Do Not Borrow

Artificial Intelligence Project--RLE and MIT Computation Center
Memo 24--Arithmetic in LISP 1.5
by
Michael Levin

April 28, 1961

As of the present, the following parts of LISP 1.5 are working. This is an excerpt from the forth coming LISP 1.5 Programmer's Manual.

4.4 Arithmetic in LISP

LISP 1.5 has provisions for handling fixed point and floating point numbers and logical words. Available functions include the arithmetic operations, arithmetic predicates, and logical and shifting operations for logical words.

Numbers read into a LISP program are treated as constants. They need not be quoted. Numbers read in are marked internally as being either fixed point or floating point, with octal numbers or bit patterns treated as fixed point numbers

a. Floating Point Numbers

The rules for punching these for the read program are:

1. A decimal point must be included but not as the first or last character.

2. A plus sign or minus sign may precede the number. The plus sign is not required.

3. Exponent indication is optional. The letter E followed by the exponent to the base 10 is written directly after the number. The exponent consists of one or two digits which may be preceded by a plus or minus sign.

4. Absolute values must lie between 2^{128} and 2^{-128} ($\sim 10^{38}$ and 10^{-38}).

5. Significance is limited to eight decimal digits.

6. Any possible ambiguity between the decimal point and the point used in dot notation may be eliminated by putting spaces before and after the LISP dot. This is not required where there is no ambiguity.

The following are examples of correct floating point numbers. These are all different forms for the same number, and will have the same effect when read in.

60.0
6.E1
600.00E-1
0.6E+2

The forms .6E+2 and 60. are incorrect because the decimal point is the first or last character.

b. Fixed Point Numbers

These are written as integers with an optional sign. Examples:

-17
32719

c. Octal Numbers or Logical Words

The correct form consists of

1. A sign (optional)
2. Up to 12 digits (0 through 7).
3. The letter Q.
4. An optional scale factor. The scale factor is a decimal integer, no sign allowed.

Examples are:

a. 777Q
b. 777Q4
c. -3Q11
d. -7Q11
e. +7Q11

The effect of the read program on octal numbers is as follows.

1. The number is placed in the accumulator three bits per octal digit with zero's added to the left hand side, to make twelve digits. The right most digit is placed in bits 33-35, the twelfth digit is placed in bits P, 1, and 2.

2. The accumulator is leftshifted three bits (one octal digit) times the scale factor. Thus the scale factor is an exponent to the base eight.

3. If there is a negative sign, it is OR-ed into the P bit. The number is then stored as a logical word.

The examples a through e above will be converted to the following octal words. Note that because the sign is OR-ed with the 36th numerical bit, c, d and e are equivalent.

a. 000000000777
b. 000007770000
c. 700000000000
d. 700000000000
e. 700000000000

d. Arithmetic Functions

These functions may be applied to mixed expressions involving fixed point and floating point numbers. If the expression contains at least one floating point number, all numbers will be floated, and the value will be a floating point number. Octal numbers are considered as fixed point.

plus[x_1;...;x_n] is a function of n arguments whose value is the algebraic sum of the arguments.

difference[x;y] has as value the algebraic difference of its arguments.

minus[x] changes the sign of its argument.

times[x_2;...;x_n] is a function of n arguments, whose value is the product (with correct sign) of its arguments.

addl[x] has x+1 as its value. The value is fixed point or floating point depending on the argument.

subtl[x] has x-1 as its value.

max[x_1;...;x_n] choses the largest of its arguments as its value.

min[x_1;...;x_n] choses the smallest of its arguments as its value.

recip[x] computes 1/x. The reciprocal of any fixed point number is defined to be zero.

These functions may occur inside of each other and may be used recursively. In some cases the arguments may be evaluated before the arithmetic function is performed. We illustrate this with the following examples.

Suppose that the evalquote operator is given the following doublet:

```
APPLY
(DIFFERENCE (PI 2) ((PI . 3.14159)) )
```

The apply function expects a list of arguments already evaluated. Since PI is not a number, an error will result.

Now consider the following doublet:

```
EVAL
( (DIFFERENCE PI 2) ((PI . 3.14159)))
```

Eval will evaluate 2 and PI and then perform the function plus. "2" will evaluate to "2" because numbers are constants in LISP 1.5. "PI" will evaluate to "3.14159" because it is tied to that number on the pair list. The correct answer will be 1.14159.

e. Arithmetic Predicates

We now list the arithmetic predicates. The rules concerning mixed expressions and evaluation of arguments are the same as for the arithmetic functions. The value of a predicate is NIL (false) or true.

lessp[x;y] is true if $x \leq y$, and false otherwise.

greaterp[x] is true if $x \geq y$.

zerop[x] is true if $x=0$, or if $|x| \leq 3^{-6}$.

onep[x] is true if $x=1$.

minusp[x] is true if x is negative.
"-0" is negative.

