Zbornik radova Filozofskog fakulteta u Nišu
Serija Matematika 6:2 (1992), 291–298

INTERPRETER FOR APPLICATION OF MATHEMATICAL SPECTRA

M. Stanković, J. Madić, P. Stanimirović

(Received 10.10.1992.)

Abstract. *In this paper it is described a programming package implementing the interpreter of programming language for manipulation of mathematical spectra, representing an extension of LISP. For this purpose several new data types and new kind of functions (called spectral data types and spectral functions) are defined and implemented.*

1. Introduction

The theory of mathematical spectra was founded by M. Petrović [9]. K. Orlov discovered the practical applications of it [1], [2].

Definition 1.1. The mathematical spectrum of a sequence of integers $a_0, \dots, a_n$, i.e. the spectrum of the polynomial $P(x) = a_0x^n + a_1x^{n-1} + \dots + a_n$ is the ordered pair (S, h), where $S = P(10^h)$ is the spectral value, and h is the spectral rhythm determined by the condition $10^h > 2 * max\,\{|a_i|\}$.

Definition 1.2. The mathematical spectrum of a sequence of fixed-point numbers $a_0, \dots, a_n$ is the ordered triad (S, h, k), where k is the maximal number of decimal digits and (S, h) is the spectrum of sequence of integers $b_0, \dots, b_n$ obtained from the equalities $b_i = 10^k * a_i, i = 0, \dots, n$.

B. Mihajlović was implemented the multiplication of two mathematical spectra using programs in FORTRAN IV ([4] [5]).

J. Madić was introduced the division of two mathematical spectra and developed the first programming package for the experimental work with mathematical spectra ([3] [7]). This programming package implements programming language representing an extension of ADVANCED FORTRAN.

2. The interpreter of the programming language

The interpreter is an infinite loop in TURBO C. It performs the following actions on every pass:

- reads a complete expression;
- translates the entered expression in an internal form;
- computes the value of the internal form in the internal form;
- translates the internal form to the external form, if the value was successfully computed.

1991 Mathematics Subject Classification: 11Y05.
Supported by Grant 0401A of FNS through Math. Inst. SANU

2.1. Data types The programming language contains two main data types.

Lisp data types:
lists, symbols, strings, integers, fixed-point numbers, rational numbers, file-pointers, NIL, T, LISP-functions.

Spectral data types :
vectors, spectral constants, spectral functions, spectral expressions.

2.2. Spectral constants A spectral constant represents the notation of the spectrum with uniform rhythm. This notation begins and ends with symbol '\$'. The sign of the spectrum follows first symbol '\$', and could be omitted. Strips are separated using character '|'.

EXAMPLE 1. Here are several spectral constants:

a) \$23|78|59\$ b) \$ − 179|054|391|502\$ c) \$ − 174.23|505.63|00.95\$.

2.3. The internal form of a spectrum The internal form of a spectrum is a structure with two elements.

One of them is defined as single-dimension array of integers containing following data stream: spectral sections number, spectral rhythm, sign of the spectrum, number of decimal digits (zero if the spectrum represents a sequence of integers), and the other is defined as a single-dimension array which elements are equal to digits of the representing spectrum.

2.4. The internal form of a spectral constant A spectral constant is represented as the element of the union representing the internal form of all data types. This element is the internal form of the corresponding spectrum. We use pointers to access the elements of this structure.

2.5. Vectors A vector is a sequence of numbers between square brackets. The internal form of the vector $[a_1, \dots, a_n]$ is the binary tree representing list $(a_1 \dots a_n)$, except the element of the union indicating data type.

2.6. Transformation of the internal form of a vector to the internal form of the corresponding spectrum

ALGORITHM 1.

