DATALOGILABORATORIET
Department of Computer Sciences
Uppsala University
June 1979

DLU 79/4

Floating point numbers and arrays in Lisp F3

by

Mats Carlsson

TABLE OF CONTENTS

1 INTRODUCTION

In 1978, the current version of the LISP interpreter LISP F3 was released. The LISP dialect used is a subset of Interlisp, and the system is written in Fortran IV.

The system is carefully documented in <No 78a> and <No 78b>, and is available from DLU.

This report describes an extension of the system. The extended system supports the following new data types:

-- arrays containing pointers,
integers and
floating point numbers;

-- floating point numbers
replacing big integers

2 INTERNAL REPRESENTATIONS

Floating point numbers

are stored exactly as the big integers of the standard version: in consecutive floating point words in upper PNAME.

Arrays

are stored in a way similar to that of atoms and strings. All array data are stored in the Fortran array PNAME. Arrays are referred to by pointers in the range (T+1, NATOM). Every array consists of three parts, each one of which may have zero length. The details are as follows:

(z is zero or more slack bytes.)
(i,j,k,l are byte pointers into PNAME.)

```
x        = the array pointer value
CAR(x)   = LISPF3-ARRAY
CDR(x)   = NIL
PNP(x)   = i
PNP(x+1) = l

PNAME:
 ... (z j k (pointers) z (integers) z (f. p. n.)) ...
      !                !           !             !
      i                j           k             l
```

The three array parts all start on a word boundary. By a word we mean the basic storage cell for a data type. (E.g. half machine word, full machine word, double machine word, etc.)

By this means it is possible to call a Fortran subroutine with an actual parameter consisting of an address to an array part if the corresponding formal parameter is a properly typed array. The advantage of this method, instead of having pointer arrays, is that we do not have to "unbox" and "box" numbers before and after calls to e.g. numerical routines. For further details see the description of subroutine ARRUTL (on page 9).

3 EXTERNAL REPRESENTATIONS

Floating point numbers

a) Input

Atoms obeying the following syntax are treated as numbers. If they cannot be stored as small integers, they are treated as floating point numbers:

```
number::= sm[Esn]
s      ::= [+!-]
n      ::= d[n]
d      ::= 0!1!2!3!4!5!6!7!8!9
m      ::= n[.[n]]!.n
```

The "E" has the same meaning as in Fortran.

b) Output

Floating point numbers are printed as compactly as possible, either with or without the "E".

Arrays

cannot be read-in. They are printed as

```
[xxx
```

where xxx is the array pointer value.

4 ADDED FUNCTIONS, CALLABLE FROM LISP

(ARRAY s si sf) = a

creates an array with space for s elements. Out of these s elements, si are to be integers and sf are to be floating. The elements are initially set to NIL, 0, and 0.0, respectively.

(ARRAYSIZE a) = (s si sf)

gives the sizes of the array a. S, si and sf have the same meaning as above.

(ELT a j) = x
(ELTI a j) = i
(ELTR a j) = f

picks the element j of the pointer, integer or floating part of a. Within each part, the elements are indexed from 1 and upwards.

(SETA a j x) = x
(SETI a j i) = i
(SETR a j f) = f

sets the element j in the proper part of array a.

(IQUOREM j k) = (quotient . remainder)

This function performs an integer division of j by k.

(ARRAYP a) = a or NIL

ARRAYP returns a if a is an array, otherwise NIL.

(FIXP i) = i or NIL

FIXP returns i if i is a small integer, otherwise NIL.

(FLOATP f) = f or NIL

FLOATP returns f if f is a floating point number, otherwise NIL.

Arithmetics

There is no special floating point arithmetics. Instead, as long as all operands in a computation are integers, integer arithmetics is used. If any operand is floating, floating point arithmetics is performed.

5 IMPLEMENTATION GUIDE

Simply follow the implementation guide of <No 78a>. Please note, however, that the following common variables have been added and must be given their proper machine-dependent value in INIT1:

JBYTES No of bytes in a pointer.

IBYTES No of bytes in an integer.

BYTES No of bytes in a floating point number.

IRESOL The precision of a floating point number (in decimal digits).

IPOWER max(abs(p)), if floating point numbers are represented as

m*10**p, abs(m)<10.

FUZZ will compensate for truncation errors in certain cases. Should be set to 5*10**(-IRESOL).

Caution! IBYTES and BYTES must both be multiples of JBYTES, otherwise GARB will fail.

6 NOTES ON THE FORTRAN CODE

To be able to make the extension of the interpreter, it was unfortunately necessary to introduce certain changes in many subroutines. In this chapter we give a list of the more substantial modifications.

6.1 SUBROUTINE ARRUTL (IPTR,IACTN,IPART,IFIRST,ILEN)

is a new subroutine that performs various array handling functions. IPTR is a pointer to the array, IPART tells which part of it (1, 2, or 3), and IACTN tells what action to be done:

```
IACTN = 1  get array element IFIRST
        2  set array element IFIRST
        3  set IFIRST and ILEN to array part bounds
```

In cases 1 and 2, IFIRST is the array part index and ILEN is the PNAME index to the value. In cases 3 and 4, IFIRST is the PNAME index of the array part and ILEN is the number of elements.

Ex:

Suppose that you want to transmit the floating part of array IA to a numerical routine. Just do:

```
CALL  ARRUTL(IA,3,3,IFIRST,ILEN)
CALL  NUMRUT(PNAME,IFIRST,ILEN,...)
```

6.2 SUBROUTINE PRIFLO(F)

is a new subroutine that prints the floating point number F.

6.3 SUBROUTINE PRIINT(I)

is a new subroutine that prints an integer.

6.4 INTEGER FUNCTION RATOM(X,IOP)

has been extended so that it will recognize floating point numbers.

6.5 INTEGER FUNCTION GARB(GBCTYP)

has been extended to take care of pointers in arrays, using the following algorithm:

In the mark phase, a list of all encountered arrays is built up. The list is headed by the variable ARRLST. Then -CDR(ARRLST) points to the next array etc., until we get to NIL. The mark phase is finished by four steps:

1. If ARRLST=NIL then exit.
2. S:=ARRLST; ARRLST:=-CDR(S); CDR(S):=-NIL.
3. "Start marking from all pointers stored in the array S".
4. Go to 1.

6.6 SUBROUTINE MARKL(IS,GBCTYP,ARRLST)

Treats arrays as in GARB.

6.7 FUNCTION MKREAL(R)

Stores a floating point number in upper PNAME and returns a Lisp pointer to it.

6.8 FUNCTION GTREAL(I,IRETUR)

Returns the value of a floating point number referred to by the Lisp pointer I. IRETUR will contain the closest integer number.

If I points to a small integer, GTREAL will return 0.0 (but IRETUR will still return the integer value).

APPENDIX A

REFERENCES

No 78a Mats Nordström, "Lisp F3 - Implementation Guide and System Description", DLU 78/3.

No 78b Mats Nordström, "Lisp F3 - User's Guide", DLU 78/4.