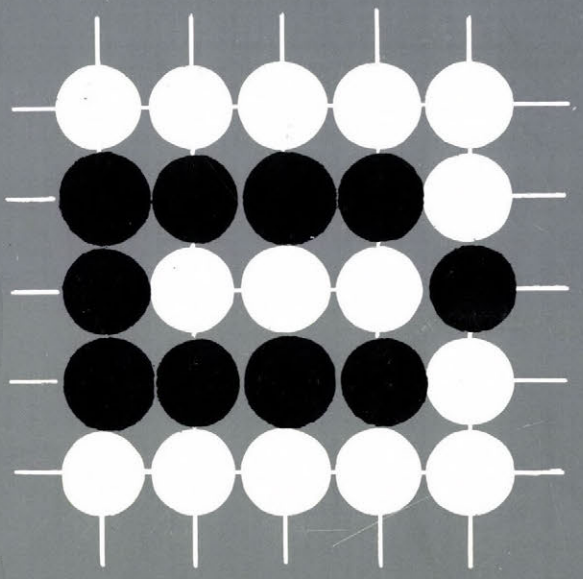
közlemények 31/1984

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest



MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

K Ö Z L E M É N Y E K

31/1984

Felelős kiadó:

Dr. VÁMOS TIBOR

ISBN 963 311 185 4
ISSN 0133-7459
SZÁMALK Repro 85/096

TARTALOMJEGYZÉK

C O N T E N T S

С О Д Е Р Ж А Н И Е

Közlemények 31/1984 7-21

FINITELY GENERATED CLONES WITH LINEAR FUNCTIONS IN P_3 AND P_5

M. CSIKÓS

University of Agricultural Sciences

INTRODUCTION

Let E_k be the set $\{0,1,\ldots,k-1\}$ for $k \geq 2$, $P_k^{(n)} = \{f \mid f: E_k^n \to E_k\}$ for $n = 1,2,\ldots$ and let $P_k = \bigcup_{n=0}^{\infty} P_k^{(n)}$, where $P_k^{(0)}$ is the set of constant functions. A set of functions $Z \subset P_k$ is a clone if it contains the projections (i.e. the functions $e_j(x_1 \ldots x_j \ldots x_n) = x_j$, $j=1,2,\ldots,n$) and all superpositions over Z.

An open problem is the following: under what conditions is an arbitrary $Z \subset P_k$ finitely generated? (Z is finitely generated if there exists a finite subset $Z_n \subset Z$ from which all functions of Z can be obtained by superpositions.)

It is known that the clone of the linear functions L_p in P_p (p is a prime) is finitely generated (Demetrovics and Bagyinszki [1]), where $L_p = \{L \mid L(x_1,\ldots,x_n) = a_0 + \sum_{i=1}^{n} a_i x_i,$ $n = 1,2\ldots\}$ (addition and multiplication are carried out mod p and a_i are residue classes mod p).

We deal with finitely generated clones of P_3 and P_5. The purpose of this paper is to prove the following theorem:

A clone Z of P_3 or P_5 is finitely generated if it contains a nontrivial n-ary linear function ($n \geq 2$) and an unary non linear function.

For P_3 a more general result was proved by Marcenkov [2]: $Z \subset P_3$ is finitely generated if it contains an n-ary linear function and an arbitrary non linear function. (Lemma 5 and its corollary). We give another proof, our method works for P_5 too (this part of the theorem is a new result).

The following statements will be useful in the sequel. If

$Z \subset P_3$ ($Z \subset P_5$) contains n-ary linear function then it contains also the function $f(x,y) = 2x + 2y$ ($F(x,y) = 2x + 4y$) [1].

If Z has a near-unanimity function then it is finitely generated (this is an immediate corollary of the results of Baker's-Pixley's [3]). The function $m: E_k^n \to E_k$ is a near-unanimity function if $m(y,x,\ldots,x) = m(x,y,x,\ldots,x) = \ldots = m(x,x,\ldots,x,y) = x$ for all $x,y \in E_k$.

For the proof of the P_3-part of the theorem is sufficient to construct with superpositions from the function $f(x,y)$ and from an arbitrary non linear unary function a three-variable near-unanimity function $M: E_3^3 \to E_3$, for which $M(x,x,y) = M(x,y,x) = M(y,x,x) = X$ hold. Moreover it is sufficient to construct the function $\wedge_o(x,y)$ and $\vee_o(x,y)$ from which one can obtain the function M using the formula of [2]:

$$M(x,y,z) = \vee_o(\vee_o(\wedge_o(x,y), \wedge_o(x,z)), \wedge_o(y,z)) \qquad (1)$$

where the Cayley tables of $\wedge_o$ and $\vee_o$:

$\wedge_o(x,y)$

x \ y	0	1	2
0	0	0	0
1	0	1	0
2	0	0	2

$\vee_o(x,y)$

x \ y	0	1	2
0	0	1	2
1	1	1	0
2	2	0	2

PROOF FOR FIRST PART OF THE THEOREM

First we construct the functions $\wedge_o$ and $\vee_o$. The unary non linear functions of P_3 can be given by the table:

x	0	1	2
$a(x)$	0	0	1
$b(x)$	0	1	0
$c(x)$	1	0	0
$d(x)$	0	0	2
$e(x)$	0	2	0
$\varphi(x)$	2	0	0
$g(x)$	0	1	1
$h(x)$	1	0	1
$i(x)$	1	1	0
$j(x)$	0	2	2
$k(x)$	2	0	2
$l(x)$	2	2	0
$m(x)$	1	1	2
$n(x)$	1	2	1
$o(x)$	2	1	1
$p(x)$	1	2	2
$q(x)$	2	1	2
$r(x)$	2	2	1

Among these functions b, d, g, j, m and q are isomorphic in the sense that all of them fix two elements and to the third they assign one of the fixed elements. Now we shall obtain $\wedge_o$ and $\vee_o$ from the functions $f(x,y)$ and $b(x)$. The function $b(x)$ is in the clone generated by either $a(x)$ or $c(x)$. In the groups of the functions d, g, j, m, and q (in the table these groups are separated with lines) similar computations can be carried out. E.g. from q can be produced (with the same steps applicated for b) the functions $\wedge_2(x,y)$

0	2	2
2	1	2
2	2	2

and

$V_2(x,y)$ $\begin{vmatrix} 0 & 2 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 2 \end{vmatrix}$ from which the function M is formed according to (1).

THE COMPUTATION FOR b(x)

x	0	1	2
$b(x)$	0	1	0

$f(x,y)$ $\begin{vmatrix} 0 & 2 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \end{vmatrix}$ $b(f(x,y))$ $\begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{vmatrix}$

$B_1(x,y) = b(f(b(x),y))$ $\begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$

$B_2(x,y) = b(f(x,b(y)))$ $\begin{vmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix}$

$f(B_1(x,y),B_2(x,y))$ $\begin{vmatrix} 0 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{vmatrix}$

$$\wedge_o(x,y) = f(f(B_1,B_2),b(f(x,y)))$$

$$\vee_o(x,y) = f(\wedge_o(x,y),f(x,y))$$

THE REDUCTION OF THE FUNCTIONS a(x) AND c(x) TO b(x)

$a(a(x)) = 0$; $a(f(x,0)) = b(x)$;

$f(c(f(x,y),\ c(c(f(x,y))))) = 2$; $c(f(x,2)) = b(x)$ and because of the isomorphisms stated the assertion is proved.

THE SECOND PART OF THE THEOREM CONCERNING P_5

The proof is similar to the previous case. We shall obtain from the function $F(x,y)$ and from an arbitrary non linear unary function of P_5 the functions

$\wedge_o(x,y)$

0	0	0	0	0
0	1	0	0	0
0	0	2	0	0
0	0	0	3	0
0	0	0	0	4

and $\vee_o(x,y)$

0	1	2	3	4
1	1	0	0	0
2	0	2	0	0
3	0	0	3	0
4	0	0	0	4

From these functions, using (1), we get a three variable near--unanimity function M of P_5.
The computation gives also the polynomial forms of the functions $\wedge_o$ and $\vee_o$.
First we derive from F two other linear functions

$$F(2x + 4y, y) = 2(2x + 4y) + 4y = 4x + 2y$$

$$F(4x + 2y, y) = 2(4x + 2y) + 4y = 3x + 3y.$$

If the functions G, H, K are elements of the clone Z then the same is true for the following linear combinations of them:

$$4G + 4H + 3K = 3(3G + 3H) + 3K$$

$$G + H + 4K = 4(4G + 4H + 3K) + 2K$$

$$G + 2H + 3K = 3(2G + 4H) + 3K$$

$$2G + 2H + 2K = 4(3G + 3K) + 2K .$$

The proof is also a reduction to such "good" functions as $b(x)$ was in P_3. In P_5, there are two kinds of such unary non linear functions:

$k(x) = 0\ 1\ 3\ 2\ 4$ (it fixes three values of x and to the fifth assigns one of the fixed values) $l_4(x) = 01233$ (it fixes four values of x and to the fifth assigns one of the fixed values.

