ГОДИШНИК НА СОФИЙСКИЯ УНИВЕРСИТЕТ "СВ. КЛИМЕНТ ОХРИДСКИ"
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Книга 1 — Математика
Том 82, 1988

ANNUAIRE DE L'UNIVERSITE DE SOFIA "ST. KLIMENT OHRIDSKI"
FACULTE DE MATHEMATIQUES ET INFORMATIQUE
Livre 1 — Mathématiques
Tome 82, 1988

ON THE REDUCTION OF POLYADIC RECURSIVE PROGRAMS TO MONADIC ONES*

DIMITER SKORDEV

Димитр Скордев. О СВЕДЕНИИ ПОЛИАДИЧЕСКИХ РЕКУРСИВНЫХ ПРОГРАММ К МОНАДИЧЕСКИМ. В работе рассматривается один метод сведения полиадических рекурсивных программ к монадическим. Он основан на идеях, использованных Л. Ивановым при некоторых применениях его алгебраической теории рекурсии. Обсуждается применимость этого метода для реализации полиадических рекурсивных программ в языках программирования, допускающих только рекурсивные процедуры без параметров.

Dimiter Skordev. ON THE REDUCTION OF POLYADIC RECURSIVE PROGRAMS TO MONADIC ONES. In the paper, a certain method is considered for the reduction of polyadic recursive programs to monadic ones. The method is based on ideas used before by L. Ivanov in some applications of his algebraic recursion theory. The applicability of this method is discussed for the implementation of polyadic recursive programs in programming languages admitting only recursive procedures without parameters.

The recursive programs considered in this paper are, roughly speaking, interpreted recursive program schemes (for the last notion, cf. e.g. [1, 2]). The monadic recursive programs can be described as recursive programs which contain only unary functional and predicate symbols and satisfy the condition that branching in them is controlled only by primitive predicates (compare with [1] pp. 7–4, 8–1). The polyadic recursive programs are those recursive programs which are not monadic. Their reduction to monadic ones can be used for their implementation in programming languages admitting only recursive procedures without parameters and hence

*An invited talk held at the Summer School and Conference on Mathematical Logic and its Applications "Heyting '88" (Varna, September 13–23, 1988). Research partially supported by the Ministry of Culture, Science and Education, Contract No. 247/1987.

not permitting to implement polyadic recursive programs in a straightforward manner (such languages are, for example, Basic and Forth, where in the case of the first of them its GOSUB-feature can be used). In the present paper, a method of reduction will be exposed which is based on ideas used before by L. Ivanov in some applications of his algebraic recursion theory [3, 4]. In a slightly different form, and without considerations about the practical implementation, this method has been exposed in [5].

Let $\mathcal{A} = \langle A;\ F_1,\ F_2,\ \ldots;\ P_1,\ P_2,\ \ldots\rangle$ be a (possibly partial) algebraic system.

D e f i n i t i o n 1 (inductive definition of the notion of an *$\mathcal{A}$-quasi-termal partial mapping* of A^m into A^n):

(1.1) $F_1,\ F_2,\ \ldots$ and the projection functions $\mathbf{pr}_{m,j}(a_1,\ \ldots,\ a_m) = a_j$, $m = 1,\ 2, \ldots,\ j = 1,\ \ldots,\ m$, are $\mathcal{A}$-quasi-termal.

(1.2) If $F : A^m - \to A^n$ and $G : A^l - \to A^m$ are $\mathcal{A}$-quasi-termal, then so is $F \circ G : A^l - \to A^n$ defined by

$$F \circ G(\bar{a}) \simeq F(G(\bar{a})).$$

(1.3) If $F : A^m - \to A^n$ and $G : A^m - \to A^p$ are $\mathcal{A}$-quasi-termal, then so is $F \times G : A^m \longrightarrow A^{n+p}$ defined by

$$F \times G(\bar{a}) \simeq F(\bar{a}).G(\bar{a})$$

(the multiplication sign in the right-hand side denotes concatenation).

(1.4) If P_i is l-ary, and $F : A^m - \to A^l$, $G,\ H : A^m - \to A^n$ are $\mathcal{A}$-quasi-termal, then so is $(P_i \circ F \Longrightarrow G,\ H) : A^m - \to A^n$ defined by

$$(P_i \circ F \Longrightarrow G,\ H)(\bar{a}) \simeq \begin{cases} G(\bar{a}) & \text{if} \quad P_i(F(\bar{a})) = \text{true}, \\ H(\bar{a}) & \text{if} \quad P_i(F(\bar{a})) = \text{false}. \end{cases}$$

D e f i n i t i o n 1′ (inductive definition of the notion of an *$\mathcal{A}$-quasi-termal operator*): a uniform relativized version of Definition 1.

