
12	24	2	4	35	13	0
13	23	2	4	35	14	0
14	22	3	4	35	15	0
15	21	1	4	35	16	0
16	26	1	5	34	17	45
17	25	1	5	14	18	25
18	24	2	5	5	19	0
19	23	2	5	5	20	0
20	22	3	5	5	21	0
21	21	3	5	5	22	0
22	20	3	5	5	23	0
23	18	2	5	5	24	25
24	17	1	5	5	0	0
25	25	2	7	14	26	0
26	24	2	7	14	27	0
27	23	2	7	14	28	0
28	22	3	7	14	29	0
29	21	2	7	14	30	0
30	28	16	7	14	31	0
31	26	2	11	13	32	38

32	25	2	11	11	33	0
33	24	2	11	11	34	0
34	23	2	11	11	35	0
35	22	3	11	11	36	0
36	21	5	11	11	37	0
37	11	1	11	11	38	0
38	26	3	13	13	39	0
39	25	2	13	13	40	0
40	24	2	13	13	41	0
41	23	2	13	13	42	0
42	22	3	13	13	43	0
43	21	4	13	13	44	0
44	13	23	13	13	45	0
45	26	3	16	34	46	0
46	25	1	16	34	47	52
47	24	2	16	16	48	0
48	23	2	16	16	49	0
49	22	3	16	16	50	0
50	21	5	16	16	51	0
51	11	1	16	16	52	0
52	25	1	18	34	53	81
53	24	2	18	25	54	0
54	23	2	18	25	55	0
55	22	3	18	25	56	0
56	21	1	18	25	57	0
57	26	3	19	24	58	0
58	25	2	19	24	59	0
59	24	2	19	24	60	0
60	23	2	19	24	61	0
61	22	2	19	24	62	0
62	21	1	20	24	63	0
63	26	2	21	23	64	72
64	25	2	21	21	65	0
65	24	2	21	21	66	0
66	23	2	21	21	67	0
67	22	3	21	21	68	0
68	21	3	21	21	69	0
69	20	3	21	21	70	0
70	18	2	21	21	71	72
71	17	1	21	21	0	0
72	26	3	23	23	73	0
73	25	2	23	23	74	0
74	24	2	23	23	75	0
75	23	2	23	23	76	0
76	22	3	23	23	77	0
77	21	3	23	23	78	0
78	20	3	23	23	79	0
79	18	2	23	23	80	81
80	17	2	23	23	0	0
81	25	2	27	34	82	0
82	24	2	27	34	83	0
83	23	2	27	34	84	0
84	22	3	27	34	85	0
85	21	2	27	34	86	0
86	28	15	27	34	87	0
87	26	2	31	33	88	94
88	25	2	31	31	89	0
89	24	2	31	31	90	0
90	23	2	31	31	91	0
91	22	3	31	31	92	0
92	21	5	31	31	93	0
93	11	1	31	31	94	0
94	26	3	33	33	95	0
95	25	2	33	33	96	0

96	24	2	33	33	97	0
97	23	2	33	33	98	0
98	22	3	33	33	99	0
99	21	4	33	33	100	0
100	13	23	33	33	101	0

TARGET STRING PRODUCED :

W''=((0*COS(X-Y)+0*(-(1-0))*SIN(X-Y))+(1*(-(0-1))*SIN(X-Y)+X*((-(0-0))*SIN(X-Y)+(-(0-1))*(1-0)*COS(X-Y))))
X = 2.000
Y = 1.000
ANALYSED SUCCESSFULLY AS TYPE 27
TABULAR ANALYSIS RECORD :