THE CONSTRUCTION OF THE FUNCTIONS $\wedge_o$ AND $\vee_o$ FROM F(x,y) AND k(x)

x	0	1	2	3	4
$k(x)$	0	1	3	2	4

(The polynomial form of $k(x)$ is x^3).

Let the function F_1 be $F_1(x,y) = k(x) + k(y) + 4k(3x + 3y)$

$F_1(x,y)$

0	4	2	3	1
4	1	0	0	0
2	0	3	0	0
3	0	0	2	0
1	0	0	0	4

The polynomial form of $F_1(x,y)$ is $4(x^3 + x^2y + xy^2 + y^3)$. The table of $3k(x) + 3k(y)$ is:

0	3	4	1	2
3	1	2	4	0
4	2	3	0	1
1	4	0	2	3
2	0	1	3	4

Using these functions

$$\wedge_o(x,y) = F_1\,(k(3x + 3y),\ 3k(x) + 3k(y))$$

The polynomial form of $\wedge_o$ is $4(x^4y + x^3y^2 + x^2y^3 + xy^4)$. To construct $\vee_o$ it is necessary to take the cube of the function $F_1(x,y)$. $F_2(x,y) = k(F_1(x,y))$. Its table is

0	4	3	2	1
4	1	0	0	0
3	0	2	0	0
2	0	0	3	0
1	0	0	0	4

With this function

$$\vee_o(x,y) = F_2(\wedge_o(x,y), F_2(x,y))$$

The polynomial form of $\vee_o$ is $3x^4y + 4x^3y^2 + 4x^2y^3 + 3xy^4 + x + y$. Using the same iteration steps one can derive also the unanimity function $M(x,y,z)$ from the other 9 such functions of P_5 which fix three values of x and interchange the another two values.

(E.g. from $u(x) = 0\ 1\ 4\ 3\ 2$ can be produced in the way pre-

sented for $k(x)$ the functions

$\wedge_3(x,y)$

0	3	3	3	3
3	1	3	3	3
3	3	2	3	3
3	3	3	3	3
3	3	3	3	4

and $\vee_3(x,y)$

0	3	3	0	3
3	1	3	1	3
3	3	2	2	3
0	1	2	3	4
3	3	3	4	4

from which the function M is formed using (1)).

THE CONSTRUCTION FROM THE FUNCTIONS F(x,y) AND $1_4(x)$

x	0	1	2	3	4
$1_4(x)$	0	1	2	3	3

We need a lot of superpositions since the function $1(x)$ make only a little change on x.

$F(x,y) = 2x + 4y$

0	4	3	2	1
2	1	0	4	3
4	3	2	1	0
1	0	4	3	2
3	2	1	0	4

$1_4(F(x,y) =$

0	3	3	2	1
2	1	0	3	3
3	3	2	1	0
1	0	3	3	2
3	2	1	0	3

$H(x,y) = 3x + 3y$

0	3	1	4	2
3	1	4	2	0
1	4	2	0	3
4	2	0	3	1
2	0	3	1	4

$l_4(H(x,y))$

0	3	1	3	2
3	1	3	2	0
1	3	2	0	3
3	-2	0	3	1
2	0	3	1	3

$A = F(F, l_4F)$

0	0	3	2	1
2	1	0	0	3
0	3	2	1	0
1	0	0	3	2
3	2	1	0	0

$B = F(H, l_4H)$

0	3	1	0	2
3	1	0	2	0
1	0	2	0	3
0	2	0	3	1
2	0	3	1	0

$C = A(A, B)$

0	2	0	0	0
1	1	0	3	1
0	1	2	2	2
2	3	0	3	3
0	0	0	0	0

$D = A(B, A)$

0	1	0	3	3
0	1	0	0	2
2	2	2	0	1
0	0	0	3	0
1	3	0	2	0

$E = H(C, D)$

0	4	0	4	4
3	1	0	4	4
1	4	2	1	4
1	4	0	3	4
3	4	0	1	0

$G = C(E, C)$

0	0	0	0	0
3	1	0	0	0
1	0	2	0	0
0	0	0	3	0
2	0	0	1	0

$J = G(G, C)$

0	0	0	0	0
0	1	0	0	0
3	0	2	0	0
0	0	0	3	0
1	0	0	3	0

$K = G(G, J)$

0	0	0	0	0
0	1	0	0	0
0	0	2	0	0
0	0	0	3	0
0	0	0	0	0

$L = F(K, F)$

0	1	2	3	4
3	1	0	1	2
1	2	2	4	0
4	0	1	3	3
2	3	4	0	1

$M = H(L, l_4H)$

0	2	4	3	3
3	1	4	4	1
1	0	2	2	4
1	1	3	3	2
2	4	1	3	2

$N = E(M, L)$

0	4	0	3	4
3	1	3	4	0
1	0	2	4	3
4	3	4	3	1
2	1	4	1	4

$O = F(H, N)$

0	2	2	0	0
3	1	0	0	0
1	3	2	1	3
4	1	1	3	1
2	4	2	1	4

$P = O(O,K)$

0	1	1	0	0
4	1	0	0	0
3	4	2	3	4
2	3	3	3	3
1	2	1	3	2

$Q = F(P,C)$

0	0	2	0	0
2	1	0	2	4
1	2	2	4	1
2	3	1	3	3
2	4	2	1	4

$R = E(K,Q)$

0	0	0	0	0
0	1	0	0	4
4	0	2	4	4
0	4	4	3	4
0	4	0	4	4

$S = R(R,L)$

0	0	0	0	0
0	1	0	0	0
4	0	2	4	0
0	0	4	3	4
0	4	0	0	4

$T = S(S,H)$

0	0	0	0	0
0	1	0	0	0
4	0	2	0	0
0	0	0	3	4
0	0	0	0	4

$\wedge_o = T(T,H)$

$U = 4x+4y+3\wedge_o$

0	4	3	2	1
4	1	2	1	0
3	2	2	0	4
2	1	0	3	3
1	0	4	3	4

$W = \wedge_o(U,F)$

0	4	3	2	1
0	1	0	0	0
0	0	2	0	0
0	0	0	3	0
0	0	0	0	4

$Z = \wedge_o(4x\ 2y, U)$

0	0	0	0	0
4	1	0	0	0
3	0	2	0	0
2	0	0	3	0
1	0	0	0	4

$\vee_o = 4W + 4Z + 3\wedge_o$.

For the following 4 functions (with 4 fixed values) the same computations can be carried out as for $l_4(x)$

x	0	1	2	3	4
$l_o(x)$	1	1	2	3	4
$l_1(x)$	0	2	2	3	4
$l_2(x)$	0	1	4	3	4
$l_3(x)$	0	1	2	4	4

E.g. from $l_o(x)$ can be produced in the manner applicated

for $l_4(x)$ the functions

$\wedge_4(x,y)$

0	4	4	4	4
4	1	4	4	4
4	4	2	4	4
4	4	4	3	4
4	4	4	4	4

and $\vee_4(x,y)$

0	4	4	4	0
4	1	4	4	1
4	4	2	4	2
4	4	4	3	3
0	1	2	3	4

From x and from an arbitrary other function g with 4 fixed values one can produce with an admissible linear combination those function $l_i(x)$ which has the same 4 fixed values as g. The proof will be complete if we show that from an arbitrary non linear unary function of P_5 and from $F(x,y)$ we can obtain a function which has one of the previous two properties of $k(x)$ and $l_4(x)$. We show this statement first for some types of functions.

a) If in the clone Z f and g two functions which differ from each other only at one value of x then $3f+2g$ is 0 for exactly 4 values of the argument, i.e. $3f + 2g + x$ has four fixed values.

b) Functions with 3 fixed values

Let f be a function which fixed 3 values of x and interchanges the other two values of x: y and z. (From f we can already derive near-unanimity function). Let g be $3f + 3x$. It fixes the same three values as f. Let $g(y) = g(z) = 3y + 3z = a_o$ which is different from y and from z. Next let h be $4g + 2x$. $h(y) = 4a_o + 2y = 2z + 4y$. Let $h(y) = a_1$ from the equations it follows that $a_1 \neq a_o$ $a_1 \neq y$, $a_1 \neq z$ $h(z) = 4a_o + 2z = 4z + 2y$. $h(z)$ differs from a_o, z, y too. The table of $F(x,y)$ shows that $2z + 4y \neq 4z + 2y$ i.e. $h(z) \neq a_1$. Let $h(z) = a_2$. Finally let j be $2g + 4x$.

x	a_o	a_1	a_2	y	z
$f(x)$	a_o	a_1	a_2	z	y
$g(x)$	a_o	a_1	a_2	a_o	a_o
$h(x)$	a_o	a_1	a_2	a_1	a_2
$j(x)$	a_o	a_1	a_2	a_2	a_1

From $g(x)$, $h(x)$ or from $j(x)$ one can derive the function f with the following linear combinations:

$$f(x) = 2g(x) + 4x \qquad f(x) = 3h(x) + 3x \qquad f(x) = 4j(x) + 2x.$$

Let $m_i(x) = a_0a_1a_2za_i$ $(i=0,1,2)$. For $i=0,1,2$, $m_i \circ m_i(x) = a_0a_1a_2a_ia_i$ and $m_i \circ m_i$ differs from m_i only at one value of x. The same is true for the functions $n_i(x) = a_0a_1a_2a_iy$ $(i=0,1,2)$.