In the case when $F_1,\ F_2,\ \ldots,\ P_1,\ P_2,\ \ldots$ are unary, we adopt also

D e f i n i t i o n 2 (inductive definition of the notion of a *monadic $\mathcal{A}$-quasi-termal mapping* of A into A):

(2.1) $F_1,\ F_2,\ \ldots$ and $\mathbf{id}_A = \mathbf{pr}_{1,1}$ are monadic $\mathcal{A}$-quasi-termal mappings.

(2.2) If F and G are monadic $\mathcal{A}$-quasi-termal mappings, then so is $F \circ G$.

(2.3) If G and H are monadic $\mathcal{A}$-quasi-termal mappings, then so is

$$(P_i \Longrightarrow G,\ H) = (P_i \circ \mathbf{id}_A \Longrightarrow G,\ H).$$

D e f i n i t i o n 2′ (inductive definition of the notion of a *monadic $\mathcal{A}$-quasi-termal operator*): a uniform relativized version of Definition 1′.

R e m a r k 1. The assumption that $F_1, F_2, \ldots,\ P_1,\ P_2,\ \ldots$ are unary does not always assure that all $\mathcal{A}$-quasi-termal mappings of A into A are monadic. Let

$\mathcal{A} = \langle \mathbf{N}; \mathbf{predec}; \mathbf{eqzero} \rangle$, where $\mathbf{N} = \{0, 1, 2, \ldots\}$, $\mathbf{predec}(a) = a - 1$, and $\mathbf{eqzero}(a)$ is equivalent to $(a = 0)$. Then the mapping $(\mathbf{eqzero} \circ \mathbf{predec} \Longrightarrow \mathbf{id_N}, \mathbf{predec})$ is not monadic (only the mappings $\lambda a.\ a - i$, $i = 0, 1, 2, \ldots$, are monadic).

We return again to the general case (i. e. $F_1, F_2, \ldots, P_1, P_2, \ldots$ are not necessarily unary).

D e f i n i t i o n 3. The set of all finite sequences of elements of A will be denoted by A^*.

D e f i n i t i o n 4. For any $F : A^m - \to A^n$, we define $F^* : A^* - \to A^*$ as follows:

$$F^*(\langle a_1, \ldots, a_s \rangle) \simeq \begin{cases} F(a_1, \ldots, a_m) \cdot \langle a_{m+1}, \ldots, a_s \rangle \text{ if } s \geq m, \\ \bot \quad \text{otherwise} \end{cases}$$

(this construction has been used in [6]).

D e f i n i t i o n 5. For any partial m-ary predicate P on A, we define a partial unary predicate P^* on A^* as follows:

$$P^*(\langle a_1, \ldots, a_s \rangle) \simeq \begin{cases} P(a_1, \ldots, a_m) \text{ if } s \geq m, \\ \bot \quad \text{otherwise.} \end{cases}$$

D e f i n i t i o n 6. The partial mappings $\mathbf{drop}$, $\mathbf{dup}$, $\mathbf{roll}_1$, $\mathbf{roll}_2$, ... of A^* into A^* are defined as follows:

$$\mathbf{drop}(\langle a_1, \ldots, a_s \rangle) \simeq \begin{cases} \langle a_2, \ldots, a_s \rangle \text{ if } s > 0, \\ \bot \quad \text{otherwise;} \end{cases}$$

$$\mathbf{dup}(\langle a_1, \ldots, a_s \rangle) \simeq \begin{cases} \langle a_1, a_1, \ldots, a_s \rangle \text{ if } s > 0, \\ \bot \quad \text{otherwise;} \end{cases}$$

$$\mathbf{roll}_n(\langle a_1, \ldots, a_s \rangle) \simeq \begin{cases} \langle a_{n+1}, a_1, \ldots, a_n, a_{n+2}, \ldots, a_s \rangle \text{ if } s > n, \\ \bot \quad \text{otherwise} \end{cases}$$

(these denotations are borrowed, mutatis mutandis, from the programming language Forth).

