29 Nov 67

To: 6.231 Staff
From: Art Evans

Here is still another attempt at PAL syntax. It is Martin's parse algorithm.

PAL Syntax as a Parse Algorithm

Introduction (This section was last modified on 11/24/67 at 16:53 by Evans.)

This section defines the syntax of PAL using a set of tables accompanied by an algorithm that uses the tables as data. The algorithm examines a string alleged to be a PAL program. If it is, the program is parsed; and if not, that fact is detected and reported. The programmer who understands this section may use it to deduce quite readily how PAL will parse any program.

Two mutually recursive procedures are defined, E and D, each of which has a single integer parameter. The effect of calling E(n) is (roughly) to read an expression with precedence n. D(n) performs similarly for definitions. In addition, routine BV reads the bound variable part of a definition or of a lambda expression. Finally, the function Scan is used to scan new characters from the source string.

The process is initiated by calling first Scan and then E(0). On return from E(0), the string has been scanned. If the string is not a legal PAL program, an error will have been signaled.

It is assumed that the character "X" has been appended as the right-most character of the input, to signal the end of the text.

The Driving Tables

The tables that drive the algorithm are shown at the end of this section. Each table is in two parts - Part A and Part B. The usual operation is to look up a character in the first column of a table. A vertical bar in that column indicates "or", so there will be a match on the fifth line of Part A of Table E if the looked-up character is either "+" or "-".

If there is more than one character in an entry in column one of Part A (as on the second line), the intent is that the first character is to be compared with Current and the second with Next. Only if both match is the line said to match. If the match does take place, Scan is called an extra time before proceeding.

The Routine "Scan"

Scan is the routine that reads successive characters from the input text. For the purposes of this discussion, it is assumed that Scan does lexicographic analysis. That is, an entire variable or an entire constant or an entire system word is regarded as a single character. Scan does the necessary processing to permit that assumption.

At any instant, the variable Current has the last character that has been read and the variable Next has the next character to be read. Calling *Scan* causes *Current* to be replaced by *Next* and the next character from the source to go into *Next*.

The Routines 'D' and 'E'

Routines *D* and *E* are identical in execution, with the sole exception that *D* uses Table D and *E* uses Table E. In the discussion that follows, reference to "the table" should be interpreted with that understanding. The routines proceed as follows:

Step 1: Let *n* be the parameter with which the routine was called.

Step 2: Call *Scan*.

Step 3: Look up *Current* and *Next* in the first column of Part A of the table. (It is a reported error if no match is found.) Let *Goal* be the contents of the second column opposite the found entry.

Step 4: If *Goal* is empty, go to step 6.

Step 5: Call the routines indicated in *Goal*.

Step 6: Look up *Next* in the first column of Part B. If it is not found, return.

Step 7: In the row in which the match was found, let *p* be the contents of the second column and *Goal* the contents of the third column.

Step 8: If n is greater than or equal to p, [redacted] then return.

Step 9: Call Scan, and then go to step [redacted] 4.

This completes the discussion of routines D and E.

The Routine "BV"

The routine BV is called to read the bound variable part of a lambda expression or of a function-form definition. It is not convenient to specify BV in the tabular form which is used for E and D, so a BNF definition of the strings read by BV is given instead:

<BV> ::= { <bv element> }$_{1}^{\infty}$

<bv element> ::= () | <$\overset{N}{\cancel{\text{variable}}}$>
| (<$\overset{N}{\cancel{\text{variable}}}$> { , <$\overset{N}{\cancel{\text{variable}}}$> }$_{0}^{\infty}$)

BV reports error if the available text does not match any string defined by this definition.

On entry, BV calls Scan to read the first character of its input. On [redacted] exit, the last character of the bound variable is in Current.

The Character-Reading Routines

When the Goal contains a character (such as "in" or "."), a routine is invoked that calls Scan and then insures that that character is in Current. If so, it returns; while if not it reports error.

identifier	number string true false N
N	programmer's variable

Table E - Part A

let	D(0) in E(0)
N :	E(3)
goto	E(9)
not	E(14)
+ | -	E(25)
J | Y	E(30)
lambda	BV . E(3)
(	E(0))
~~name~~ identifier	

Table E - Part B

¤	0	
where	1	D(0)
;	3	E(1)
:=	5	E(3)
,	9	E(~~8~~)9 $\{, E(9)\}_0^\infty$
->	10	E(9) , E(9)
log or	13	E(10)
log and	14	E(13)
= | < | >	20	E(20)
+ | -	25	E(25)
* | /	30	E(25)
(	35	E(0))
~~name~~ identifier	35	

Table D - Part A

(	D(0))
N =	E(0)
N	BV = E(0)
pp | rec	D(6)

Table D - Part B

within	3	D(0)
and	6	D(3)