STANFORD ARTIFICIAL INTELLIGENCE PROJECT JULY 24, 1964
Memo No. 18

AN EXPRESSION INPUT ROUTINE FOR LISP

by Jan Hext

Abstract: The expression input routine is a Lisp function, Mathread [ ], with associated definitions, which reads in expressions such as

(A + 3 - F(X,Y,Z)).

Its result is an equivalent S-expression. The syntax of allowable expressions is given, but (unlike ALGOL's) it does not define the precedence of the operators; nor does the program carry out any explicit syntax analysis. Instead, the program parses the expression according to a set of numerical precedence values, and reports if it finds any symbol out of context.

The research reported here was supported in part by the Advanced Research Project Agency of the Office of the Secretary of Defense (SD-183)

AN EXPRESSION INPUT ROUTINE FOR LISP

by Jan Hext

1. INTRODUCTION

Mathread is a LISP input routine for reading a conventional mathematical expression. It converts it to a list operator-operand structure. Anything preceding the first "(" is ignored, and input is absorbed through to the matching ")" .

Examples: Input	Result
(A)	A
(A + B ** C)	(PLUS A (EXPT B C))
(A * F(X,(Y),Z))	(TIMES A (F X Y Z))
1964 (3 * (4.2 + M))	(TIMES 3 (PLUS 0.42E1 M))

2. DESCRIPTION

The routine falls into two parts, input and analysis.

Input

A function, Input [ ] , produces the next syntactic element from the input stream. This may be any of the following:-

(1) Name: a letter followed by any number of letters and digits.

(2) Number: a digit followed by any number of digits, periods and E's.

(3) Operator:+ - * / ** = , . ( )

Spaces are ignored, except as terminators of (1) and (2). So is the end-of-row symbol, EOR. All input is through the functions Startread and Advance, so that the data must follow the end of the Lisp program.

Analysis

The syntax of allowable expression is

< expression > ::= < individual> | < individual> <operator> <expression>

< individual > ::= <name > | < number> | (< expression >) | < name >
(< expression >) | < unary operator > < individual>

< operator > ::= + | - | * | / | ** | = | , | .

< unary operator > ::= + | -

The whole expression must be delimited by a surrounding pair of parentheses.

But no syntax analysis, as such, is carried out. If Mathread cannot interpret any part of the input, it ignores the offending symbols, carries through to the end of the expression, and reports, using the Error function. The offending symbols are printed out, preceded by "INPUTERROR", and the report also prints the value of the expression which is finally constructed. This indicates how much input has been absorbed.

3. PRECEDENCE RULES

A function, Group, converts an expression into an operator-operand list structure according to a set of operator precedence values. Pr [O] stands for the binding power of the operator O when it is on the right of an individual; Pl [O] when it is on the left. If O,Q are operators, and x, y, z are individuals, the rules for bracketing

(x O y Q z)

are as follows:

		Meaning	List
Pl [O] >	Pr [Q]	((x O y) Q z)	(Q (O x y) z)
Pl [O] <	Pr [Q]	(x O (y Q z))	(O x (Q y z))
Pl [O] =	Pr [Q]	(x O y O z)	(O x y z)

The last case can occur only when O = Q. The binding powers of the operators are given below: " ** " and "/" associate to the right; "-" to the left. "(" and ")" are treated as pseudo-operators with zero binding power. The internal values of the operators are also given:

O	Pl	Pr	Value
+	30	30	PLUS
-	50	40	MINUS
*	60	60	TIMES
/	70	80	QUOTIENT
**	90	100	EXPT
=	20	20	EQUAL
,	10	10	COMMA
.	110	120	DOT

(	0	0	LPAR
)	0	0	RPAR

"-" as a binary operator has the value DIFFERENCE, but is treated as though it were " + - ", i.e. PLUS MINUS, where MINUS is its unary value. MINUS and DIFFERENCE therefore have the same binding powers. Unary PLUS is ignored.