The remaining 6 functions with the same three fixed values are $a_0a_1a_2a_0a_1$, $a_0a_1a_2a_0a_2$, $a_0a_1a_2a_1a_0$, $a_0a_1a_2a_1a_0$, $a_0a_1a_2a_1a_1$, $a_0a_1a_2a_2a_0$ and $a_0a_1a_2a_2a_2$. One can reduce them with linear combinations to the functions m_i and n_i.
(E.g. $3(a_0a_1a_2a_2a_2) + 3x = n_1(x)$).

To function with three fixed values can be reduced such a function b which differs from $b \circ b$ only at two values of x. The function $3b + 2(b \circ b)$ is 0 for exactly 3 values of x, i.e. the function $3b + 2(b \circ b) + x$ of the clone Z has three fixed values.

c) Functions with 2 fixed values

The following table shows the functions for which a_0 and a_1 are the two fixed values. One (two) point(s) behind the function f means that $f \circ f$ differs from f only at one (two) value(s) of x. (4) means that $f \circ f$ has 4 fixed values.

x $a_0a_1a_2a_3a_4$ f(x)	x $a_0a_1a_2a_3a_4$ f(x)	x $a_0a_1a_2a_3a_4$ f(x)	x $a_0a_1a_2a_3a_4$ f(x)
$a_0a_1a_0a_0a_0$	$a_0a_1a_1a_0a_0$	$a_0a_1a_3a_0a_0$.	$a_0a_1a_4a_0a_0$.
$a_0a_1a_0a_0a_1$	$a_0a_1a_1a_0a_1$	$a_0a_1a_3a_0a_1$.	$a_0a_1a_4a_0a_1$.
$a_0a_1a_0a_0a_2$.	$a_0a_1a_1a_0a_2$.	$a_0a_1a_3a_0a_2$..	$a_0a_1a_4a_0a_2$(4)
$a_0a_1a_0a_0a_3$.	$a_0a_1a_1a_0a_3$.	$a_0a_1a_3a_0a_3$..	$a_0a_1a_4a_0a_3$..

x $a_0a_1a_2a_3a_4$ f(x)	x $a_0a_1a_2a_3a_4$ f(x)	x $a_0a_1a_2a_3a_4$ f(x)	x $a_0a_1a_2a_3a_4$ f(x)
$a_0a_1a_0a_1a_0$	$a_0a_1a_1a_1a_0$	$a_0a_1a_3a_1a_0$·	$a_0a_1a_4a_1a_0$·
$a_0a_1a_0a_1a_1$	$a_0a_1a_1a_1a_1$	$a_0a_1a_3a_1a_1$·	$a_0a_1a_4a_1a_1$·
$a_0a_1a_0a_1a_2$·	$a_0a_1a_1a_1a_2$·	$a_0a_1a_3a_1a_2$··	$a_0a_1a_4a_1a_2$(4)
$a_0a_1a_0a_1a_3$·	$a_0a_1a_1a_1a_3$·	$a_0a_1a_3a_1a_3$··	$a_0a_1a_4a_1a_3$··
$a_0a_1a_0a_2a_0$·	$a_0a_1a_1a_2a_0$·	$a_0a_1a_3a_2a_0$(4)	$a_0a_1a_4a_2a_0$··
$a_0a_1a_0a_2a_1$·	$a_0a_1a_1a_2a_1$·	$a_0a_1a_3a_2a_1$(4)	$a_0a_1a_4a_2a_1$··
$a_0a_1a_0a_2a_2$··	$a_0a_1a_1a_2a_2$··	$a_0a_1a_3a_2a_2$(4)	$a_0a_1a_4a_2a_2$(4)
$a_0a_1a_0a_2a_3$··	$a_0a_1a_1a_2a_3$··	$a_0a_1a_3a_2a_3$(4)	q: $a_0a_1a_4a_2a_3$
$a_0a_1a_0a_4a_0$·	$a_0a_1a_1a_4a_0$·	$a_0a_1a_3a_4a_0$··	$a_0a_1a_4a_4a_0$··
$a_0a_1a_0a_4a_1$·	$a_0a_1a_1a_4a_1$·	$a_0a_1a_3a_4a_1$··	$a_0a_1a_4a_4a_1$··
$a_0a_1a_0a_4a_2$··	$a_0a_1a_1a_4a_2$··	p: $a_0a_1a_3a_4a_2$	$a_0a_1a_4a_4a_2$(4)
$a_0a_1a_0a_4a_3$(4)	$a_0a_1a_1a_4a_3$(4)	$a_0a_1a_3a_4a_3$(4)	$a_0a_1a_4a_4a_3$(4)

For the function p and q $p \circ p = q$ and $q \circ q = p$ hold. So either none or both of them are in the clone Z. $3p + 3q$ takes 3 different values for a_2, a_3 and a_4 because $3a_3+3a_4$, $3a_4+3a_2$ and $3a_2+3a_3$ differ from each other. An easy computation shows that $3p+3q \neq x$. So $3p + 3q$ such a function at least with two fixed values which differs from p, q and from the functions within the frames.

Let $f(x) = a_0a_1a_ia_ja_k$ $(i,j,k \in \{0,1\})$ one of the two valued functions of the table. The values $2a_i+4a_2$ and $4a_i+2a_2$ differ from a_i, a_2 and from each other. So at least one of them is a_3 or a_4. Therefore at least one of the functions $2f(x) + 4x$ and $4f(x) + 2x$ is such a function with two fixed values which differ from every function with the frames.

Finally if the function 4 differs from $h \circ h$ at three values of x then the function $3h + 2(h \circ h) + x$ has two fixed values. Now we list the unary non linear functions of P_5 and reduce them to such functions which are in the lexicographical ordering more ahead.

x 0 1 2 3 4 $f(x)$	Reduction of f(x)
0 0 a b c	Every of these 124 functions has one of the previous properties.
0 1 a b c	Every of these 124 functions has at least two fixed values.
0 2 a b c	$2x+4(02abc) = 00def$
0 3 a b c	$4x+2(03abc) = 00def$
0 4 a b c	$3x+3(04abc) = 00def$
1 0 a b c	$f \circ f = 01a'b'c$
1 1 a b c	Every has one of the listed properties
1 2 0 a b	$f \circ f \circ f = 012a'b'$
1 2 1 a b	$f \circ f = 212cd$, $4(21\,2cd)+2(1\,21ab) = 030ef$
1 2 2 a b	$f \circ f = 222cd$, $4(222cd)+2(122ab) = 022ef$
1 2 3 a b	$f \circ f = 23acd$ $4(23acd)+2(123ab) = 01ghi$
1 2 4 a b	$f \circ f = 24bcd$ $4(24bcd)+2(124ab) = 00ghi$
1 3 a b c	$f \circ f = 3bdhi$ $2(3bdhi)+4(13abc) = 0jklm$
1 4 a b c	$f \circ f = 4cdhi$ $3(4cdhi)+3(14abc) = 0jklm$
2 a b c d	$3x+3(2abcd) = 1ghij$
3 a b c d	$4x+2(3abcd) = 1ghij$
4 a b c d	$2x+4(4abcd) = 1ghij$

ACKNOWLEDGEMENT

The author is indepted to *J. Demetrovics* for his remarks and advices during preparation of this paper.

REFERENCES

[1] Demetrovics, J., J. Bagyinszki; The lattice of linear classes in prime-valued logics; *Discrete Mathematics Banach Center Publications* 7(1982), 105-123.

[2] Marcenkov, S.S.; On clones in P_k containing homogeneous functions (Russian); preprint *Inst. Appl. Mathem. the USSR Academy of Sciences* 35(1984), 1-28.

[3] Baker, K.A., A.F., Pixley; Polynomial interpolation and the Chinese Remainder Theorem for algebraic systems; *Mathematische Zeitschrift*, 43(1975), 165-174.

Ö S S Z E F O G L A L Á S

A P_3 ÉS P_5-BEN LEVŐ VÉGESEN GENERÁLT KLÓNOK JELLEMZÉS LINEÁRIS FÜGGVÉNYEKKEL

Csikós M.

A cikkben a szerző a következő tételt bizonyitja be:

A P_3 és P_5-ben egy klón akkor és csakis akkor végesen generált, ha tartalmaz egy nemtriviális n-szeres lineáris függvényt ($n \geq 2$) és egy egyszeres nemlineáris függvényt. Ez Demetrovics és Bagyinszki ill. Marčenkov idevágó eredményeit egésziti ki.

