VIII. On the terminal symbols.

"T defines the same reference language \{9.4\} and the same standard environment \{10\} as D."

By translating the terminal symbols into Chinese the mnemonic character of these symbols were taken into account. It was somewhat difficult to translate the 'bold to symbol', which is used in the revised report both in the to-part of a loop clause and in go-to-option of a strong-MOID-NEST-Jump. There is no Chinese word having the meaning of both. Hence in the Chinese version there are two terminal symbols corresponding to 'bold to symbol', one of which is 'bold end value symbol' (used in loop-clauses), the other is 'bold to symbol' (used in go-to-option). They have the same representation.

IX. On metaproductions.

"Additional means for the creation of extra metaproduction rules".

A new metaproduction rule for APPENDIXETY is introduced, while some other metaproductions are modified. In fact, we know this already from the discussion above.

X. ALGOL 68 and Algebraic Manipulation.

D. C. Ince.
The Open University, Milton Keynes, U.K.

The decade has seen research into algebraic manipulation by computer progress to the point where the majority of computer users have access to at least one algebraic manipulation system. The growth of these systems has not however been accompanied by a corresponding growth of reports of their applications. A number of reasons have been put forward by users and potential users to explain this disparity, of which the two that occur the most are:

1) The user interface for a number of present systems is poor, manipulations on algebraic expressions being expressed in a non-natural way, usually by means of a series of subroutine calls, or in a language which, although well suited to the construction of such systems (eg LISP), tends to be alien to the programming experience of the average scientist or engineer that make up the potential user community. ABC ALGOL and its derivatives (1) and SAC (2) are examples of systems to which this criticism can be applied, while even (3), the only previous attempt to write an algebraic manipulation system in Algol 68 could have this criticism levelled at it.

2) The majority of users do not wish to perform algebraic manipulations in isolation, the manipulations being just one stage in what may be a large program in which numerical or even other symbolic processes may take a dominating part. A number of systems, CAMAL (4) and REDUCE (6) for example, while having excellent facilities for algebraic
manipulation tend to fall short with respect to the numerical and data structuring facilities required and even fall short with respect to the control facilities necessary for efficient programming.

There does however exist in Algol 68 a number of features which not only make it eminently suitable for constructing algebraic manipulation systems but also lead to the construction of systems to which the above two objections do not apply. These features are:

1) **Operator declarations**

The availability of operator declaration in Algol 68 means that a user can express his manipulations in a manner near to his normal working notation. It should be possible in an Algol 68 based system to write

\[
a := (1 + x) \times (1 + y - x)
\]

assuming declarations of +, -, \times on suitable data structures, rather than a series of subroutine calls, or worse still as a LISP expression. This facility alone removes the first objection outlined previously.

2) **List processing facility**

A property of the problems that are capable of solution by an algebraic manipulation system is that the user is unable to specify in advance the amount of storage required for the algebraic expressions that are created. It is for this reason that almost all present systems are based on list structures and languages that are particularly good in handling lists. Although Algol 68 falls short, in terms of list processing facilities, of those languages that have been used to construct algebraic manipulation systems it does however provide enough facilities for the construction of such systems.

**Additional numeric modes**

One of the alarming properties of the problems that have been solved by algebraic manipulation systems is the speed at which numeric coefficients expand and overflow the exact arithmetic capability of the host computer. An example of such a problem is the computation of the \( f \) and \( g \) series (5) of celestial mechanics which overflows a 48 bit integer after 15 iterations. One solution adopted by a number of systems is to convert to floating point when overflow has occurred. This solution is not ideal by any means as often the user of an algebraic manipulation system is interested in the patterns generated by his programs, thus it would be a lot more difficult for a user to discern a pattern in

\[
.400x + .429x^2 + .444x^3 + .455x^4
\]

rather than

\[
2/5x + 3/7x^2 + 4/9x^3 + 5/11x^4
\]

An alternative solution to the problem of integer overflow is to arrange that numeric coefficients can be represented in a number of alternative integer forms of increasing maximum magnitude, thus a language such as Algol 68, which has the capability for additional numeric modes as well as the ability to switch between these modes via unions, would seem to be the answer.
An algebraic manipulation system, written in Algol 68R, has been developed to handle polynomials in arbitrarily many indeterminates with real or long int coefficients. It consists of a series of operators and procedures which operate on expressions defined by

```plaintext
mode expression = ref term;
mode term = struct (ref[] element factor, coeff coeff,
                   int order, ref term next term);
mode coeff = union (real, rational);
mode rational = struct (long int num, denom);
mode element = struct (int atom, power)
```

Where `term` describes a term in a polynomial and `coeff` describes the numeric coefficients in each term. The system was written with the aim of making the user interface compatible not only with the normal algebraic notation familiar to its potential users but with the numeric facilities available in Algol 68.