Input parameter is the pointer to the tree representing a vector.

```
STEP 1. Using cars of the tree we obtain elements of the vector.
STEP 2. Return the internal form of the spectrum corresponding to the given numbers.
```

2.7. Transformation of the internal form of a spectral constant to the internal form of corresponding vector

ALGORITHM 2. Pointer to the internal form of a spectral constant of n strips is given.

```
STEP 1 Define a tree with initial value cons(NIL, NIL).
STEP 2. The body of the for-loop performs steps A and B, repeating $n$ times.
    Step A. Compute the effective value of the current strip.
    Step B. Transform given integer to the internal form and replace cdr of the tree.
```

STEP 3. Return pointer to the given tree.

2.8. Spectral functions The interpreter evaluates all arguments of the spectral functions. Arguments of these functions could be spectral expressions or LISP expressions whose values are numbers, vectors or spectral constants.

We can note several different functions types.

A. Functions for the implementation of spectral operations:
spectral addition $+ ;
spectral multiplication $* ;
spectral subtraction $- ;
right lengthening $rlen ;
left lengthening $llen ;
left condensation $lcon ;
Conversion of a spectrum into a vector $eval ;
Effective value of a strip $efval ;
Inverse spectrum $sinver ;

B. Forming new spectra:
Spectral generation $gensp ;
Generation of the spectrum whose strips are 1 $1 ;
Generation of the spectrum whose strips are 0 $0 ;

C. Spectral relation functions:
Equality of two spectra $= ;
Identity of two spectra $ident ;
Comparing of absolute values of two spectra $\$>, \$<$.

2.9. Spectral expressions Spectral expressions could be separated into three groups:
1. Spectral expressions whose values are spectra;
2. Spectral expressions whose values are vectors;
3. Spectral relation expressions.

The first group contains following expressions: spectral constants, symbols bounded with spectra, calls of the spectral functions $\$+$, $\$-$, $\$*$, $\$rlen$, $\$llen$, $\$gensp$, $1, $0.

The second group contains: vectors, numbers as single-element vectors, symbols bounded with vectors, calls of the spectral functions $\$+$, $\$-$, $\$*$, $\$eval$, $\$efval$.

Elements of the third group evaluate to the LISP constants NIL (as false) or T (as true). This group of spectral expressions contains: NIL, T, any expression whose value is T or NIL, calls of the spectral functions $\$=$, $\$>$, $\$<$, $\$ident$.

A call of spectral function is the list whose head is the name of the called function and the tail is the argument list.

2.10. Functions $+ , $- and $*

DESCRIPTION. $(\$+ \quad x\ y)$ $(\$- \quad x\ y)$ $(\$* \quad x\ y)$.
Both arguments are spectral expressions.

IMPLEMENTATION. The corresponding C function detects the following cases:

A. Both arguments are pointers to the internal forms of spectral constants. The result is the internal form of the spectral constant equal to the sum, difference or product of input spectra.
B. Both arguments are pointers to the internal form of vectors. The result is pointer to the internal form of vector obtained in this way:
STEP 1. Convert input parameters into the internal form of the corresponding spectra using Algorithm 1.
STEP 2. Compute the resulting spectra applying corresponding arithmetic operation.
STEP 3. Return the internal form of the vector obtained according to Algorithm 2.
C. Pointer to the internal form of a number n replace by the pointer to the internal form of the vector $[n]$.
D. If exactly one argument is pointer to the spectral constant, transform the other argument into the internal form of the spectral constant by applying Algorithm 1. Then the case A is applied.

EXAMPLE 2.

```
> ($+  $25|78|14$  $-170|543$)
@$+025|807|471$
> ($+  (setq  a  $-44|09|57|11$) (setq  b  [1  -2  34455  928]))
@[-43 -12 34498 917]
> ($+  $-12.23|11.20$  $-19.23|65.11$)
@$-07.00|53.91$
> ($*[46 -5448][2621 -23])
@[1196 -438 -944 2250 -1104]
```

2.11. Function $eval

DESCRIPTION. ($*eval* x)

x must be a spectral expression. If the value of x is the spectral constant, then the result is the vector whose elements are effective values of the spectral strips, otherwise, if the value is a vector, the result is the same vector.