КОНЕЧНО ПОРОЖДЕННЫЕ КЛОНЫ В P_3 И P_5 С ЛИНЕЙНЫМИ ФУНКЦИЯМИ

М. Чикош

В статье доказывается следующая теорема:

Клон содержащийся в P_3 или P_5 является конечно порожденным тогда и только тогда, если содержит n-арную линейную функцию ($n \geq 2$) и унарную нелинейную функцию. Этот результат добавляет новые информации к результатам Бадьински - Деметровича и Марченкова.

 Közlemények 31/1984. 23-31

STRUCTURAL CHARACTERISTICS OF ONE CLASS OF BOOLEAN FUNCTIONS

ČIMEV K.N. and ASLANSKI M.

Blagoevgrad Branch of Sofia University
Blagoevgrad, Bulgaria

The terminology used in this paper is from [1-15]. The paper treats the Boolean functions f, which essentially depend on at least six (seven) variables and for which there are variables x_i and x_j such that $\{x_i,x_j\} \in S_f$ and the subgraph of f with the elements of $R_f \setminus \{x_i,x_j\}$ as apexes has the shape of an elementary chain, opened or closed.

The set of all essential variables of f and all spearable sets of arguments of f is denoted by R_f and S_f, respectively.

$S_{f,2}$ stands for the set of all separable two-element sets of arguments of f.

The number of all separables pairs of the function $f(x_1,\ldots,x_n)$ wherein x_i takes part, will be referred to as the order of the variables x_i for the above mentioned function.

Theorem I. If the set $\{x_i,x_j\}$ is separable for a Boolean function $f(x_1,\ldots,x_n)$ which essentially depends on $(n \geq 6)$ variables and the subgraph of f with the elements of the set $\{x_1,\ldots,x_n\} \setminus \{x_i,x_j\}$ as apexes has the shape of an elementary open chain, then one of the variables x_i,x_j is of order $n-1$ for f, and the other variable is of order not smaller than $n-4$.

Proof. Under the conditions stated in the theorem for the function $f(x_1,\ldots,x_n)$, let us accept that $\{x_{n-1},x_n\} \in S_f$ and that the subgraph of f with apexes $x_1,x_2,\ldots,x_{n-2}$ has the

shape of an elementary open chain, where for every $i = 1,\ldots,n-3$

$$\{x_i, x_{i+1}\} \in S_f.$$

At least one of the variables x_{n-1}, x_n is of order $n-1$ for f. Without restricting the subject examination let us accept that x_{n-1} is of an order $n-1$ for f. We shall prove that x_n is of an order not smaller than $n-4$.

We shall do it by using the method of mathematical induction.

We shall prove the theorem when $n = 6$. We must prove that x_6 is of an order not smaller than 2. Let us assume the opposite. So x_6 must be of order 1 for f, moreover $\{x_5,x_6\} \in S_f$.

Let us denote by α the value of x_5 for which it is true that

$$x_6 \in R_{f(x_5=\alpha)}$$

Then it must be true that

$$R_{f(x_5=\bar{\alpha})} = \{x_1,x_2,x_3,x_4\},$$

and

$$S_{f(x_5=\bar{\alpha}),2} = \{\{x_1,x_2\},\{x_2,x_3\},\{x_3,x_4\}\},$$

which is impossible, according to theorem 21 from [14]. Let us prove the theorem for $n=7$. We must prove that x_7 is of order not smaller than the 3. Let us suppose it is not true, that is x_7 is of order 1 or 2 for $f(x_1,\ldots,x_7)$.

The variable x_7 cannot be of order 1 for $f(x_1,\ldots,x_7)$. Indeed, if $f(x_6 = \alpha)$ depends essentially on x_7, it means that

$$S_{f(x_6=\bar{\alpha}),2} = \{\{x_1,x_2\},\{x_2,x_3\},\{x_3,x_4\},\{x_4,x_5\}\},$$

which is not possible.

So x_7 must be of order 2 for $f(x_1,\ldots,x_7)$ where $\{x_6,x_7\} \in S_f$.

Thus x_7 forms exactly one separatable pair for f with a variable from the set $\{x_1,\ldots,x_5\}$. Let us accept that $\{x_7,x_5\} \in S_f$. Let α be a value for x_5 for which $f_1 = f(x_5 = \alpha)$ depends essentially on x_7. According to theorem 26 from [14],

$$R_{f_1} = R_f \setminus \{x_5\}.$$

From theorem 21 and 26 from [14] it follows that

$$\{x_6,x_7\} \in S_{f_1},$$

and the subgraph of f_1 with the apexes $x_1,\ldots,x_4$ has the shape of elementary open chain. Moreover x_7 must be of order 1, for f_1 with respect to the separable pairs, which contradicts to the already proved case of the theorem when $n = 6$.

By analogy we come to a contradiction if we accept that $\{x_1,x_7\} \in S_f$. Let us accept that

$$\{x_4,x_7\} \in S_f.$$

Let c_2 be a value of x_2 for which

$$x_1 \in R_f, \quad f_2 = f\ (x_2 = c_2).$$

Then

$$R_{f_2} = R_f \setminus \{x_2\},$$

and x_1 will be of order 1 for f_2 and

$$\{x_1,x_6\} \in S_{f_2}.$$

Let c_6 be a value for x_6 such that the function $f_2\ (x_6=c_6)$ depends essentially on x_1. Then

$$R_{f_3} = R_f \setminus \{x_1,x_2,x_{n-1}\},$$

where

$$f_3 = f_2\ (x_6 = \bar{c}_6).$$

The function f_3 must have three variables of order 1. But this is impossible (see [4] and [14]).

By analogy we come to a contradiction if we accept that

$$\{x_2,x_7\} \in S_f.$$

Let us accept that

$$\{x_3, x_7\} \in S_f.$$

Let c_3 be a value for x_3 such that the function $f_4 = f(x_3 = c_3)$ essentially depends on x_7. In this case x_6 must be of the order 5 for f_4 nd separable pairs for f must not be formed from the elements of the set $\{x_7\}, \{x_1, x_2\}$, $\{x_4, x_5\}$, which is impossible.

This proves the theorem when $n = 7$, too.
Let us accept that the theorem is true for some $n \geq 7$. We shall prove, that it is also true for the Boolean functions, which essentially depend on $n + 1$ variables and fulfil all the conditions of the theorem.

Let $f(x_1, \ldots, x_{n+1})$ be a Boolean function, which essentially depends on $n + 1$ $(n \geq 7)$ variables and $\{x_n, x_{n+1}\} \in S_f$. According to the conditions of the theorem the subgraph of f with the elements of the set $\{x_1, \ldots, x_{n-1}\}$ as apexes has the shape of an elementary open chain. For example let us accept that for every $i = 1, \ldots, n-2$,

$$\{x_i, x_{i+1}\} \in S_f.$$

At least one of the variables x_n, x_{n+1}, is of order n. For example, let x_n be of order n for the function $f(x_1, \ldots, x_{n+1})$, with respect tothe separable pairs. We shall prove that x_{n+1} is of order not smaller than $n - 3$ for $f(x_1, \ldots, x_{n+1})$.

We shall prove that under the conditions given in the theorem $\{x_1, x_{n+1}\} \in S_f$ or $\{x_{n-1}, x_{n+1}\} \in S_f$.

Let us assume this is not true, i.e.

$$\{x_1, x_{n+1}\} \notin S_f \quad \text{and} \quad \{x_{n-1}, x_{n+1}\} \notin S_f.$$

Let c_2 be a value for x_2 such that

$$x_1 \in R_{f_5}, \; f_5 = f(x_2 = c_2).$$

Then

$$R_{f_5} \supset \{x_3, \ldots, x_{n-1}\}.$$

The variable x_1 must be of order 1 for f_5. Thus $x_n \in R_{f_5}$. We shall prove that $x_{n+1} \in R_{f_5}$. If we assume the opposite and choose α in such a way that

$$x_1 \in R_{f_5}(x_n = \alpha),$$

then the function

$$f_6 = f_5\,(x_n = \bar{\alpha}),$$

will depend essentially on $n-3$ variables and its graph will have the shape of an elementary open chain; and this is impossible. Therefore $x_{n+1} \in R_{f_5}$.

Since we have assumed that $\{x_{n-1}, x_{n+1}\} \notin S_f$, so $\{x_{n-1}, x_{n+1}\} \notin S_{f_5}$. Let α be a value for x_n such that

$$x_1 \in R_{f_5}(x_n = \alpha).$$

Then the function

$$f_6 = f_5(x_n = \bar{\alpha})$$

will essentially depend on $n-2$ variables. So x_{n+1} will be of order 1 for f_6 and $\{x_{n-1}, x_{n-2}\} \in S_{f_6}$. Therefore it must be true that $\{x_3, x_{n-2}\} \in S_{f_6}$, and $\{x_3, x_{n-2}\} \in S_f$ where $n \geq 7$. The last statement contradicts to the initially given conditions.

