KENT ON-LINE SYSTEM

Document: KUSE/ALGEBRA/2

The ALGEBRA sub-system

P. J. Brown
University of Kent at Canterbury
December 1970

Introduction

The ALGEBRA sub-system allows the user to declare a set of objects and then to define a number of operators that can be applied to these objects. The objects are called values. Once the operators and values have been defined the user can investigate their properties by evaluating expressions involving variables, operators and values.

The following is an introductory example, where there are two values, TRUE and FALSE, and the operators are the well-known Boolean operators "implies" and "not". The underlined parts are typed by the computer and the remainder by the console user. A commentary appears to the right of the example.

Introductory example

```
& ENTER ALGEBRA
```
Bring ALGEBRA into action.

```
*** ALGEBRA SYSTEM (VERSION...) READY

VALUES=   TRUE FALSE
```
Declare values.

```
: OPERATOR -
```
Define an operator. This is the operator "not", which is represented by a minus sign.

```
UNARY OR BINARY= UNARY

PRECEDENCE= 100
```
See below for meaning of this.

```
-TRUE=  FALSE

-FALSE= TRUE
```
Define meaning of "not" for all possible values.

```
: OPERATOR HOOK
```
Define another operator. This is the operator "implies", which is represented by the symbol "HOOK".

```
UNARY OR BINARY= BINARY

PRECEDENCE= 50
```
Precedence has been specified as less than that of "not". This means that if "implies" and "not" are used in the same expression, then "not" is done first, i.e.
-A HOOK B
is taken as
(-A) HOOK B
and not as
-(A HOOK B)

```
TRUE HOOK TRUE  = TRUE
TRUE HOOK FALSE = FALSE
FALSE HOOK TRUE = TRUE
FALSE HOOK FALSE= TRUE
```

Now that the operators and values have been defined, it is possible to evaluate expressions involving them. Any symbol in an expression that has not been defined as a value or an operator is taken as a variable and is enumerated for all possible values. The TABLE statement illustrates this.

```
: TABLE A HOOK - (A HOOK B)

A     B     : VALUE
TRUE  TRUE  : FALSE
TRUE  FALSE : TRUE
FALSE TRUE  : TRUE
FALSE FALSE : TRUE
```

This entire table is typed out by the computer. The last column is the value of the expression for the given values of the variables A and B.

```
: TABLE -(P HOOK Q) HOOK(P HOOK -Q)

P     Q     : VALUE
TRUE  TRUE  : TRUE
TRUE  FALSE : TRUE
FALSE TRUE  : TRUE
FALSE FALSE : TRUE
```

Table typed by the computer.

The TRY statement is an abbreviated form of the TABLE statement, e.g.

```
: TRY A HOOK -(A HOOK B)
IS A CONTINGENCY
```

This means that the value of the expression depends on the values of its variables.

```
: TRY -(P HOOK Q) HOOK (P HOOK -Q)
= TRUE
```

This means that the expression has the same value, TRUE, for all values of its variables.

```
:.
```

This means the end of console input.

```
&EXIT
```

This means finish. At this stage "CONT" would have meant continue.

```
*** EXIT FROM ALGEBRA SYSTEM
&
```

Ready for next user.

Basic Units

The above example should have given the user enough knowledge to use the system for himself, and this he is recommended to do.

The rest of this manual explains more exactly the concepts that have been illustrated by example. Firstly the fundamental units with which ALGEBRA deals, namely symbols, expressions and statements, will be described.

Symbols

A symbol is used to represent the name of a value, operator or variable. A symbol must be either

(a) a name symbol, which is a sequence of one or more letters or digits. The sequence may be arbitrarily long and all characters are significant, but symbols will be truncated to six characters if they appear in tables.

or (b) a punctuation symbol, which is a single character that is not any of the following: a letter, a digit, a comma, a tab, a space, a semi-colon, a left parenthesis, a right parenthesis or the newline character. (In addition, when input is from a console, the characters "!" and "←" should be added to this list, since these have special meanings in KOS.)

For example the following are legitimate symbols:

ABLE,A,1,12A3,+,/,.,VERYLONGNAME

and the following are not:

A B,P.J.B.,(,A+

Expressions

An expression is a series of operators with values or variables as operands. Any symbol that is not the name of an existing operator or value can be used as the name of a variable.

(To be precise the syntax of an expression is, in Backus Normal Form:

<expression>::= <expression> <binary operator> <expression> | <unary operator> <expression> | (<expression>) | <variable> | <value

where <unary operator> etc. are all symbols.)

During the evaluation of an expression, operators of highest precedence are performed first. Subject to this, operators are evaluated from left to right. Parentheses can be used to override precedence and redundant pairs of parentheses are permissible.

Thus if "+" and "*" are binary operators and "*" has the higher precedence then:

A+B*C is taken as A+(B*C)
A+B+C is taken as (A+B)+C
A*B+C*D is taken as (A*B)+(C*D)
A*(B+C)*D is taken as (A*(B+C))*D

The choice of precedence for operators is a matter of convention and no fixed rules can be stated save that unary operators should normally be given higher precedence than binary operators.

Statements

Each line of data fed to ALGEBRA must be a statement. There are four kinds of statement: the OPERATOR statement, the TABLE statement, the TRY statement and the null statement. The first symbol on a line identifies what statement it is.