ENTRY	TYPE	CAT	FROM	TO	LINKS	
1	27	1	1	106	2	11
2	26	3	1	3	3	0
3	25	2	1	3	4	0
4	24	2	1	3	5	0
5	23	2	1	3	6	0
6	22	3	1	3	7	0
7	21	6	1	3	8	9
8	12	1	1	1	9	0
9	16	1	2	3	10	0
10	16	2	3	3	11	0
11	26	3	5	106	12	0
12	25	2	5	106	13	0
13	24	2	5	106	14	0
14	23	2	5	106	15	0
15	22	3	5	106	16	0
16	21	1	5	106	17	0
17	26	1	6	105	18	110
18	25	2	6	37	19	0
19	24	2	6	37	20	0
20	23	2	6	37	21	0
21	22	3	6	37	22	0
22	21	1	6	37	23	0
23	26	1	7	36	24	52
24	25	1	7	16	25	32
25	24	2	7	7	26	0
26	23	2	7	7	27	0
27	22	3	7	7	28	0
28	21	3	7	7	29	0
29	20	3	7	7	30	0
30	18	2	7	7	31	32
31	17	1	7	7	0	0
32	25	2	9	16	33	0
33	24	2	9	16	34	0
34	23	2	9	16	35	0
35	22	3	9	16	36	0
36	21	2	9	16	37	0
37	28	16	9	16	38	0
38	26	2	13	15	39	45
39	25	2	13	13	40	0
40	24	2	13	13	41	0
41	23	2	13	13	42	0
42	22	3	13	13	43	0
43	21	5	13	13	44	0
44	11	1	13	13	45	0
45	26	3	15	15	46	0
46	25	2	15	15	47	0
47	24	2	15	15	48	0

48	23	2	15	15	49	0
49	22	3	15	15	50	0
50	21	4	15	15	51	0
51	13	23	15	15	52	0
52	26	3	18	36	53	0
53	25	1	18	36	54	61
54	24	2	18	18	55	0
55	23	2	18	18	56	0
56	22	3	18	18	57	0
57	21	3	18	18	58	0
58	20	3	18	18	59	0
59	18	2	18	18	60	61
60	17	1	18	18	0	0
61	25	1	20	36	62	90
62	24	2	20	27	63	0
63	23	2	20	27	64	0
64	22	3	20	27	65	0
65	21	1	20	27	66	0
66	26	3	21	26	67	0
67	25	2	21	26	68	0
68	24	2	21	26	69	0
69	23	2	21	26	70	0
70	22	2	21	26	71	0
71	21	1	22	26	72	0
72	26	2	23	25	73	81
73	25	2	23	23	74	0
74	24	2	23	23	75	0
75	23	2	23	23	76	0
76	22	3	23	23	77	0
77	21	3	23	23	78	0
78	20	3	23	23	79	0
79	18	2	23	23	80	81
80	17	2	23	23	0	0
81	26	3	25	25	82	0
82	25	2	25	25	83	0
83	24	2	25	25	84	0
84	23	2	25	25	85	0
85	22	3	25	25	86	0
86	21	3	25	25	87	0
87	20	3	25	25	88	0
88	18	2	25	25	89	90
89	17	1	25	25	0	0
90	25	2	29	36	91	0
91	24	2	29	36	92	0
92	23	2	29	36	93	0
93	22	3	29	36	94	0
94	21	2	29	36	95	0
95	28	15	29	36	96	0
96	26	2	33	35	97	103
97	25	2	33	33	98	0
98	24	2	33	33	99	0
99	23	2	33	33	100	0
100	22	3	33	33	101	0
101	21	5	33	33	102	0
102	11	1	33	33	103	0
103	26	3	35	35	104	0
104	25	2	35	35	105	0
105	24	2	35	35	106	0
106	23	2	35	35	107	0
107	22	3	35	35	108	0
108	21	4	35	35	109	0
109	13	23	35	35	110	0
110	26	3	39	105	111	0
111	25	2	39	105	112	0