IMPLEMENTATION. If the parameter of the corresponding C function is pointer to the internal form of a vector the result is the same pointer. If the parameter is pointer to the internal form of a spectral constant the result is pointer to the internal form of the vector determined after the application of Algorithm 2.

EXAMPLE 3.

```
> ($eval  $-23|67|09$)
@[-24 33 -9]
> ($eval  (+  [1/2  (-  1/4  2)
    (setq  z  (*  2  3))]  [7/2  1  (+  3  1)])
@[4 -3/4 10]
> ($eval  $-375.25|012.26|673.20$)
@[-375.250000 -12.270000 326.800000]
```

2.12. Function $gensp

DESCRIPTION. (*$gensp x*) (*$gensp x y*).
If one argument is taken, it must be a spectral expression. The result is the spectral constant of the array which is contained in the given vector or the same spectral constant given as argument.
If the function is called using two arguments, then first argument is an expression whose value is integer n, and the second is an expression whose value is number b. The result is the spectral constant of n spectral strips whose nominal values are b.

IMPLEMENTATION. The corresponding C routine detects the following cases:
A. If one pointer to a spectral constant is given as formal parameter, it is returned.
B. If one pointer to the internal form of a vector is given, then the result is pointer to the internal form of the corresponding spectral constant (Algorithm 1).
C. If we use two formal parameters, then first must be pointer to the internal form of an integer n, and the second pointer to a number b. Nominal values of the spectral strips are equal to b.

EXAMPLE 4.

```
> ($gensp [1 -2 34 2726])
@$+0000|9998|0034|2726$
> ($gensp [1.2 3.4942668 229.202])
@$+001.200000|003.494267|229.20200$
```

2.13. Functions $rlen i $llen

DESCRIPTION. (*$rlen x y*) (*$llen x y*)

First argument should evaluate to a spectral constant or a vector. The second argument is an expression whose value is an integer n. The result is spectral constant equal to the previous constant, whose strips are lengthened on the right or on the left for n figures.

IMPLEMENTATION. The description of the corresponding subprogram follows:
STEP 1. If the first pointer points to the internal form of an vector, transform it applying Algorithm 2.
STEP 2. Return pointer to the internal form of the lengthened spectrum.

2.14. Function $lcon

DESCRIPTION. (*$lcon x*).

x must be a spectral expression. If x evaluates to a vector it is transformed into corresponding spectrum (Algorithm 1). The result is spectral constant equal to the previous constant, obtained after condensation on the left minimizing the number of figures.

2.15. Function $sinver

DESCRIPTION. (*$sinver x*).
x must be a spectral expression. The result is spectral constant representing inverse spectrum of the spectrum given after evaluation of x.

2.16. Functions \$1 i \$0

DESCRIPTION. (\$1 x) (\$0 x)
The argument is an expression whose value is integer n. Values of these expressions are the spectral constants defined in this way: the rhythm is one, and contains n spectral strips whose nominal values are one (zero).

2.17. Function \$efval

DESCRIPTION. (*\$efval* x y)
The first argument must be a spectral expression and the value of the second expression must be integer n. The result is the nth element of the vector or the effective value of the nth strip of the given spectral constant.

EXAMPLE 5.

```
> ($efval  $-1475|0932|673|1129$  ($efval  $-11|25|47$  1))
@ - 1475
> ($rlen  [1  (-  2 4)  34  2726]
    ($efval  [(/  2  5)  2(*  2/3  1/40)]  2))
@$ + 100000|999998|000034|002726$
```

2.18. Functions \$ident , \$= , \$> , \$<

DESCRIPTION. (*\$ident* x y) (\$ = x y) (\$ > x y) (\$ < x y).
Both arguments must be spectral expressions. The vector given as value of any argument is converted into corresponding spectral constant. The result is T if the spectral constants satisfy the given relation, otherwise NIL.

3. Several applications of the programming language

The theory of mathematical spectra was applied in the following cases: matrices, determinants, manipulations with polynomials, numerical tables, solution of algebraic and differential equations, arithmetic operations with large integers. In all of these cases the introduced language could be applied.

