object \( \equiv \) objprim \mid objprim : id

objprim \( \equiv \) id \mid (objloc)

wordloc \( \equiv \) objprim \& octexp \mid \& octexp

octexp \( \equiv \) octterm \mid octexp + octterm \mid octexp - octterm \mid - octterm

octterm \( \equiv \) octprim \mid octterm \times octprim

octprim \( \equiv \) id \mid integer octinteger \mid (octexp)

wordexpr \( \equiv \) wordpart \mid wordexpr \& wordpart

wordpart \( \equiv \) decimal integer \& octprim

\[ \text{read unit} \]

syntax of parameters (no blanks)

object \( \equiv \) objprim \mid objprim : id

objprim \( \equiv \) id \mid objprim \& octexp \mid \& octexp

objprim \( \equiv \) id \mid (objloc)

wordloc \( \equiv \) objprim \& octexp \mid \& octexp

octexp \( \equiv \) octterm \mid octexp + octterm \mid octexp - octterm \mid - octterm

octterm \( \equiv \) octprim \mid octterm \times octprim

octprim \( \equiv \) id \mid integer octinteger \mid (octexp)

wordexpr \( \equiv \) wordpart \mid wordexpr \& wordpart

wordpart \( \equiv \) decimal integer \& octprim

\[ \text{may contain letters, etc.} \]
Each expression has a value. This value is computed by first computing the value of each subexpression. The rule used in forming the main expression then determines a function to be applied to the values of the subexpressions to determine the value of the main expression.

For each rule we now state the function determined by it. For some rules with exactly one subexpression that function is the identity function. Those rules will not be listed.

There are several different kinds of values. They are:

- object
- object locations
- a list and an index within it
- a capability
- a word of less than or equal to 60 bits
- an index within it
- a directory, name and accessory
- an index and none
data locations

a file and index within it

The user specifies full core and
address within it

(?)

The user chooses a pair and
within it

a name

ascii text

available

can contain other object values

certain primitive expressions have values as follows

id a name
object element datum
decimal integer datum

The locations are represented by an obvious map illustrating

\texttt{CLIST\_SLOT} \(\left( C, I \right)\)

meaning \(C\) is the clist and \(I\) is in it
and \texttt{NEWTOK expression} means the
8th slot within clist \(C\).
The functions determined by rules which are not subject to functions are as follows:

\[ \text{The value of objprim is used as:} \]

\[ \text{The context of objprim is used as:} \]

The notation is to first state the rule. Then we tell you how to compute the desired function. The name of a sub-expression stands for the value of the sub-expression, \( \text{val(exp)} \) means either the value of the exp if it is an object or datum or the contents of the location that is a datum loc or objective.

\[ \text{objloc} \text{ is objprim } \ast \text{id} \]

\[ \text{scan} 2.3 + (\text{val(objprim)}, 1d) \]

\[ \text{objloc} \text{ is objprim } \ast \text{id} \ast \text{objprim} \]

\[ \text{directory} (\text{val(objprim)}, 1d, \text{val(objprim)_2}) \]
objloc := objprim * octexp

cdata (val(objprim), octexp)

objloc := objprim * octexp

userfullclist (octexp + cclistbase)

objprim := id

if there is a variable of name id then that variable
else scandlist (currentscanlist, id)

wobjloc := objprim * octexp

fileaddr (val(objprim), octexp)

wobjloc := * octexp

userfullcore (octexp + corebase)
outexp = outform

use obvious arithmetic operation on a full 60-bit word

outprim ss = id

if id is the name of a variable whose value is a decimal,
then that decimal,

wordpost ss = decimalize id outprim

a pair <decimalize, outprim>

wordexpss = wordpost

2nd half of wordpost

wordexp ss = wordexp, wordpost

wordexp + 2 * half (wordpost) + 2

(as assigned 60 bit integer)