112	24	2	39	105	113	0
113	23	2	39	105	114	0
114	22	3	39	105	115	0
115	21	1	39	105	116	0
116	26	1	40	104	117	174
117	25	1	40	58	118	125
118	24	2	40	40	119	0
119	23	2	40	40	120	0
120	22	3	40	40	121	0
121	21	3	40	40	122	0
122	20	3	40	40	123	0
123	18	2	40	40	124	125
124	17	2	40	40	0	0
125	25	1	42	58	126	154
126	24	2	42	49	127	0
127	23	2	42	49	128	0
128	22	3	42	49	129	0
129	21	1	42	49	130	0
130	26	3	43	48	131	0
131	25	2	43	48	132	0
132	24	2	43	48	133	0
133	23	2	43	48	134	0
134	22	2	43	48	135	0
135	21	1	44	48	136	0
136	26	2	45	47	137	145
137	25	2	45	45	138	0
138	24	2	45	45	139	0
139	23	2	45	45	140	0
140	22	3	45	45	141	0
141	21	3	45	45	142	0
142	20	3	45	45	143	0
143	18	2	45	45	144	145
144	17	1	45	45	0	0
145	26	3	47	47	146	0
146	25	2	47	47	147	0
147	24	2	47	47	148	0
148	23	2	47	47	149	0
149	22	3	47	47	150	0
150	21	3	47	47	151	0
151	20	3	47	47	152	0
152	18	2	47	47	153	154
153	17	2	47	47	0	0
154	25	2	51	58	155	0
155	24	2	51	58	156	0
156	23	2	51	58	157	0
157	22	3	51	58	158	0
158	21	2	51	58	159	0
159	28	15	51	58	160	0
160	26	2	55	57	161	167
161	25	2	55	55	162	0
162	24	2	55	55	163	0
163	23	2	55	55	164	0
164	22	3	55	55	165	0
165	21	5	55	55	166	0
166	11	1	55	55	167	0
167	26	3	57	57	168	0
168	25	2	57	57	169	0
169	24	2	57	57	170	0
170	23	2	57	57	171	0
171	22	3	57	57	172	0
172	21	4	57	57	173	0
173	13	23	57	57	174	0
174	26	3	60	104	175	0
175	25	1	60	104	176	181

176	24	2	60	60	177	0
177	23	2	60	60	178	0
178	22	3	60	60	179	0
179	21	5	60	60	180	0
180	11	1	60	60	181	0
181	25	2	62	104	182	0
182	24	2	62	104	183	0
183	23	2	62	104	184	0
184	22	3	62	104	185	0
185	21	1	62	104	186	0
186	26	1	63	103	187	236
187	25	1	63	79	188	216
188	24	2	63	70	189	0
189	23	2	63	70	190	0
190	22	3	63	70	191	0
191	21	1	63	70	192	0
192	26	3	64	69	193	0
193	25	2	64	69	194	0
194	24	2	64	69	195	0
195	23	2	64	69	196	0
196	22	2	64	69	197	0
197	21	1	65	69	198	0
198	26	2	66	68	199	207
199	25	2	66	66	200	0
200	24	2	66	66	201	0
201	23	2	66	66	202	0
202	22	3	66	66	203	0
203	21	3	66	66	204	0
204	20	3	66	66	205	0
205	18	2	66	66	206	207
206	17	1	66	66	0	0
207	26	3	68	68	208	0
208	25	2	68	68	209	0
209	24	2	68	68	210	0
210	23	2	68	68	211	0
211	22	3	68	68	212	0
212	21	3	68	68	213	0
213	20	3	68	68	214	0
214	18	2	68	68	215	216
215	17	1	68	68	0	0
216	25	2	72	79	217	0
217	24	2	72	79	218	0
218	23	2	72	79	219	0
219	22	3	72	79	220	0
220	21	2	72	79	221	0
221	28	15	72	79	222	0
222	26	2	76	78	223	229
223	25	2	76	76	224	0
224	24	2	76	76	225	0
225	23	2	76	76	226	0
226	22	3	76	76	227	0
227	21	5	76	76	228	0
228	11	1	76	76	229	0
229	26	3	78	78	230	0
230	25	2	78	78	231	0
231	24	2	78	78	232	0
232	23	2	78	78	233	0
233	22	3	78	78	234	0
234	21	4	78	78	235	0
235	13	23	78	78	236	0
236	26	3	81	103	237	0
237	25	1	81	103	238	266
238	24	2	81	88	239	0
239	23	2	81	88	240	0