numberp[x] is true if x is a number (fixed point or floating point).

fixp[x] is true only if x is a fixed point number. If x is not a number at all, an error will result.

floatp[x] is similar to fixp[x] but for fixed point numbers.

equal[x;y] works on lists, fixed point numbers, and floating point numbers. Its value is true if the arguments are identical lists, or identical numbers. Floating point numbers must be exactly equal for the value to be true.

f. Logical Functions

These operate on 36 bit patterns. The only acceptable arguments are fixed point numbers. These may be read in as octal or decimal integers, or they may be the result of a previous computation.

logor[x_1;...;x_n] performs a logical OR on its arguments.

logand[x_1;...;x_n] performs a logical AND on its arguments.

logxor[x_1;...;x_n] performs an exclusive OR

$(0 \vee 0 = 0,\ 1 \vee 0 = 1,\ 1 \vee 1 = 0)$.

g. Example

In the beginning of this manual, an example of a recursive definition was given for n!.

n! = [n=0 → 1;T → n·(n-1)!]

To define this in a LISP program, we punch:

```
DEFINE((
(FACTORIAL (LAMBDA (X) (COND
     ((ZEROP X) 1)
     (T (TIMES N (FACTORIAL (SUB1 N)))) )))
))
```

(End of section 4.4 LISP 1.5 Manual)
If you have any complaints, bring them to 26-265.

Note on special forms:

Functions of an indefinite number of arguments are called special forms. Special forms are treated in the same manner as functions with the following exceptions:

1. The apply function will not recognize a special form as its first argument.

2. The evalquote operator, when it encounters a special form as its first argument, gives the list of the form and its arguments to eval, i.e.

evalquote[(f;(a,b,c,d))] = eval[(f,a,b,c,d);NIL]

if f is a special form.

Special forms include PLUS, TIMES, MAX, MIN, LOGOR, LOGAND, and LOGXOR.

Consider the previous example

```
APPLY
(DIFFERENCE(PI 2) ((PI . 3.14159)) )
```

which we said would cause an error. If DIFFERENCE were changed to PLUS, another type of error would result because apply will not recognize a special form. Since evalquote will recognize special forms and evaluate them, the following example is correct.

```
TIMES
(2 (TIMES 3 5)
The answer is 30.
```

Read Program for Numbers

There is another method of reading in decimal numbers. It uses a binary scale factor B, and its result is a fixed point number. Examples are

```
6E1B2   120
6B2     12
6.5B2   13
6.3B2   12
```

All LISP number conversion is done by Share UADBC1, except for octal numbers.

Winne ille pu informs us that *T* has been changed to VERITAS-NUNQUAM-PERIT. "-" should be written as an 11 punch.

There is a new function for printing the time: TEMPUS-FUGIT ().

Chapter 4

f. Arrays

Provision is made in LISP 1.5 for allocating blocks of storage for data. The data may consist of list structure or data words. Arrays may have up to three indicies.

To declare an array use the function array. Its argument is a list of arrays to be declared. Each one must be for list structure or non-list structure.

Suppose ALPHA is to be a 10 x 10 array of list structure, and BETA a 5 x 5 x 5 array of non list structure. The array function is

```
ARRAY ((
     (ALPHA (10,10) LIST)
     (BETA (5,5,5) NONLIST)    ))
```

To find the value of $B_{3,4,2}$: (BETA 3,4,2) or (BETA I J K) with a pair list.

To set $ALPHA_{3,4}$ to "YES": (ALPHA (QUOTE SET) (QUOTE YES) 3 4)

Array uses marginal indexing for maximum speed. The total number of words used by an array whose size is $D_1 \times D_2 \times D_3$ is $4 + D_1 + D_1D_2 + D_1D_2D_3$. If the array is 2 dimensional, $D_1 = 1$. If the array is 1 dimensional, D_1 and $D_2 = 1$.

To save space, specify dimensions in increasing order.

ALPHA (3,4,5) takes less words than ALPHA (5,3,4).

Compatability of LISP 1 and LISP 1.5.

1. EVALQUOTE has two arguments while APPLY has three. To change a LISP 1 program for LISP 1.5 simply eliminate the p-list if it is null. If the p-list is needed, then the function apply is available in LISP 1.5.

2. Arithmetic in LISP 1.5 is new, improved, and generally incompatible with LISP 1.

3. LISP 1.5 has many extra features. There are being written up for the new LISP 1 Programmer's Manual. Until it comes, check in Room 26-265.

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM - 24

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 8(12-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☒ Single-sided or
☐ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: COPY OF MIMEOGRAPH

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-8)	1-8
(9-12) SCANCONTROL, TRGT'S (3)	

Scanning Agent Signoff:
Date Received: 11/30/95 Date Scanned: 12/12/95 Date Returned: 12/14/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 12/18/1995

M.I.T. Libraries
Document Services