The system consists of a series of operators and procedures which can be separated into a number of categories:

1) The common algebraic operations defined on polynomials
2) Procedures for the input and display of polynomials
3) Procedures for the integration and differentiation of polynomials
4) Procedures for the numeric evaluation of polynomials and symbolic substitution in polynomials
5) Procedures for the selection of polynomial terms based on a variety of criteria.

In order to illustrate the system I shall take two problems and display the programs needed to solve them.

1) The generation of Legendre polynomials using the relation

\[ P_n(x) = \frac{(2n - 1)xP_{n-1}(x) - (n - 1)P_{n-2}(x)}{n} \]

where \( P_0(x) = 1 \) and \( P_1(x) = x \)

and \( P_n(x) \) is the \( n \)th Legendre Polynomial

```plaintext
'BEGIN'
'EXPRESS' X = GENEXP("X");
[0:20] 'EXPRESS' P;
P[0]:= 'EXPRN' 1;
P[1]:= +X;
PRINT ("P[0]=");
EXPOUT (P[0]);
PRINT ("PI=1=");
EXPOUT (P[1]);
'FOR' I 'FROM' 2 'TO' 20 'DO'
'BEGIN'
P[I]:=((2*I-1)*X*P[I-1]-(I-1)*P[I-2])/I;
PRINT ("P[,,, I, " =")
EXPOUT (P[I])
'END'
'END'
'FINISH,'
2) The computation of the f and g series in celestial mechanics

The f and g series are given by

\[ f_n = u \frac{\partial f}{\partial a} a_{n-1} + v \frac{\partial f}{\partial b} b_{n-1} + w \frac{\partial f}{\partial c} c_{n-1} - a_{n-1} \]
\[ g_n = u \frac{\partial g}{\partial a} a_{n-1} + v \frac{\partial g}{\partial b} b_{n-1} + w \frac{\partial g}{\partial c} c_{n-1} + f_{n-1} \]

where \( u = -3ab, \ v = c - 2b, \ w = -ab - 2bc \) and \( f_0 = 1 \)

and \( g_0 = 0 \)

The program for this computation is

```
'BEGIN' 'C' CALCULATION OF THE FIRST 'C'
'O:20' 'EXPRESSION' F, G;
'EXPRESSION' A = GENEXP("A"),
'B' = GENEXP("B"),
'C' = GENEXP("C"),
'U' = GENEXP("-3#A*B"),
'V' = GENEXP("C-2#B"),
'W' = GENEXP("-B*(A+2*C)");
F[O] = 'EXPRN' 1;
G[O] = 'EXPRN' 0;
PRINT ("F[O]="); EXPOUT (F[O]);
PRINT ("G[O]="); EXPOUT (G[O]);
'FUN' I 'TO' 20 'DO'
  'BEGIN'
    F[I] = DIFF(F[I-1], A)*U + DIFF(F[I-1], B)*V + DIFF(F[I-1], C)*W
      - G[I-1]*A;
    G[I] = DIFF(G[I-1], A)*U + DIFF(G[I-1], B)*V + DIFF(G[I-1], C)*W
      + F[I-1];
  PRINT ("F[", I, "]="); EXPOUT (F[I]);
  PRINT ("G[", I, "]="); EXPOUT (G[I]);
'END'
'END'
'FINISH'
```

1. Riet R P Van de
   'ABC Algol: A language for formula manipulation systems - Part 1 The language'
   Mathematical Centre Tracts No 46, 1975.

2. Collins G E
   'The SAC1 system for algebraic manipulation'
   Proc. 2nd symposium on symbolic and algebraic manipulation 1971.

3. Danicic I, Long F W
   'Algebraic manipulation of polynomials in several indeterminates'

4. Barton D, Bourne S R, Fitch J P
   'An algebra system'

   'Some applications of the Cambridge Algebra System'

6. Hearn A C
   'REDUCE 2 users manual'
   U of Utah, Salt Lake City, Utah. 2nd edn 1973.