240	22	3	81	88	241	0
241	21	1	81	88	242	0
242	26	3	82	87	243	0
243	25	2	82	87	244	0
244	24	2	82	87	245	0
245	23	2	82	87	246	0
246	22	2	82	87	247	0
247	21	1	83	87	248	0
248	26	2	84	86	249	257
249	25	2	84	84	250	0
250	24	2	84	84	251	0
251	23	2	84	84	252	0
252	22	3	84	84	253	0
253	21	3	84	84	254	0
254	20	3	84	84	255	0
255	18	2	84	84	256	257
256	17	1	84	84	0	0
257	26	3	86	86	258	0
258	25	2	86	86	259	0
259	24	2	86	86	260	0
260	23	2	86	86	261	0
261	22	3	86	86	262	0
262	21	3	86	86	263	0
263	20	3	86	86	264	0
264	18	2	86	86	265	266
265	17	2	86	86	0	0
266	25	1	90	103	267	289
267	24	2	90	94	268	0
268	23	2	90	94	269	0
269	22	3	90	94	270	0
270	21	1	90	94	271	0
271	26	2	91	93	272	280
272	25	2	91	91	273	0
273	24	2	91	91	274	0
274	23	2	91	91	275	0
275	22	3	91	91	276	0
276	21	3	91	91	277	0
277	20	3	91	91	278	0
278	18	2	91	91	279	280
279	17	2	91	91	0	0
280	26	3	93	93	281	0
281	25	2	93	93	282	0
282	24	2	93	93	283	0
283	23	2	93	93	284	0
284	22	3	93	93	285	0
285	21	3	93	93	286	0
286	20	3	93	93	287	0
287	18	2	93	93	288	289
288	17	1	93	93	0	0
289	25	2	96	103	290	0
290	24	2	96	103	291	0
291	23	2	96	103	292	0
292	22	3	96	103	293	0
293	21	2	96	103	294	0
294	28	16	96	103	295	0
295	26	2	100	102	296	302
296	25	2	100	100	297	0
297	24	2	100	100	298	0
298	23	2	100	100	299	0
299	22	3	100	100	300	0
300	21	5	100	100	301	0
301	11	1	100	100	302	0
302	26	3	102	102	303	0
303	25	2	102	102	304	0

304	24	2	102	102	305	0
305	23	2	102	102	306	0
306	22	3	102	102	307	0
307	21	4	102	102	308	0
308	13	23	102	102	309	0