D e f i n i t i o n 7. Let $\mathcal{A}^*$ be the unary partial structure defined by

$$\mathcal{A}^* = \langle A^*; F_1^*, F_2^*, \ldots, \mathbf{drop}, \mathbf{dup}, \mathbf{roll}_1, \mathbf{roll}_2, \ldots; P_1^*, P_2^*, \ldots \rangle.$$

T h e o r e m 1. Whenever F is an $\mathcal{A}$-quasi-termal partial mapping of A^m into A^n, then F^* is a monadic $\mathcal{A}^*$-quasi-termal partial mapping of A^* into A^*.

T h e o r e m 1′. If Γ is an $\mathcal{A}$-quasi-termal operator then there is a monadic $\mathcal{A}^*$-quasi-termal operator Γ^* having the same number of arguments and satisfying the equality

$$\Gamma^*(G_1^*, \ldots, G_k^*) = (\Gamma(G_1, \ldots, G_k))^*$$

for all sequences $G_1, \ldots, G_k$ belonging to $\mathbf{dom}\ \Gamma$.

Proof (of both theorems). We use the equalities

$$\begin{aligned}(\mathbf{pr}_{m,j})^\bullet &= (\mathbf{drop} \circ \mathbf{roll}_1)^{m-j} \circ (\mathbf{drop})^{j-1},\\ (\mathbf{pr}_{1,1})^\bullet &= \mathbf{drop} \circ \mathbf{dup},\\ (F \circ G)^\bullet &= F^\bullet \circ G^\bullet,\end{aligned}$$

the fact that, whenever $F : A^m \dashrightarrow A^n$ and $G : A^m \dashrightarrow A^p$, then

$$(F \times G)^\bullet = \begin{cases} F^\bullet \circ G^\bullet & \text{if } m = 0,\\ F^\bullet \circ (\mathbf{roll}_{p+m-1})^m \circ G^\bullet \circ \mathbf{copy}_m & \text{otherwise,}\end{cases}$$

where $\mathbf{copy}_m = (\mathbf{pick}_{m-1})^m$ if $m > 1$, $\mathbf{copy}_1 = \mathbf{dup}$, $\mathbf{copy}_0 = \mathbf{id}_{A^*}$, and $\mathbf{pick}_i = \mathbf{roll}_i \circ (\mathbf{roll}_{i+1})^i \circ \mathbf{dup} \circ \mathbf{roll}_i$, as well as the fact that, whenever P_i is l-ary, and $F : A^m \dashrightarrow A^l$, $G, H : A^m \dashrightarrow A^n$, then

$$(P_i \circ F \Longrightarrow G,\ H)^\bullet = (P_i^\bullet \Longrightarrow G^\bullet \circ (\mathbf{drop})^l,\ H^\bullet \circ (\mathbf{drop})^l) \circ F^\bullet \circ \mathbf{copy}_m.$$

If we modify the definition of $\mathcal{A}^\bullet$ by adding suitable other primitive operations, then the above proof may be replaced by a simpler one. For example, in the case of Theorem 1′ we could add all monadic mappings quasi-termal with respect to the original $\mathcal{A}^\bullet$ as additional operations, and we would have yet a non-tautological result. When applying the construction from the proof to concrete operators, sometimes also the addition of certain predicates of the form $P_i \circ \Phi$ with monadic Φ could happen to be convenient.

A certain kind of calculus can be developed for the construction of the operators $\Gamma^\bullet$. Let us adopt the convention that, when given an expression intended to denote some element of some set A^n, the same expression will denote also the corresponding mapping of A^0 into A^n. Then, given an $\mathcal{A}$-quasi-termal operator Γ having k arguments and producing m-ary functions, we form the corresponding expression $U^\bullet$, where U is an expression for $\Gamma(g_1, \ldots, g_k)(x_1, \ldots, x_m)$, assuming $g_1, \ldots, g_k$ to be variables for functions of the suitable types, and $x_1, \ldots, x_m$ to be variables for elements of A. The next step is to transform $U^\bullet$ into an expression of the form $V \circ x_1^\bullet \circ \ldots \circ x_m^\bullet$, where V does not contain $x_1^\bullet, \ldots, x_m^\bullet$, and V is obtained from $g_1^\bullet, \ldots, g_k^\bullet$, from denotations of monadic mappings and from predicates $P_i \circ \Phi$ by means of composition and branching (this transformation can be done by using equalities from the proof of Theorems 1, 1′ and some other equalities, e. g. $\mathbf{dup} \circ F^\bullet = F^\bullet \circ F^\bullet$ for each $F : A^0 \dashrightarrow A$). Taking the expression V and replacing the occurrences of $g_1^\bullet, \ldots, g_k^\bullet$ in it by variables $\gamma a_1, \ldots, \gamma a_k$ for unary operations in A^*, we obtain an expression for $\Gamma^\bullet(\gamma a_1, \ldots, \gamma a_k)$.