We proved that

$$\{x_1, x_{n+1}\} \in S_f \quad \text{or} \quad \{x_{n-1}, x_{n+1}\} \in S_f.$$

Let us discuss the case when $\{x_1, x_{n+1}\} \in S_f$. Let c_1 be a value for x_1 such that

$$\{x_n, x_{n+1}\} \in S_{f_7}, \quad f_7 = f(x_1 = c_1).$$

But

$$R_{f_7} = \{x_2, \ldots, x_{n+1}\}$$

and the subgraph of f_7 with the elements of the set $\{x_2, \ldots, x_{n-1}\}$ as apexes has the shape of elementary open chain.

From the inductive assumption it follows that one of the variables x_n, x_{n+1} is of order $(n-1)$ for f_7, and the other one is of an order not smaller than $n-4$.

Therefore the variable x_{n+1} will be of order not smaller than $x-4$ for f_7. Thus x_{n+1} will be of order not smaller than $n-3$ for f.

By analogy we can prove the theorem if $\{x_{n-1}, x_{n+1}\} \in S_f$. The theorem is proved. □

Is it possible to strenghten theorem 1 in the sense that, under the conditions of the theorem, one of the variables x_i, x_j is of order $n-1$ $(n \geq 6)$ and the other one is at least of order $n-3$ for f, in regard to the separable pairs? The answer is NO.

For example, let us take the Boolean function

$$f = x_1(x_2x_3 + \bar{x}_3x_4) + \bar{x}_1(x_4x_5 + x_5\bar{x}_6) \quad (\mathrm{mod}\ 2).$$

The subgraph of f with apexes x_2, x_3, x_4, x_5 has the shape of an elementary open chain. The pair $\{x_1, x_6\}$ is separable for f, x_1 is of order 5 and x_6 is of order 2 for f in regard to the separable pairs.

Theorem II. If the set $\{x_i, x_j\}$ is separable for the Boolean function $f(x_1, \ldots, x_n)$, which essentially depends on n $(n \geq 7)$ variables and the subgraph of f with the elements of the set $\{x_1, \ldots, x_n\} \setminus \{x_i, x_j\}$ as apexes has the shape of an elementary closed chain, then one of the variables x_i, x_j is of order $n-1$ for f, and the other one is or order not smaller than $n-4$.

Proof. Let us assume that the function $f(x_1,\ldots,x_n)$, $(n \geq 7)$, satisfies the conditions of the theorem. We may assume that

$$\{x_{n-1},\ x_n\} \in S_f,$$

and that the subgraph of f with the elements of the set $\{x_1,\ldots,x_{n-2}\}$ as apexes has the shape of a closed elementary chain, and for every $i = 1,\ldots,n-3$

$$\{x_i,\ x_{i+1}\} \in S_f \quad \text{and} \quad \{x_{n-2},x_1\} \in S_f.$$

At least one of the variables x_{n-1},x_n is of order $n-1$ for f. Let us accept that x_{n-1} is of order $n-1$ for f. We shall prove that x_n is of order not smaller than $n-4$ for f.

It is impossible for x_n to be of order 1 for f. Let us assume the opposite. If α is a value of x_{n-1} such that

$$x_n \in R_{f(x_{n-1}=\alpha)},$$

then the graph of $f(x_{n-1} = \bar{\alpha})$ must have the shape of a closed elementary chain, which is impossible.

From what we have assumed it follows that there is a variable x_i, $i \in \{1,\ldots,n-2\}$ such that $\{x_i,\ x_n\} \in S_f$. Let x_i be such a variable and let c_i be a value such that

$$\{x_{n-1},\ x_n\} \in S_{f(x_i=c_i)}.$$

But in this case

$$R_{f(x_i=c_i)} = R_f \setminus \{x_i\},$$

and the subgraph of $f(x_i = c_i)$ with the elements of the set $\{x_1,\ldots,x_{n-2}\}\setminus\{x_i\}$ as apexes has the shape of an elementary open chain. According to theorem 1 the variable x_n must be of order not smaller than $n-5$ for $f(x_i=c_i)$. Then x_n will be of order not smaller than $n-4$ for f.

Thus the theorem is proved.

REFERENCES

1. S.V. JABLONSKI, -Functional constructions in the k-valued logic (in Russian), Trudy Mat. Inst. Steklov, 51 (1958), 5-142.

2. O.B. LUPANOV, On a class of schemes consisting of functional elements (in Russian), Problemy Kibernet., 7(1962), 61-114.

3. N.A. SOLOV'EV, On the question of essential dependence of Boolean functions (in Russian), Problemy Kibernet., 9 (1963), 333-335.

4. JU. JA. BREITBART, Essential variables of Boolean functions (in Russian), Dokl. Akad. Nauk SSSR, 172 (1967), 9-10.

5. A. SALOMAA, On essential variables of functions, especially in the algebra of logic, Ann. Acad. Sci. Fenn. Ser. A. I., 339 (1963), 3-11.

6. R.E. SCHWARTZ, Existence and uniqueness properties of subfunctions of Boolean functions, SIAM J. Appl. Math., 18(1970), 454-461.

7. J. DEMETROVICS J., L. HANNAK, S. MARCHENKOV. Some remarks on the structure of P_3. C.R. Math. Rep. Acad. Sci. Canada, vol. II (1980), 4, 215-219.

8. CIMEV, K.N. On some properties of functions. Colloquia Math. Soc. Janos Bolyai. Finite algebra and multiple--valued logic, Szeged (Hungary), 1979, 97-110.

9. K.N. ČIMEV, Dependence of the functions of P_k on their arguments (in Bulgarian), Godisnik Viss. Tehn. Ucebn. Zaved. Mat., 4:3 (1967), 6-13.

10. K.N. ČIMEV, Dependence of the functions of k-valued logic on their arguments (in Bulgarian), Godisnik Viss. Tehn. Ucebn. Zaved. Math., 6:2 (1970), 58-62.

11. K.N. ČIMEV, Separable pairs of function (in Bulgarian), Godisnik Viss. Tech. Ucebn. Zaved. Math., 7:3 (1971), 7-12.

12. K.N. ČIMEV, Separable subsets and strongly essential variables of functions (in Bulgarian), Godisnik Viss. Tehn. Ucebn. Zaved. Mat., 10:4 (1974), 7-13.

13. K.N. ČIMEV, Some properties of functions (in Bulgarian), Godisnik Viss. Tehn. Ucebn. Zaved. Mat., 7:1 (1971), 23-32.

14. K.N. ČIMEV, Subfunctions and separable sets of arguments of functions (in Bulgarian), Mathematics and education in mathematics, Sunny Beach, April 1982, 108-122.

15. K.N. ČIMEV, Separable sets of arguments of functions (in Bulgarian). Blagoevgrad, 1982, 207.

Ö S S Z E F O G L A L Á S

A BOOLE FÜGGVÉNYEK EGY OSZTÁLYÁNAK STRUKTURÁLIS TULAJDONSÁGAIRÓL

K.N. Čimev, M. Aslanski

A szerzők a cikkben $n(\geq 6)$ változós Boole függvényeknek egy osztályára adnak jellemzést. Bebizonyitják, hogy a függvény változói közül egynek a rendje $n-1$ és egy másiknak a rendje legalább $n-4$.

О СТРУКТУРАЛЬНЫХ СВОЙСТВАХ ОДНОГО КЛАССА БУЛЕВЫХ ФУНКЦИЙ

К.Н. Чимев, М. Аслански

В статье дается характеризация одного класса Булевых функций $n(\geq 6)$ переменных. Доказывается, что для функций из одного класса одна из двух переменных имеет порядок $n-1$ и порядок другой не меньше $n-4$.

LANGUAGES OF SYNTHESIZED CONCURRENT SYSTEMS

DANG VAN HUNG
Institute of Computer Science and Cybernetics
Hanoi, Vietnam

1. Introduction

This paper deals with the analysis of behaviours of concurrent systems synthesized from particular parts. Such systems have attracted a great deal of attention. Important contributions are [1,3,4,5].

We propose in this paper to describe a set of computation sequences of a concurrent system by mean of the formal languages. This, to our mind, has a little sense in understanding concurrent systems more essentially.

In the second section, we shall defined synthesized languages and in the third one, its application will be discussed.

2. Synthesized language

The concept defined in this section closely concernes the concept of the synchronization behaviours introduced by Nivat [5].

Let L_1, L_2 be languages on alphabets Σ_1, Σ_2, respectively, $\Sigma=\Sigma_1\cap\Sigma_2$. Homomorphisms h_1 and h_2 are defined on Σ_1 and Σ_2, take values in Σ.

$$h_1(a) = \textit{if } a\in\Sigma \quad \textit{then } a \textit{ else } e,$$
$$h_2(a) = \textit{if } a\in\Sigma \quad \textit{then } a \textit{ else } e,$$

where e is the emty word.

Suppose $L=h_1(L_1)\cap h_2(L_2)$. Each element of L is considered as a sequence of synchronization elements of L_1 and L_2.

(The reason will be seen later.)