TARGET STRING PRODUCED :
W''= 1.922

****END OUTPUT THIS DOCUMENT

```
END OF RUN;
****END OUTPUT FOR THIS SET OF DOCUMENTS
```

APPENDIX E

COMPLETE SYNTAX AND SEMANTICS LISTING FOR DIFFERENTIATION

```
09
<>()|E;&.+=QU,WKLCPSNAZID
500;2;400;400;
+++
DIFFERENTIATION OF ALGEBRIC EXPRESSION;
28,450;1200,1;
14   <ALPHABETS>     =A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z;
16   <QUOTE>         ='(<16>|<E>);
17   <DIGITS>        =0|1|2|3|4|5|6|7|8|9;
18   <INTEGERS>      =<17>(<18>|<E>);
19   <FRACTION>      =<17>(<19>|<E>);
20   <UNSIGNED DECIMAL NUMBER> =<18>.<19>|.<19>|<18>;
21   <PRIMARY>       =<(><26><)>|<28>|<20>|<13>|<11>|<12>(<16>|<E>);
22   <SIGNED PRIMARY>=(+|-|<E>)<21>;
23   <FACTOR>        =<22>(@<23>|<E>);
24   <TERM1>         =<23>(/<24>|<E>);
25   <TERM2>         =<24>(*<25>|<E>);
26   <S.A.E.>        =<25>(+<26>|-<26>|<E>);
27   <STATEMENT>     =<26>(<=><26>|<E>);
28   <FUNCTION>      =(EXP|LN|SINH|COSH|TANH|COTH|SECH|CSCH|
                      ARCSIN|ARCCOS|ARCTAN|ARCCOT|ARCSEC|ARCCSC|
                      SIN|COS|TAN|COT|SEC|CSC|
                      ARSINH|ARCOSH|ARTANH|ARCOTH|ARSECH|ARCSCH)<(><26><)>;
0;
1;
11   <INDEPENDENT VARIABLE>=W<1>;
12   <DEPENDENT VARIABLE>=PW<'>;
13   <CONSTANT>      =W<0>;
16   <QUOTE>         =P;
20   <UNSIGNED DECIMAL NUMBER>=W<0>;
21   <PRIMARY>       =K<1>W<(>C<1>W<)>.
                     K<2,3,4,5,6,7>C<1>.
                      K<6>C<2>.;
22   <SIGNED PRIMARY>=K<2>W<->.
                       K<1,2,3>C<1>.;
23   <FACTOR>        =K<1>L<1>W<@>L<2>W<*((>L<2>W<*>C<1>W<)/>L<1>W<+>C<2>
                      W<*LN(>L<1>W<))>&K<2>C<1>.;
24   <TERM1>        =K<1>W<(>C<1>W<*>L<2>W<->L<1>W<*>C<2>W<)/>L<2>W<@2>&
                     K<2>C<1>.;
25   <TERM2>         =K<1>W<(>C<1>W<*>L<2>W<+>L<1>W<*>C<2>W<)>&
                     K<2>C<1>.;
26   <S.A.E.>      =K<1,2,3>C<1>.
                    K<1>W<+>.
                    K<2>W<->.
                    K<1,2>C<2>.;
27   <STATEMENT>     =C<1>K<1>W<=>C<2>.;
28   <FUNCTION>      =K<1>W<(>C<1>W<)*EXP(>L<1>W<)>&
                      K<2>W<(>C<1>W<)/(>L<1>W<)>&
                       K<3>W<(>C<1>W<)*COSH(>L<1>W<)>&
                       K<4>W<(>C<1>W<)*SINH(>L<1>W<)>&
                       K<5>W<(>C<1>W<)*(SECH(>L<1>W<))@2>&
                       K<6>W<(-(>C<1>W<))*(CSCH(>L<1>W<))@2>&
                       K<7>W<(-(>C<1>W<))*SECH(>L<1>W<)*TANH(>L<1>W<)>&
                       K<8>W<(-(>C<1>W<))*CSCH(>L<1>W<)*COTH(>L<1>W<)>&
                      K<9>W<(>C<1>W<)/(1-(>L<1>W<)@2)@(1/2)>&
                      K<10>W<(-(>C<1>W<))/(1-(>L<1>W<)@2)@(1/2)>&
                      K<11>W<(>C<1>W<)/(1+(>L<1>W<)@2)>&
                      K<12>W<(-(>C<1>W<))/(1+(>L<1>W<)@2)>&
                      K<13>W<(>C<1>W<)/((>L<1>W<)*((>L<1>W<)@2-1)@(1/2))>&
```

```
                    K<14>W<(-(>C<1>W<))/((>L<1>W<)*((>L<1>W<)@2-1)@(1/2))>&
                   K<15>W<(>C<1>W<)*COS(>L<1>W<)>&
                   K<16>W<(-(>C<1>W<))*SIN(>L<1>W<)>&
                   K<17>W<(>C<1>W<)*(SEC(>L<1>W<))@2>&
                   K<18>W<(-(>C<1>W<))*((CSC(>L<1>W<))@2>&
                   K<19>W<(>C<1>W<)*SEC(>L<1>W<)*TAN(>L<1>W<)>&
                   K<20>W<(-(>C<1>W<))*CSC(>L<1>W<)*COT(>L<1>W<)>&
                    K<21>W<(>C<1>W<)/(1+(>L<1>W<)@2)@(1/2)>&
                    K<22>W<(>C<1>W<)/((>L<1>W<)@2-1)@(1/2)>&
                    K<23>W<(>C<1>W<)/(1-(>L<1>W<)@2)>&
                    K<24>W<(>C<1>W<)/(1-(>L<1>W<)@2)>&
                    K<25>W<(-(>C<1>W<))/((>L<1>W<)*(1-(>L<1>W<)@2)@(1/2))>&
                    K<26>W<(-(>C<1>W<))/((>L<1>W<)*(1+(>L<1>W<)@2)@(1/2))>.;
0;
0;
27;2;1;
W=X*COS(X-Y);
INDEPENDENT VARIABLE IS &X AND DEPENDENT VARIABLE IS &W
INDEPENDENT VARIABLE IS &Y AND DEPENDENT VARIABLE IS &W
&X=2.00;  &Y=1.00;#
27;2;1;
W=X*COS(X-Y);
INDEPENDENT VARIABLE IS &Y AND DEPENDENT VARIABLE IS &W
INDEPENDENT VARIABLE IS &X AND DEPENDENT VARIABLE IS &W
&X=2.00;  &Y=1.00;#
0;0;0;
+++
+++
END OF RUN;
0;
```