E x a m p l e 1. Let $\mathcal{A}$ be the same as in Remark 1. Let U be the expression

$$g(g(\mathbf{predec}(y),\ z),\ g(\mathbf{predec}(x),\ z)),$$

where g is a variable for functions of type $\mathbf{N}^2 \dashrightarrow \mathbf{N}$, and x, y, z are variables for elements of $\mathbf{N}$. Then

$$\begin{aligned}U^\bullet &= g^\bullet \circ (g(\mathbf{predec}(y),\ z))^\bullet \circ (g(\mathbf{predec}(x),\ z))^\bullet\\ &= g^\bullet \circ g^\bullet \circ (\mathbf{predec}(y))^\bullet \circ z^\bullet \circ (g(\mathbf{predec}(x),\ z))^\bullet\end{aligned}$$

$$= g^* \circ g^* \circ \mathbf{roll}_2^{-1} \circ (g(\mathbf{predec}(x),\ z))^* \circ (\mathbf{predec}(y))^* \circ z^*$$

$$= g^* \circ g^* \circ \mathbf{roll}_2^{-1} \circ g^* \circ (\mathbf{predec}(x))^* \circ z^* \circ (\mathbf{predec}(y))^* \circ z^*$$

$$= g^* \circ g^* \circ \mathbf{roll}_2^{-1} \circ g^* \circ F^* \circ x^* \circ y^* \circ z^*,$$

where $\mathbf{roll}_2^{-1}$ is the inverse mapping of $\mathbf{roll}_2$, and the mapping F of type $\mathbf{N}^3 \to \mathbf{N}^4$ is defined by

$$F(x,\ y,\ z) = \langle \mathbf{predec}(x),\ z,\ \mathbf{predec}(y),\ z \rangle.$$

Hence, taking γ to be a variable for unary operations in A^*, we may set

$$\Gamma^*(\gamma) = \gamma \circ \gamma \circ \mathbf{roll}_2^{-1} \circ \gamma \circ F^*.$$

D e f i n i t i o n 8. A *recursive program over* $\mathcal{A}$ is a system of equations of the form

$$g_i = \Gamma a_i(g_1,\ \ldots,\ g_k), \qquad i = 1,\ \ldots,\ k,$$

where $\Gamma a_1,\ \ldots,\ \Gamma a_k$ are $\mathcal{A}$-quasi-termal operators. The above program is called *monadic* if $g_1,\ \ldots,\ g_k$ are variables for unary functions, and the operators $\Gamma a_1,\ \ldots,\ \Gamma a_k$ are monadic. *The functions computed by the program* are the components of the least solution of the system.

R e m a r k 2. Some authors use the term "monadic program" in a sense which is not the same as in the above definition. Cf., for example, [7, p. 544].

D e f i n i t i o n 9. Given a recursive program over $\mathcal{A}$ having the form from Definition 8, *the monadic transform* of this program is the monadic recursive program

$$\gamma a_i = \Gamma a_i^*(\gamma a_1,\ \ldots,\ \gamma a_k), \qquad i = 1,\ \ldots,\ k;$$

over $\mathcal{A}^*$, where $\gamma a_1,\ \ldots,\ \gamma a_k$ are variables for unary operations in A^*.

Using the continuity of the $\mathcal{A}$-quasi-termal operators and of the operation $\lambda F,\ F^*$, one easily proves.

T h e o r e m 2. If $G_1,\ \ldots,\ G_k$ are the functions computed by a given recursive program over $\mathcal{A}$, then $G_1^*,\ \ldots,\ G_k^*$ are the functions computed by its monadic transform.

Since

$$G_i(a_1,\ \ldots,\ a_{m_i}) = G_i^*(\langle a_1,\ \ldots,\ a_{m_i} \rangle),$$

where m_i is the arity of G_i, Theorem 2 shows a way for the reduction of recursive programs over $\mathcal{A}$ to monadic recursive programs over $\mathcal{A}^*$.