3.1. Internal product of two vectors Let given sequences $a_i, b_i\ i = 1, \dots, n$. Their internal product $\sum_{i=1}^{n} a_i b_i$ is equal to the effective value of the central strip (i.e. nth strip) of the spectrum obtained after multiplication of the spectrum of sequence a and the spectrum of reversed sequence b ([7]).

```
> (defun  inprod(s1  s2)
       (cond  ((=  (length  s1)(length  s2))
           ($efval  ($*  s1  ($sinver  s2))(length  s1))
       )    (t  (print  'Product_not_found))))
@inprod
> (inprod  [172  -2928 181][-1 292 33.1])
@ - 849156.900000
```

3.2. Computing the binomial coefficients For a given k and n, the all binomial coefficients

$$\binom{k}{i},\ i = 1, \dots, k; \quad k = 1, \dots n$$

can be computed by using the functions *bin* and *bin*1 defined below.

```
> (defun   bin(n)
       (prog  ((b  $1|1$ ))
           again
           (cond  ((=   n   1)  (return  b)))
           (setq  b  ($ *   $1|1$  b))
           (setq  n  (1 -   n))
           (go  again)))
> (bin  25)
@$00000001|00000025|00000300|00002300|00012650|00053130|
00177100|00480700|01081575|02042975|03268760|04457400|05200300|
05200300|04457400|03268760|02042975|01081575|00480700|00177100|
00053130|00012650|00002300|00000300|00000025|00000001$
> (defun   bin1(n)
       (cond  ((=   n 1)  [1 1])
           (t  ($ *   [1 1]  (bin  (1 -   n))))))
```

3.3. Value of a polynomial Value of a polynomial $P(x) = a_1x^{n-1} + \ldots + a_n$ could be obtained as the internal product of vectors $[a_1 \ldots a_n]$ and $[x^{n-1} \ldots 1]$, i.e. as the value of the spectral expression:

($efval ($ * $[a_1 \ldots a_n]$[1 x (* x x)...(* x...x)]) n)).

EXAMPLE 6. The value of the polynomial

$$141.235 * x^3 + 1234.025 * x^2 - 356 * x + 1235.279 \text{ for } x = 0.1$$

could be computed as follows:

```
> ($efval  ($ *   [123  - 4592  + 762 1134]
                 [1  23(*  23  23)(*  23  23  23)])
          4)
@ - 0.900000
```

REFERENCES

[1] K. Orlov, *Numerička spektarlna aritmetika i algebra*, 1973, Beograd.

[2] K. Orlov, *Nove računske operacije inspirisane teorijom matematičkih spektara*, Matematički vesnik, 5(20)(1968), 393-398.

[3] J. Madić, *Division of mathematical spectra and its applications*, VI Conference on Applied Mathematic, 1988, 113-120.

[4] B. Mihalović, *The programme for multiplication of two spectra in FORTRAN IV and its applications on the solution of Numerical Algebraic and Differential Equations,* Matematički vesnik, 6(21)(1969), 151-155.

[5] B. Mihajlović, *Realizacija proizvoda dva spektra na cifarskim računskim mašinama,* Matematički vesnik, 5(20)(1968), 303-306.

[6] B. Mihajlovć *O prvim radovima M. Petrovića koji se odnose na primenu spektralne metode u algebri i aritmetici iz 1917, 1918 i 1919 godine,* Matematički vesnik, Beograd, 1968, 399-403.

[7] J. Madic, *Prilog teoriji i primeni matematičkih spektara* , Doktorska teza, Filozofski fakultet, Niš, 1987.

[8] P. Stanimirović, *Implementacija LISP interpretatora u TURBO PASCALU* , Magistarski rad, Filozofski fakultet, Niš, 1989.

Department of Mathematics
Philosophical Faculty
18000, Niš, Cirila i metoduja 2
Yugoslavia.