APPENDIX F

LISTINGS OF SIMPLIFIED EXPRESSION

The numeric results of the partial derivatives on Appendix D are done directly on the non-simplified resulting symbolic expressions. If simplified expressions are desired the following two approaches can be used.

1. The resulting non-simplified expressions are passed alternatively with SIM 1 and SIM 2 until the simplest expressions are obtained.
2. After each differentiation, the resulting expressions are simplified first with SIM 1 and SIM 2 before the next differentiation takes place, the results of which are then simplified similarly.

Both approaches are tried in the case of (a) $\frac{\partial^2 x \cos(x-y)}{\partial y \partial x}$ and (b) $\frac{^2x \cos(x-y)}{\partial x \partial y}$

I. First Approach.

solutions for (a) $\frac{\partial^2 x \cos(x-y)}{\partial y \partial x}$.

(i) Original expression

```
W = X*COS(X-Y);
```

(ii) After first differentiation with respect to X.

```
W' = (1*COS(X-Y)+X*(-(1-0))*SIN(X-Y));
```

(iii) After elimination of 0 and 1 (i.e. pass through SIM 1)

W' = (COS (X-Y)+X*(-SIN(X-Y)));

(iv) After elimination of brackets (i.e. pass through SIM 2)

W' = (COS (X-Y)+X*(-SIN(X-Y)));

(v) After second differentiation with respect to Y.

W'' = ((-(0-1))*SIN(X-Y)+(0*(-SIN(X-Y))
+X*(-(0-1)*COS(X-Y))));

(vi) After second pass through SIM 1

W'' = (SIN(X-Y)+(X*(-(-COS(X-Y)))));

(vii) After second pass through SIM 2

W'' = (SIN(X-Y)+(X*(-(-COS(X-Y)))));

Corresponding solutions for (b) $\frac{\partial^2 x \cos(x-y)}{\partial x \partial y}$:

(i) W = X * COS (X-Y);

(ii) After first differentiation with respect to Y.

W' = (0* COS (X-Y)+ X*(-(0-1))*SIN(X-Y));

(iii) W' = (X * SIN (X-Y));

(iv) W' = (X * SIN (X-Y));

(v) After second differentiation with respect to X.

W'' = (1* SIN (X-Y) + X*(1-0)* COS(X-Y)));

(vi) W'' = ((SIN (X-Y) + X* COS (X-Y)));

(vii) W'' = (SIN (X-Y) + X* COS (X-Y));

II. Second Appraoch.

solution for (a) $\frac{\partial^2 x \cos (x-y)}{\partial y \, \partial x}$:

(i) Original expression

W = X * COS (X-Y) ;

(ii) After two differentiations

W'' = ((0*COS(X-Y)+1*(-(0-1))*SIN(X-Y))
+(0*(-(1-0))*SIN(X-Y)+X*((-(0-0))
SIN(X-Y)+(-(1-0))(0-1)*COS(X-Y))));

(iii) After first pass through SIM 1

W'' = ((SIN(X-Y))+(X*((-0)*SIN(X-Y)
+(-(-COS(X-Y))))));

(iv) After first pass through SIM 2

W'' = (SIN(X-Y)+(X*((-0)*SIN(X-Y)
+(-(-COS(X-Y))))));