A comment may be appended to any statement. The comment must be preceded by a semicolon, e.g.

```
OPERATOR + ; THIS IS THE 'OR' OPERATOR
```

If two adjacent name symbols occur within a statement, they must be separated by one or more spaces, commas or tabs. Redundant commas, tabs and spaces are ignored. Thus

```
TRY ( A,, + B, ) ; XXX
```

is equivalent to

```
TRY(A+B)
```

but

```
TRY A AND B
```

is not the same as

```
TRY AANDB
```

Similar formatting rules apply to the answers to the questions asked by ALGEBRA, i.e. these may also contain redundant spaces, tabs and commas and they may also have comments appended to them.

Declaration of values

When ALGEBRA is initially entered it asks the question

```
VALUES =
```

At this point the user types a list of different symbols, separated by spaces, commas and/or tabs. These symbols are used thereafter as the value names. They cannot be changed without starting again from scratch.

Note that it is possible to define any number of value names. For example ALGEBRA has proved very useful in examining three-valued logics.

After the values have been declared ALGEBRA proceeds to execute the sequence of statements fed to it. The four different kinds of statement are described below.

The null statement

Null statements are completely ignored and can be used to place comments, for example

;TRY DE MORGAN'S LAW

The OPERATOR statement

Syntax: OPERATOR symbol

Any symbol that is not the name of a value may be used as the name of an operator. If the operator symbol being defined is the same as an existing operator then, provided the OPERATOR statement is completed without error, the new definition overrides the old. Operators can therefore be re-defined. OPERATOR statements need not necessarily be at the start of the data; it is quite legal to add new operators at any time during the use of ALGEBRA.

The OPERATOR statement is followed by a series of data questions-and-answers.

The first question is

UNARY OR BINARY=

and the answer must be either "UNARY or "BINARY"

The second question is

PRECEDENCE=

and the answer must be an integer in the range [0,1000]. The magnitude of the integer does not matter in itself - it is its magnitude relative to the precedence of other operators that counts. For example a precedence of one is the same as a precedence of 998 if the only other operator has precedence 999.

After the above two question-and-answers a series of question-and-answers follows that define the value of the operator for all possible values of its operands. This takes the form

var 1 symbol var 1 =

var 1 symbol var 2 =

.
.
.

var 2 symbol var 1 =

.
.
.

where var 1 is the first value defined in the answer to the

VALUES =

question, var 2 is the second, and so on. For a unary operator the above questions are abbreviated, as shown in the introductory example.

If the answer to any of the above questions is incorrect, then the message

*** EH

is output and the question is repeated.

In some uses of ALGEBRA the user may wish to define operators that are undefined for certain values. This can be done, albeit rather tediously, by including an extra value, called UNDEF say, in the list of values.

The TABLE statement

Syntax: TABLE expression

A table is output of the values of the expression for all possible values of its variables. The expression must include at least one variable. There is no upper limit on the permitted number of variables in the expression, but if the size of the rows of the table exceeds the width of a line the format of the table is upset.

The TRY statement

Syntax: TRY expression

If the expression has the same value, V say, for all possible values of its variables, then the result

= V

is given; otherwise the result

IS A CONTINGENCY

is given.

Abbreviations

Statements are, in fact, identified only by the first two characters of the initial symbol on a line. Thus OPERATOR may be written OP (or even OPZQP). The same applies to the answers to the "UNARY OR BINARY =" question.

User-defined symbols, however, cannot be abbreviated. Thus, for example, if IMPOSSIBLE is chosen as a value name this must be spelt out in full every time it is used.

Errors

All errors are diagnosed and give rise to a message on the message device. These messages should be self explanatory. Except for some errors in answers to questions, which cause the question to be repeated, all errors cause the current statement to be abondoned, and the next one taken.

Interface with KOS

Finally the interface between ALGEBRA and KOS will be described.

Entry and exit

ALGEBRA is entered by the command

&ENTER ALGEBRA

followed by an optional DR-spec. On being entered, ALGEBRA borrows the largest available block of user's workspace, leaving 100 words of it behind in case the user wishes to create job files. If the resultant block contains less than 50 words, a forced break occurs with the message

*** NOT ENOUGH WORKSPACE - BREAK

In general ALGEBRA does not require more than about a hundred words of user's workspace. This is used for storing symbols, the definitions of operators and intermediate values during expression evaluation. ALGEBRA contains no restrictions on the number or size of anything, but since all information is stored in user's workspace, it is possible (though in practice unusual) for ALGEBRA to run out of room if excessive demands are made on the user's workspace that is available. For example overflow would be very likely if a user tried to define fifty different operators, each with a name twenty characters long. If overflow occurs this is detected and an appropriate error message given.

After borrowing its workspace, ALGEBRA borrows its results and data devices, if necessary, outputs an introductory message, and then asks the question

VALUES=

On exit ALGEBRA returns everything it has borrowed and outputs a closing message.

Use of devices

All input is taken from the data device and all output from statements is sent to the results device.

All question-and-answers use the data and results devices. In the non-conversational case, all question-and-answers are compulsory and if a question is not matched an exit is made after the appropriate message has been output.

Breaks and supplementary commands

User breaks are always allowed. If a break occurs during initial entry before the values have been declared, then an exit is made; otherwise all breaks cause ALGEBRA to return to command status within itself.

When ALGEBRA is in command status, either because of a break or because data is exhausted, there is one supplementary command. This can be used to cause ALGEBRA to resume operation, possibly with new results and data devices. It has the form

&CONT

and is followed by an optional DR-spec. The CONT command causes ALGEBRA to continue by reading a new statement from the data device defined on the CONT command. All previously defined value names and operators remain in operation.