E x a m p l e 2. Let $\mathcal{A} = \langle \mathbf{N};\ \mathbf{predec},\ C,\ L, R;\ \mathbf{eqzero} \rangle$, where

$$C(x,\ y) = \frac{1}{2}(x+y)(x+y+1) + x,$$

$$L(C(x,\ y)) = x, \qquad R(C(x,\ y)) = y$$

for all $x,\ y$ in $\mathbf{N}$. Consider the two-argument partial recursive function ω defined recursively by

$$\omega(C(x,\ y),\ z) \simeq \begin{cases} C(x,\ z) & \text{if } y = 0, \\ y - 1 & \text{if } y > 0,\ x = 0, \\ \omega(\omega(y-1,\ z),\ \omega(x-1,\ z)) & \text{otherwise} \end{cases}$$

(according to [8], ω is an universal function for the unary partial recursive functions). The above recursive definition of ω can be represented by a recursive program in the sense of Definition 8 such that ω will be the function computed by that recursive program. We could, for example, take the recursive program $g = \Delta(g) \circ G$, where g is a variable for two-argument partial functions in $\mathbf{N}$, the operator Δ transforms such functions into three-argument partial functions in $\mathbf{N}$, G is a mapping of $\mathbf{N}^2$ into $\mathbf{N}^3$, and the following defining equalities hold:

$$\Delta(g)(x,\ y,\ z) \simeq \begin{cases} C(x,\ z) & \text{if } y = 0, \\ y - 1 & \text{if } y > 0,\ x = 0, \\ g(g(y-1,\ z),\ g(x-1,\ z)) & \text{otherwise,} \end{cases}$$

$$G(t,\ z) = \langle L(t),\ R(t),\ z \rangle.$$

Since $(\Delta(g) \circ G)^\bullet = (\Delta(g))^\bullet \circ G^\bullet = \Delta^\bullet(g^\bullet) \circ G^\bullet$, the monadic transform of the above recursive program can be written as $\gamma = \Delta^\bullet(\gamma) \circ G^\bullet$. For constructing an explicit expression for $\Delta^\bullet(\gamma)$, we consider the expression $U^\bullet$, where U is the expression $\Delta(g)(x,\ y,\ z)$. We have the equality

$$U^\bullet = (\mathtt{eqzero}^\bullet \circ y^\bullet \Longrightarrow (C(x,\ z))^\bullet,\ (\mathtt{eqzero}^\bullet \circ x^\bullet \Longrightarrow (\mathtt{predec}(y))^\bullet,\ (g(g(\mathtt{predec}(y),\ z),\ g(\mathtt{predec}(x),\ z)))^\bullet)).$$

Making use of Example 1, we get

$$\begin{aligned} U^\bullet &= (\mathtt{eqzero}^\bullet \circ y^\bullet \Longrightarrow C^\bullet \circ x^\bullet \circ z^\bullet,\ (\mathtt{eqzero}^\bullet \circ x^\bullet \Longrightarrow \mathtt{predec}^\bullet \circ y^\bullet, \\ &\qquad g^\bullet \circ g^\bullet \circ \mathtt{roll}_2^{-1} \circ g^\bullet \circ F^\bullet \circ x^\bullet \circ y^\bullet \circ z^\bullet)) \\ &= (\mathtt{eqzero}^\bullet \circ \mathtt{drop} \Longrightarrow C^\bullet \circ \mathtt{drop} \circ \mathtt{roll}_1, \mathtt{eqzero}^\bullet \Longrightarrow \mathtt{predec}^\bullet \circ \mathtt{drop}, \\ &\qquad g^\bullet \circ g^\bullet \circ \mathtt{roll}_2^{-1} \circ g^\bullet \circ F^\bullet)) \circ x^\bullet \circ y^\bullet \circ z^\bullet. \end{aligned}$$

Hence

$$\begin{aligned} \Delta^\bullet(\gamma) = (\mathtt{eqzero}^\bullet \circ \mathtt{drop} \Longrightarrow\ & C^\bullet \circ \mathtt{drop} \circ \mathtt{roll}_1, (\mathtt{eqzero}^\bullet \Longrightarrow \\ & \mathtt{predec}^\bullet \circ \mathtt{drop}, \gamma \circ \gamma \circ \mathtt{roll}_2^{-1} \circ \gamma \circ F^\bullet)). \end{aligned}$$

So we have found the explicit form of the monadic transform of the original recursive program computing ω. According to Theorem 2, this monadic transform computes the function $\omega^\bullet$, and therefore could be used for computing values of ω.