(v) After second pass through SIM 1

W'' = (SIN(X-Y)+(X*((-(-COS(X-Y)))))));

(vi) After second pass through SIM 2

W'' = (SIN(X-Y)+(X*(-(-COS(X-Y))))));

Corresponding solutions for (b) $\frac{\partial^2 x \cos(x-y)}{\partial x \partial y}$

(i) `W = X* COS(X-Y);`

(ii)
```
W'' = ((0* COS(X-Y)+0*(-(1-0))*
      SIN(X-Y))+(1*(-(0-1))*SIN(X-Y)+
      X*((-(0-0))*SIN(X-Y)+(-(0-1))*
      (1-0)*COS(X-Y))));
```

(iii)
```
W'' = ((SIN(X-Y)+X*((-0)*SIN(X-Y)
      +COS(X-Y))));
```

(iv)
```
W'' = (SIN(X-Y)+X*((-0)&SIN(X-Y)
      +COS(X-Y)));
```

(v) `W'' = (SIN(X-Y)+X*COS(X-Y)));`

(vi) `W'' = (SIN(X-Y)+X*COS(X-Y));`

REFERENCES

1. Backus, J.W., The syntax and semantics of the proposed international language of the Zurich ACM-GAMM conference, Proc. Internal Conf. Information Processing, UNESCO, Paris, France, June 1959, pp. 125-132.

2. Bar-Hillel, Y., Perles, M., and Shamir, E., On formal properties of simple phrase structure grammars. In Readings in Mathematical Psychology vol. 2 (Ed. L. Duncan et al) John Wiley and Sons, Inc. N.Y. London 1965. pp. 75-104.

3. Barnett, M.P., and Futrelle, R.P., Syntactic analysis by digital computer. Comm. ACM 5 (Oct. 1962), 515-526.

4. Brooker, R.A., Top-to-Bottom Parsing Rehabilitated? Comm. ACM 10 (April 1967), 223-224

5. Brooker, R.A., and Morris, D., The compiler-compiler. Annual Review in Automatic Programming, vol. 3, 1963, p. 229.

6. Brown, W.S., The ALPAK system for non-numerical algebra on a digital computer - I : polynomials in several variables and truncated power series with polynomial coefficients. Bell System Tech. J. 42, (1963).

7. Brown, W.S., Hyde, J.P., and Tague, B.A., The ALPAK system for non-numerical algebra on a digital computer - II : rational function of several variables and truncated power series with rational-function coefficients. Bell System Tech. J. 43, No. 2 (1964).

8. Cantor, D.G., On the ambiguity problem of Backus Systems. J. ACM 9 (Oct. 1962), 477-479.

9. Cheatham, T.E. and Sattley, K. Syntax directed compiling. Proc. AFIPS 1964 SJCC, Vol. 25, pp. 31-57.

10. Chomsky N., Formal properties of grammars. In Handbook of Mathematical Psychology, Vol. 2, R.D. Luce, R.R. Bush and E.H. Galanter (Eds.), John Wiley & Sons, Inc., 1963, pp. 323-418.

11. _____., On certain formal properties of grammars. Inform. Contr. 2 (1959), 137-167.

12. Cohen, Doron J., and Gotlieb, C.C., List Structure Form of Grammars for Syntactic Analysis. Comm. Surveys 2 (Mar. 1970), 65-82.

13. Eickel, J., Paul, M., Baur, F.L., and Samelson, K., A syntax controlled generator of formal language processors. Comm. ACM 6 (Aug. 1963) 451-455.

14. Ershov, G.P., Programming Program for the BESM Computer. Translated from the Russian by M. Nadler and edited by J.P. Cleave. Pergamon Press, London - New York - Paris (1959).

15. Feldman, J., and Gries, D., Translator writing systems. Comm. ACM 11 (Feb. 1968), 77-113.

16. Floyd, R.W., The syntax of programming languages - a survey. IEEE Trans. EC 13, (Aug. 1964), 346-353.

17. _____., Syntactic analysis and operator precedence. J. ACM 10 (July 1963), 316-333.