Definition: The language

$$\bigcup_{w \in L} h_1^{-1}(w) \circledast h_2^{-1}(w)$$

is called a synthesized language (SL in short form) of L_1 and L_2.

In our definition, $\circledast$ is the symbol of the projetive product defined by E.Knuth, Gy.György and L.Ronyai [1].

To make the definition more clear, we should note that:

$$u \in h_1^{-1}(w) \circledast h_2^{-1}(w) \quad iff$$

$$u = a_1 a_2, \ldots, a_n,$$

$$w_1 = a_{i_1} a_{i_2}, \ldots, a_{i_l} \in h_1^{-1}(w), \quad i_1 < i_2 <, \ldots, < i_n,$$

$$w_2 = a_{j_1} a_{j_2}, \ldots, a_{j_s} \in h_2^{-1}(w), \quad j_1 < j_2 <, \ldots, < j_s,$$

$$\{i_1, i_2, \ldots, i_l\} \cup \{j_1, j_2, \ldots, j_s\} = \{1, 2, \ldots, n\}.$$

and w is the longest common word of w_1 and w_2.
Considering L_i as sets of behaviours of processes $Q_i, i=1,2$, we can identify L with the sets of sequences of synchronization promitives of these processes.

A synthesized language of two regular languages is regular. This fact can be proped by constructing an automaton from ones recognizing the component languages.

Assume that M_1, M_2 are automatons recognizing L_1 and L_2:

$$M_1 = (Q_1, \Sigma_1, \delta_1, q_1^o, F_1)$$

$$M_2 = (Q_2, \Sigma_2, \delta_2, q_2^o, F_2) .$$

We construct automaton M as follows:

$$M = (Q, \Sigma_1 \cup \Sigma_2, \delta, q^o, F)$$

$$Q = Q_1 x Q_2, q^o = (q_1^o, q_2^o), \; F = F_1 x F_2 ,$$

$$\delta((q_1,q_2),a) = \begin{cases} (\delta_1(q_1,a), q_2) & \text{if a } \in \Sigma_1 \setminus \Sigma, \\ (q_1, \delta_2(q_2,a)) & \text{if a } \in \Sigma_2 \setminus \Sigma, \\ (\delta_1(q_1,a), \delta_2(q_2,a)) & \text{if a } \in \Sigma. \end{cases}$$

It is easy to prove that the language recognized by M is SL of L_1 and L_2 . The proof is omitted.

By induction we also defined the SL of the n given languages $L_1, L_2, \ldots, L_n$. That is, the SL of $L_1, L_2, \ldots, L_n$ is the SL of $L_1, L_2, \ldots, L_{n-1}$ and L_n. It can be seen that this definition does not depend on the order of $L_1, L_2, \ldots, L_n$.

The concept introduced above can be used to study state-machine decomposable systems, such as a sets of behaviours of vector of processes introduced by Nivat [5]. In this paper we shall apply this concept to present the set of computation sequences of the computation system introduced by Janicki [2,3].

3. Languages generated by S-nets

S-nets (simple nets) introduced by Janicki [2] are a particular case of Petri nets.

Definition (2)

A simple net (S-net) $N=(T,P)$ consist of:

i) a set T of transitions,

ii) $P \subseteq 2^T x 2^T$ - a relation over 2^T, a set of places,

where T and P satisfy the following conditions:

$(\forall a \in T)(\exists p,q \in P): (a \in left(p) \cap right(q))$; $P=\emptyset$ *iff* $T=\emptyset$.

$((x,y) \in X x X, \; right(x,y)=y, \; left(x,y)=x)$.

Denote by SNETS the set of S-nets. For $N_1, N_2 \in SNETS, N_1 \subseteq N_2$ iff $P_1 \subseteq P_2$. Relations is a partial order over SNETS.

We shall adopt the following notations:

$$N_1 \cup N_2 = sup\{N_1, N_2\}, \quad N_1 \cap N_2 = inf\{N_1, N_2\},$$

$$\bigcup_{N \in S} N = sup\{N \mid N \in S\}, \qquad \bigcap_{N \in S} N = inf\{N \mid N \in S\}.$$

Theorem 1 [2]: Algebra (SNETS, $\cup$, $\cap$) is a complete lattice with the smallest element $(\emptyset, \emptyset)$

A marked S-net $M=(N,B,E)$ consists of:

a) S-net N,
b) $B \in 2^P$ - initial marking, and
c) $E \in 2^P$ - final marking.

1-step reaching relation $R_1^N : T \to 2^P x 2^P$ is defined by:

$$(\forall M_1, M_2 \in 2^P), (M_1, M_2) \in R_1^N(a) \quad iff$$

$$M_1 - {}^\bullet a = M_2 - a^\bullet \; and \; {}^\bullet a \subseteq M_1, \; a^\bullet \subseteq M_2.$$

(where $a^\bullet = \{p \in P \mid a \in right(p)\}$, ${}^\bullet a = \{p \in P \mid a \in left(p)\}$.
A word $w = a_1 a_2, \ldots, a_n$ is called a firing sequence from marking M' to marking M'' if there exists a sequence of markings $M' = M_0, M_1, \ldots, M_n = M''$ such that:

$$(M_{i-1}, M_i) \in R_1^N(a_i), \quad i = 1, 2, \ldots, n.$$

The language $L(M)$ of M is the set of all firing sequences from the initial marking to the final marking.

The main result can be stated as follows:

Theorem 2: Let $M^1 = (N_1, B_1, E_1)$ and $M^2 = (N_2, B_2, E_2)$ be marked S-nets. Suppose $M^0 = (N_1 \cup N_2, B_2 \cup B_2, E_1 \cup E_2)$.

Then the language generated by M is the SL of $L(M^1)$ and $L(M^2)$.

Proof: Denote

$$L_1=L(M_1^1),\ L_2=L(M_2^2),\ L=L(M^o),$$

$$\bar{T}=T_1\cap T_2,\ T=T_1\cup T_2,\ P=P_1\cup P_2,\ B=B_1\cup B_2,\ E=E_1\cup E_2.$$

Our task is to prove that:

$$L=SL \quad of \quad L_1 \quad and \quad L_2.$$

Taking $w=a_1a_2,\ldots,a_m\in L$, we show that w belongs to SL of L_1 and L_2. By definition of $L(M)$, there exist $M_1,M_2,\ldots,M_m\in 2^P$ such that:

$$(M_{i-1},M_i)\in R_1^N(a_i),\ i=1,2,\ldots,m,\ M_o=E,M_m=E. \qquad (2)$$

Assume that $a_{i_1},a_{i_2},\ldots,a_{i_l}$ is the subsequence of w containing occurences of transitions in T, $a_{j_1},a_{j_2},\ldots,a_{j_s}$ is the subsequence of w containing occurences of transitions in T_2, and $c_1c_2,\ldots,c_q$ is the longest common subsequence of $a_{i_1}a_{i_2}\ldots a_{i_l}$ and $a_{j_1}a_{j_2}\ldots a_{j_s}$ containing occurences of transitions in $\bar{T}$. That is,

$$\{j_1,j_2,\ldots,j_s\}\cup\{i_1,i_2,\ldots,i_1\}=\{1,2,\ldots,m\},$$

$$j_1<j_2<\ldots<j_s,\quad i_1<i_2<\ldots<i_l\ .$$

We shall show that:

$$(M_{i_{h-1}}\cap P_1,\ M_{i_h}\cap P_1)\in R_1^{N_1}(a_{i_h}),\quad h=1,2,\ldots,l,$$

and

$$(M_{j_{k-1}}\cap P_2,\ M_{j_k}\cap P_2)\in R_1^{N_2}(a_{i_k}),\quad k=1,2,\ldots,s.$$

This follows from the fact that N_1 and N_2 are S-nets.

Since N_1 and N_2 are S-nets,

$$\forall t \notin T_1 \Rightarrow \dot{t} \cup \dot{\dot{t}} \subseteq P_2 \setminus P_1,$$
$$\forall t \notin T_2 \Rightarrow \dot{t} \cup \dot{\dot{t}} \subseteq P_1 \setminus P_2.$$

Consequently,

$$\forall h=1,2,\ldots,l, \quad if \quad i_{h-1}<i_h-1 \quad then$$
$$M_{i_{n-1}} \cap P_1 = M_{i_{n-1}+1} \cap P_1 = \ldots = M_{i_n-1} \cap P_1,$$
$$\forall k=1,2,\ldots,s, \quad if \quad j_{k-1}<j_k-1 \quad then$$
$$M_{j_{k-1}} \cap P_2 = M_{j_{k-1}+1} \cap P_2 = \ldots = M_{j_k-1} \cap P_2,$$
$$if \quad j_s<m \quad then \quad M_{j_s+1} \cap P_2 = \ldots = M_m \cap P_2,$$
$$if \quad i_l<m \quad then \quad M_{i_l+1} \cap P_1 = \ldots = M_m \cap P_1.$$

Since $M_o \cap P_1 = B_1$, $M_o \cap P_2 = B_2$, $M_m \cap P_1 = E_1$, $M_m \cap P_2 = E_2$ and (2) $a_{i_1} a_{i_2} \ldots a_{i_l} \in L_1$, $a_{j_1} a_{j_2} \ldots a_{j_s} \in L_2$. By definition of SL, w in SL of L_1, L_2.