Of course, a recursive program in Pascal for computing values of ω can be written in a straightforward manner. But the monadic transform constructed above gives us the possibility to construct relatively simple programs computing values of ω in programming languages holding out much more restricted possibilities for a straightforward recursive programming. To demonstrate this, we wrote a program in Applesoft.Basic for the some purpose, using the obtained monadic transform and the GOSUB-feature of Basic. The length of the program is less than 1000 bytes. Sequences from $\mathbf{N}^*$ are represented on a stack principle, the members of the sequence following each other in direction from top to bottom. The stack is realized by means of an array A supplied with a counter D, and the dimension of A is chosen in concordance with the restriction in Applesoft.Basic about the number of the nested subroutine calls. The flow diagram of the program and the way of

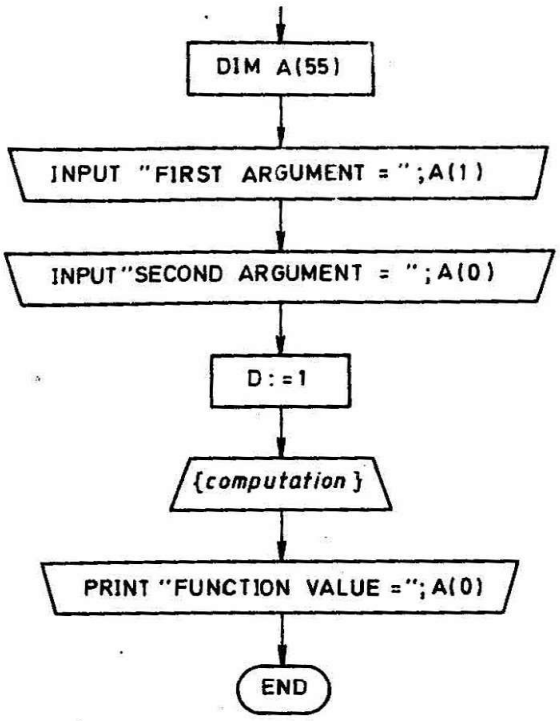

Fig. 1. Flow diagram of the program in Basic for computing ω

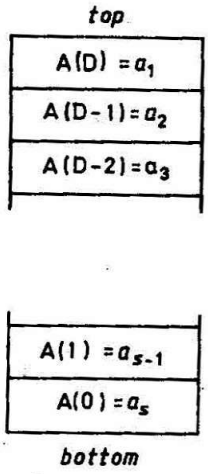

Fig. 2: The stack representation of the sequence $\langle a_1, \ldots a_s \rangle$

using the array **A** for the stack representation of elements of $\mathbf{N}^*$ are shown on Fig. 1, 2. An essential part of the program is a subroutine called "*computation*" which contains three calls of itself. The flow diagram of this subroutine is shown on Fig. 3. As to the efficiency of the considered program, the situation is not so bad as somebody could suspect. Namely, some experiments were made by the author on a "Pravetz-8M" microcomputer, and they showed the compiled variant of the program running faster then the directly written recursive Pascal program processed in the Apple UCSD Pascal system. For obtaining more information about the efficiency of using monadic transforms, the author wrote also a monadic recursive and a non-recursive program in Apple Pascal for computing values of ω