18. _____., On ambiguity in phrase structure languages. Comm. ACM 5 (Oct. 1962), 526, 534.

19. _____., On the nonexistence of a phrase structure grammar for ALGOL - 60. Comm. ACM 5 (Sept. 1962), 483-484.

20. Foxley E., and King P., The implementation of syntax analysis using ALGOL, and some mathematical applications. Comp. J. 10 (Feb. 1968), 325-335.

21. Ginsburg, S., The Mathematical Theory of Context Free Languages. McGraw-Hill, Inc., New York, 1966.

22. Griffiths, T.V., and Petrick, S.R., On the relative efficiencies of context-free grammar recognizers. Comm. ACM 8 (May 1965), 289-299.

23. Hanson, J.W., Caviness, J.S., and Joseph, C., Analytic differentiation by computer. Comm. ACM 5 (June 1962), 349-355.

24. Hopcroft, J.E., and Ullman, J.D., Formal languages and their relation to Automata. Addision-Wesley, Reading, Mass., 1969.

25. Hyde, J.P., The ALPAK system for non-numerical algebra on a digital computer - III : systems of linear equations and class of side relations. Bell System Tech. J. 43, No. 4, Pt. 2 (1964).

26. Irons, E.T., A syntax directed compiler for ALGOL 60. Comm. ACM 4 (Jan. 1961), 51-55.

27. _____., The structure and use of the syntax-directed compiler. Annual Review in Automatic Programming, Vol. 3, 1963, pp. 207-227.

28. _____., Structural connections in formal languages. Comm. ACM 7 (Feb. 1964), 67-71.

29. Kahrimanian, H.G., Analytical differentiation by a digital computer. M.A. thesis, Temple Univ., Philadelphia, Pennsylvania, (May 1953).

30. King, P.R., A top-down syntax oriented translator (Unpublished doctoral dissertation, University of Nottingham, England, May 1969).

31. Ledley, R., and Wilson, J., Automatic-programming language translation through syntactic analysis. Comm. ACM 5 (Mar. 62), 145.

32. McKeeman, W.M., An approach to computer language design. Tech. Rpt. CS 48, Computer Science Dept., Standford U., Standford, Calif., Aug. 1966.

33. Naur, P., (ed.) Report on the algorithmic language ALGOL 60. Numer. Math. 2 (1960), 106-136; Comm. ACM 3 (May 1960), 299-314.

34. Naur, P., et al., Revised report on the algorithmic language ALGOL 60. Comm. ACM 6 (Jan. 1963), 1-17; Numer. Math. 4 (1963), 420-452; Comp. J. 5(1963), 349-367.

35. Nolan, J.F., Analytic differentiation on a digital computer. M.A. thesis, Mass. Inst. Technology, Cambridge, Massachusetts, (May 25, 1953).

36. Sammet J.E., Formula Manipulation by Computer. In Advances in Computers Vol. 8, edited by F.L. Alt and M. Rubinoff, Academic Press, New York - London, 1967, pp. 47-102.

37. _____., An annotated descriptor based bibliography on the use of computers for non-numerical mathematics. Computing Rev. 7 (1966).

38. _____., An overall view of FORMAC, IBM, Systems Development Div., Tech. Rept. TR 00 . 1367, Dec., 1965.

39. Schorr, H., Analytic differentiation using a syntax-directed compiler. Comput. J. 7 (Jan.1965), 290-298.

40. Selby, S.M., and Girling, B., (ed.) CRC Standard Mathematical Tables. 14 edition. The Chemical Rubber Co., Cleveland, Ohio, 1967.

41. Unger, S.H., A global parser for context-free phrase structure grammars. Comm. ACM 11 (April 1968), 240-247.

42. Weissman, C., LISP 1.5 primer. Dickenson Publishing Company Inc., Belmont, California, 1967.

43. Wirth, N., and Weber, H. EULER- a generalization of ALGOL, and its formal definition : Part I, Part II Comm. ACM 9 (Jan., Feb. 1966), 13-25, 89-99.