Reversely, take $w=a_1 a_2 \ldots a_m$ in SL of L_1 and L_2. By definition of SL, we have:

$$w_1 = a_{i_1} a_{i_2} \ldots a_{i_l} \in L_1 \quad (if \quad w=e, \ i_l=0),$$
$$w_2 = a_{j_1} a_{j_2} \ldots a_{j_s} \in L \quad (if \quad w=e, \ j_s=0),$$

$w=a_{r_1} a_{r_2} \ldots a_{r_q}$ is the longest common subsequence of w_1 and w_2 (if $w=e$, $r=o$),

$$\{i_1, i_2, \ldots, i_l\} \cup \{j_1, j_2, \ldots, j_s\} = \{1, 2, \ldots, m\},$$
$$\{i_1, i_2, \ldots, i_l\} \cap \{j_1, j_2, \ldots, j_s\} = \{r_1, r_2, \ldots, r_{r_q}\},$$
$$i_1<i_2<\ldots<i_{i_l}, \ j_1<j_2<\ldots<j_s, \ r_1<r_2<\ldots<r_{r_q},$$

($i_o=j_o=r_o=0$ by convention).

Therefore, there exist $M^1_{i_o}, M^1_{i_1}, \ldots, M^1_{i_l}$ and $M^2_{j_o}, M^2_{j_1}, \ldots, M^2_{j_s}$ such that:

$$\forall h=1,2,\ldots,l,\quad M^1_{i_h} \in 2^{P_1}, (M^1_{i_{h-1}}, M^1_{i_h}) \in R_1^{N_1}(a_{i_h}),$$

$$\forall k=1,2,\ldots,s,\quad M^2_{j_k} \in 2^{P_2}, (M^2_{j_{k-1}}, M^2_{j_k}) \in R_1^{N_2}(a_{j_k}),$$

$$M^1_{i_o}=B_1,\ M^1_{i_l}=E_1,\ M^2_{j_o}=E_2,\ M^2_{j_s}=E_2.$$

We construct a sequence $M_o, M_1, \ldots, M_m$ of markings of N by:

$$M_o = B,$$

$$M_i = M^1_i \cup M^2_i \quad \text{if} \quad i \in \{r_1, r_2, \ldots, r_q\}.$$

In remain cases, if $i=i_h$ with $h \in \{1,2,\ldots,l\}$ then $M_i=M^1_{i_h} \cup M^2_{j_k}$ with k in $\{1,2,\ldots,s\}$ so that j_k is the bigest number, which is less than i, if $i=j_k$ with k in $\{1,2,\ldots,s\}$ then $M_i=M^1_{i_h} \cup M^2_{j_k}$ with h in $\{1,2,\ldots,l\}$ so that i_h is the bigest number which is less than j_k. (3)

We claim that

$$\forall i=1,2,\ldots,m,\ (M_{i-1}, M_i) \in R_1^N(a_i).$$

There are the following cases:

a) $i=i_h$, $h \in \{1,2,\ldots,l\}$, $i_h \notin \{r_1, r_2, \ldots, r_q\}$. In this case, $M_i=M^1_{i_h} \cup M^2_{j_k}$ with k defined in the definition of M_i and $i-1=i_{h-1}$ (i_{h-1} can be in $\{r_1, r_2, \ldots, r_q\}$ or $i-1=j_k$ (j_k does not belong to $\{r_1, r_2, \ldots, r_q\}$). By definition, $M_{i-1} = M^1_{i_{k-1}} \cup M^2_{j_k}$. Noting in this case $a_{i_h} \cup^{\cdot} a_{i_h} \in P_1$ we have

$(M_{i-1},M_i)\in R_1^N(a_i)$.

b) $i=j_k$, $k\in\{1,2,\ldots,l\}$ and $j_k\in\{r_1,r_2,\ldots,r_q\}$. In exactly the same way with interchanging the role of N_1 and N_2, we have $(M_{i-1},M_i)\in R_1^N(a_i)$.

c) $i\in\{r_1,r_2,\ldots,r_q\}$. In this case, there exist h and k such that $j_k=i_h=i$. So, $M_i=M^1_{i_h}=M^l_{j_k}$.

If $i-1=i_{h-1}=j_{k-1}$, then $M_{i-1}=M^1_{i_{h-1}}\cup M^2_{j_{k-1}}$.

If $i-1=i_{h-1}\neq j_{k-1}$, then j_{k-1} is the begest number in $\{j_1,\ldots,j_s\}$ which is less then i_{h-1}; if $i-1=j_{k-1}\neq i_{h-1}$ then i_{h-1} is the begest number in $\{i_1,\ldots,i_l\}$ which is less then j_{k-1}. So, in any case of $i-1$, $M_{i-1}=M^1_{i_{h-1}}\cup M^2_{j_{k-1}}$, and since that:

$$(M_{i-1},M_i)\in R_1^N(a_i).$$

The proved property of sequence $M_o,M_1,\ldots,M_m$ shows that $w\in L(M^o)$.

This completes the proof of our theorem.

Corollary: The language generated by proper net $M=(N,B,E)$ is the sythesized language of languages generated by sequential components creating it.

Proof: Since M is proper net, there are elementary nets $N_1,N_2,\ldots,N_k$ such that:

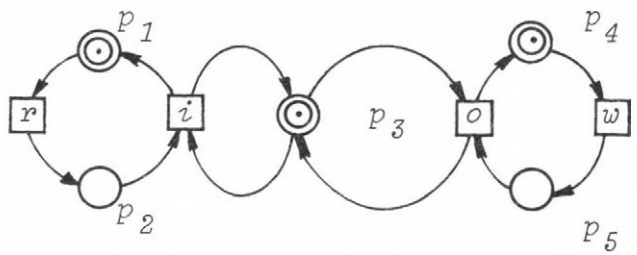
$$N = N_1\cup N_2\cup\ldots\cup N_k.$$

For every i, $M^i=(N_i,B\cap P_i,E\cap E_i)$ is a marked S-net $(i=1,\ldots,k)$. The corollary is an immediate consequence of theorem 2.

Janicki has used proper nets to model parallel computation

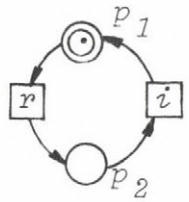
systems. The corollary given above can be used in proving the correctness of synthesizing concurrent systems from sequential parts.

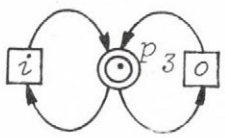
Example: Consider the following proper net:

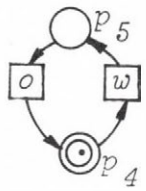
$N =$ 

where: r : read,
i : input,
o : output,
w : write, ⊙ : initial place, ◎ final place.

This net is the union of three sequential components:

$N_1 =$ 

, $N_2 =$ 

, $N_3 =$ 

$L_1=(ri)^*$, $L_2=(i^*o^*)^*$, $L_3=(wo)^*$

By the corollary, we have:

$$L_N = \{\lceil\{r^n w^m o^m i^n \mid n\geq 0, m\geq 0\}\rceil\} ,$$

here $\lceil\{r^n w^m o^m i^n \mid n\geq 0, m\geq 0\}\rceil$ is the trace language on $\{r,w,i,o\}$ under the relation $I=\{(r,w),(r,o),(o,i),(i,w)\}$.

L_N is the behaviours of read-write systems.

Conclusion:

In this paper we have only considered representation of behaviours of systems synthesized from their components. The combination of concept given above and trace languages used in

searching parallel system will be considered in the future.

The author would like ot thank prof.E.Knuth for helpful advices.

REFERENCE

[1] E.Knuth,Gy.György, L.Ronyai: A study of the projection operation. Apllication and theory of Petri nets, 52, Springer-Verlag, Berlin Heidelberg-New York 1982.

[2] Ryszard Janicki: An algebraic structure of Petri nets. Lecture Notes in computer science, Vol.83,1980.

[3] Ryszard Janicki: A construction of concurrency relation. Lecture Notes in computer science, IV/1981.

[4] Peter Hendersen, Yechezkel Zalestein: Sychronization problem solvable by generalized PV-systems. JACM, Janury, 1980, Vol.27, n-1

[5] Nivat,M.: Synchronization of concurrent processes. In pre. N-80-3, Janvier 1980. Paris.

[6] E.Knuth: Petri nets and regular trace languages. 25-th. April, 1978, the University of Newcastle Upon Time, computing labor

Ö S S Z E F O G L A L Á S

SZINTETIZÁLT KONKURENS RENDSZEREK NYELVEI

Dang Vah Hung

A dolgozatban a szerző bevezeti a "szinkron-nyelv" fogalmát, amely Knuth Előd [6] által már korábban definiált fogalomnak továbbfejlesztése. A szerző a fogalom segitségével tanulmányozza a Janicki [3] által bevezetett rendszereket. A dolgozat fő célja jobb betekintést nyerni a parallel rendszerek strukturájába.