4. THE FUNCTIONS

The function Input is straightforward. It assumes that the next character to be processed is in CURCHAR and that BOFFO is clear. This state is initially set up by Mathread and is re-set on every exit from Input. The function switches according to the value of CURCHAR and returns the first significant item as its result. It interprets " - " as " + - ", so that " - " will never occur as a binary operator.

The function Mathread prepares for the initial entry to Input, and then sets up three arguments for the recursive function Group. It ignores any input preceding the first " (".

Group [U, H, S] carries out the analysis in a single, forward scan of the input. It is written as a recursive function with three arguments, but since it is in iterative form, it is simpler to think of it as a routine which absorbs the input into three variables; each recursion is equivalent to a cycle assigning new values to these three variables. The significance of the variables is best illustrated by an example.

Suppose that the stream being processed is

(A - B ** 2 * C - D)

By the time we reach "C", the bracketing will effectively be

$$\underbrace{(}_{u_0}\ \underbrace{A\ -}_{u_1}\ \underbrace{B\ **\ 2\ *}_{u_2}\ \underbrace{C}_{H}\ \underbrace{-}_{S}\ D\ )\ \ldots.$$

U = List[u_2, u_1, u_0] - the list of incomplete phrases

H = the item in hand.

S = the next item in the input stream.

The phrases are all held in operator-operand form, so the u_2, for example, is the list

(TIMES (EXPT B 2))

If H is an individual, it is the subject of some competition between the operators adjacent to it; these are in Caar [U] and S. Its destiny is decided by the precedence rules: the operator with higher precedence annexes H to itself.

If H is a unary operator, it opens a new phrase in U, and awaits its one and only argument.

Parentheses require special treatment. " (" is treated as a unary operator; f(x,y,z) is made to look like (f,x,y,z), which eventually yields the completed phrase (, f x y z). The comma, if any, is therefore removed.

Exit is made when U becomes empty.

5. ERRORS

Errors, i.e. symbols occurring where the syntax does not allow them are of two kinds.

(1) Adjacent individuals, as in 1.2B. This is detected when H and S are both individuals, and causes the report ("INPUTERROR1" S).

(2) The second of adjacent operators is not unary. This occurs when H is a non-unary operator, and causes the report ("INPUTERROR2" H).

In both cases, the offending symbol is ignored and input is continued. But NIL is added to the end of the list U, which causes entry to the ERROR function when the analysis is complete. Thus no gash symbols are left on the input stream. If (2) causes ") " to be ignored, input may be absorbed ad lib.; but this could occur in any case, if the opening " (" were not paired properly.

"Numbers", such as 1.2.3, are legitimate (see section 2) and are given a numerical value; but this value is not specified. All number conversion is done according to the standard LISP conventions.

6. MATHREAD

The essentials of Mathread are given below in CPL. Pl [O] and Pr [O] give the precedence values of the operator O; Prefix [O] is true if O is a unary operator.

let Input = ...

This gives the next item on the input stream; we assume that this is a string, e.g. '+', 'Abc' or '1.2'.

```
let  Mathread = result of $ let x = Input
                    until x = '(' do x := Input
                    result := Group [List [List[x]], Input, Input] $$
where rec Group [list U, string H, S] =
Prefix [H] →
   Group [(H='+' → U, Cons [List [H], U]), S, Input],
S = '(' →
   Group [Cons [List [S], U], H, ',' ],
Pr[S] ≥ Pl [Caar[H]]→
   Group [Cons [List [S,H], U], Input, Input] ,
Pr[S] ≤  Pl [Caar [H]]→
   Group [Cdr [U], Pend [Car [U] , H] , S ],
Pr [S] = 0 →
   Group [Cdr [U], (Car[H] = ',' → Cdr [H], H), Input] ,
   Group [Cons [Pend [Car[U], H], Cdr [U]], Input, Input]
```

Pend [L,A] is pseudo-function which adds A to the end of list L.

7. CONCLUSION

It will be realized that the basic outline of the program affords a general precedence analysis, but that various devices have been incorporated to cope with the syntax analysis. If further complications were introduced, it would soon become preferable to include some sort of strict syntax analysis as a preliminary stage.