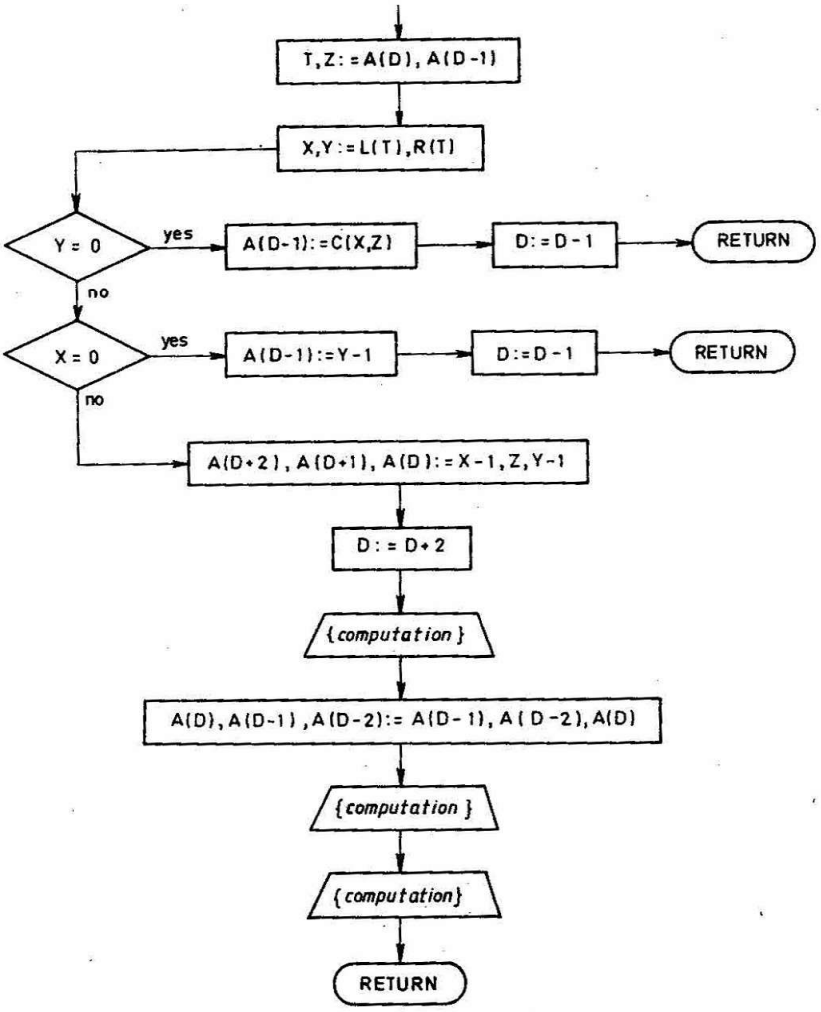

Fig. 3. Flow diagram of the subroutine "*computation*"

using again the monadic transform of the original recursive program. The table below contains some average ratios of run-times (the timing-experiments were based on computing $\omega(4,\ 0)$, $\omega(11,\ 1)$, $\omega(11,\ 3)$, $\omega(41,\ 50)$, $\omega(60,\ 61)$ and $\omega(312,\ 3)$:

Applesoft.Basic		Apple UCSD Pascal		
interpreted	compiled	polyadic recursive	monadic recursive	non-recursive
1.94	0.86	1	1.06	1.10

Acknowledgments. I am grateful to the organizers of "Heyting '88" for giving me the opportunity to deliver this talk. Thanks are due to my son Gencho who instructed me in the programming for "Pravetz-8M", as well as to my colleagues Ivan Soskov and Tinko Tinchev who helped me in preparing and carrying out a demonstration on an IBM PC computer. I thank also Professor V.A. Nepomniaschy for some useful discussions (in particular, for attracting my attention to the paper [7] and to other papers by the same author).

BIBLIOGRAPHY

1. Greibach, S. A. Theory of Program Structures: Schemes, Semantics, Verification. — Lecture Notes in Computer Science, **36**, Berlin—Heidelberg—New York, 1975.
2. Котов, В. Е. Введение в теорию схем программ. Новосибирск, 1978.
3. Иванов, Л. Л. Итеративни операторни пространства.С., 1980 (канд. дисертация).
4. Ivanov, L. L. Algebraic Recursion Theory. Chichester, 1986.
5. Skordev, D. G. A reduction of polyadic recursive programs to monadic ones. — In: Symposium on Math. Foundations of Computer Science (Diedrichshagen, December 6–11, 1982), Seminarbericht Nr. 52, Sektion Math. der Humboldt—Univ. zu Berlin, Berlin, 1983, 124–132.
6. Скордев, Д. Г. Об одном погружении итеративных алгебр Поста в полугруппы. — Алгебра и логика, **21**, № 2, 1982, 228–241.
7. Langmaack, H. On a theory of decision problems in programming languages. — In: Mathematical Studies in Information Processing (edited by E.K. Blum, M. Paul and S. Takasu). — Lecture Notes in Computer Science, **75**, Berlin—Heidelberg—New York, 1979, 538–558.
8. Скордев, Д. Г. Некоторые простые примеры универсальных функций. — ДАН СССР, **190**, № 1, 1970, 45–46.

Received 14.IV.1989