ЯЗЫКИ СИНТЕЗИРОВАННЫХ КОНКРУРЕНТНЫХ СИСТЕМ

Данг Ван Хунг

В работе вводится понятие синхронного языка, которое является развитием базисного понятия введенного Кнутхом для изучения систем введенных Яницким. Цель работы состоит в том, чтобы лучше понять поведение параллельных систем.

 Közlemények 31/1984. 45-50

SOFTWARE DEVELOPED IN CUBA FOR DATA PROCESSING ON MINICOMPUTERS

E. MUNIZ - M. FONFRIA - M. BRAGADO

ICID, Habana, Cuba

The introduction and evolution of the electronic computer into the Cuban economy dates from the end of the 1960's. During this period the first models of computer machines were imported. They came from England, France, and the Soviet Union. At the same time began the researches to obtain a minicomputer to be designed and constructed in Cuba.

At the beginning, the minicomputer was to be used in the railroad service of the sugar cane industry. Our first minicomputer was constructed in 1970. After this year we began to use our own computer techniques.

Our Cuban computer systems are used in process control, business application, scientific calculation; mainly in the different levels of our administrative structure. The more generalized application implemented on our computer systems is the commercial type.

There are many different data processing systems developed fundamentally over our CID 201 B (compatible PDP-8) and CID 300 (compatible PDP-11/20) systems.

- The field of Electronic Data Processing has become a highly specialized technical area. Data are basically defined as information. Any information, whether it relates to a person, a business operation, or something else, may be considered data; the job of processing information is data processing.

A data processing system is an integrated group of programs which are used to create and to manipulate one or more files in

order to achieve some specific data processing objective. Inventory and payroll systems are, among others, the more common data processing systems.

To facilitate the implementation of commercial data processing systems, using CID 201 B, a COBOL language processor and several utility programs were developed.

The COBOL language was chosen because it is the most widely used language. We have had experience in COBOL language programming, using the French computer IRIS 50.

The utility programs allow basic functions such as listing of COBOL data files, file copy and file sorting.

To provide CID 300 with facilities for data processing, we considered our experiences on the CID 201 and the requirements of this kind of application.

Then, we decided to implement an operating system which should automatize to a high degree the development of commercial applications. This operating system must run over FOBOS operating system (compatible RT-11 DEC operating system). Then, some business oriented systems, developed for minicomputers similar to our machine, were analyzed. To continue our work tradition, the application programs must be written in COBOL language. For this reason, the operating system includes a processor for COBOL ANS 74 specification X.3.23.1974. This processor includes, among other features, the indexed organization for COBOL files and the possibility to perform input/output operations with additional terminals using ACCEPT and DISPLAY statements.

Our operating system was called GES 300. GES 300 is composed by three monitors and several utility programs. The monitors provide three different environments of task execution. They are a singly job monitor (SJ), a foreground-back-groung monitor (FB) and a multitask monitor (MTJ). These monitors do not need to run more than 56 k bytes of memory.

The SJ monitor is for a single program execution. The FB monitor allows the execution of two tasks, one of them with

higher priority than the other. The MTJ monitor permits previously compiled COBOL programs to be independently loaded and they are executed in a multiprogramming environment. The programs can be loaded either from the same terminal or from different terminals. During the executions, each program seems to have at its disposal the full resources of the GES 300 system. Besides, under MTJ the COBOL programs can share data files.

The MTJ monitor dynamically manages the allocation of the available memory, used by COBOL programs. The number of COBOL tasks that may be executed simultaneously depends on the program's length and the number and kind of data files used by them. The maximum number of tasks that may run simultaneously can be defined at GES 300 generation time.

The utility programs have been conceived as a comfortable environment to program classic processes in the commercial data treatment.

We have used the "program generator technique" in the conception of some of the utility programs.

A program generator can be informally defined as follows:

> A program generator is a type of translator which, using an easy and comfortable source language (Le), performs a transformation function (Ft) of this language and creates a program (Pr) which is written in a new language (Ls) which is capable of performing an execution function (Fe). That is:

$$Ft\ (Le) = Pr; \qquad LPr = Ls$$

$$Fe\ (Ls) = result$$

In the case of GES 300, Ls is the COBOL language.

All program generators use the same general algorithm but changes in the input-output information produce treatment particularities.

The program generator technique is used in Management Information systems.

The GES 300 utility programs developed using that technique are:

- update program generator (GEPACT)
- report writer generator (REPGEN)
- program generator to create relative files using
- randomizing methods (GENALE)
- consolidation program generator (GPCON)
- program generator to obtain sample COBOL files (GEFENS)

Other utility programs are:

- interactive edit program for COBOL files (EDICOB)
- program to collect and validate data (PREDAM)
- programs to sort COBOL files (COSORD, COSORT)
- programs to merge COBOL files (MERGED, MERGET)

We have developed three utility programs, one for each monitor, which provide line printer spooling.

We provide, to develop application programs, the FOBOS components PIP, EDIT, SRCCOM and BATCH and the COBOL processor.

The utility program GENGES tailors the system environment to the user's needs. GES 300 can be generated according to the specific hardware and software of the installation.

STATUS is another utility program which only runs over MTJ monitor. Using STATUS, the user can know how tasks are running on the system and which are the system generation parameters.

The operating system GES 300 was internationally tested in 1982 in Sofia, Bulgaria, with successful results.

CONCLUSION

At present, GES 300 is being used in many Cuban companies and institutes. There, the programming and debugging time to develop application systems has been considerable decreased. Besides, the introduction of new tools (the multiprogramming, the multiterminal access and the possibility to share COBOL files) has been allowed to develop more complete and integral application systems.

This new conception has been contributed to increase the experience of our application programmers.

REFERENCES

[1] Wooley, G. Contemporary COBOL. Rinehart Press, San Francisco, 1971, 1-2, 47-56.

[2] Singletary, W.E., ANS COBOL, a pragmatic approach. McGraw-Hill Book Company, New York, 1975. 81-94.

[3] Hernández, R., Los generadores de programas en la automatización de la programación. Revista CID, Enero-marzo de 1983, 43-44.

[4] Pedroso, O., Introducción y desarrollo de las técnicas de computación en Cuba. Situación actual y perspectivas. Revista CID, Ano 1, numero 4, 82 2-8.

[5] Hansen, P., Operating System Principles. Prentice-Hall, Inc., 1973.

[6] Shaw Alen, C., The logical design of operating systems. Prentice-Hall, Inc., New Jersey, 1974. 88-109.

[7] Cárdenas, A., Computer Science. Wiley Interscience, New York, 1972. 183-222, 461-486.

[8] Price, W., Introduction to Data Processing. Rinehart Press, San Francisco, 1972. 235-262.

[9] Roger, W., Considerations for computer implementation. Interface Age 3(8), Aug. 1978, 83-94.

[10] Fonfría, Miguel. COBOL 74 for Cuban computer, 1984.

[11] CTS 300 System. User's guide.

[12] Philippakis, Andreas S., and Leonard J. Kazmier, Information system through COBOL. McGraw-Hill, Inc. 1974.

Ö S S Z E F O G L A L A L Á S

A MINI-SZÁMITOGÉPEK SZÁMÁRA KUBÁBAN KIFEJLESZTETT ADAT-FELDOLGOZÁSI SZOFTVER-ESZKÖZÖK

E. Muniz, M. Fonfria, M. Bragado

A cikkben a szerzők áttekintik a CID 300 kubai mini-számitógépre tervezett szoftver-eszközöket. Részletesebben ismertetik a CID 300 számára kifejlesztett GES 300 operációs-rendszert.

РАЗВИТИЕ МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ РАЗРАБОТКИ ДАННЫХ НА МИНИ-КОМПЬЮТЕРАХ НА КУБЕ

Е. Муниз, М. Фонфриа, М. Брагадо

В статье описаны проблемы разработки данных на кубинском мини-компьютере CID 300. Авторы дают отчет о новой операционной системе GES 300 конструированной для CID 300.

Közlemények 31/1984. 51-61

ON THE SEMANTICS OF DESCRIPTION LANGUAGES

P. RADÓ

Computer and Automation Institute,
Hungarian Academy of Sciences

The dynamic growth (and rate of growt) of the amount and global cost of software products has been a tendency for the past 20 years all over the world. This process has brought about not only quantitative changes. The problems to be solved have been becoming more and more complex, the solutions - software systems - larger and larger, their construction more and more complicated. The efforts to solve the arising new problems have resulted in the birth of a new branch of the computer sciences: software engineering.

As experience has shown, the problems capable of making a software project unsuccessful, arise mainly at the early phases of the project [1]. After many years of research and numerous failed software projects the importance of requirement specification and design is generally recognized, although the available tools and methods provide no unique and universal solution for the practicioner.

One of the most promising approaches to the problem of requirements specification and design description is that of computer aided systems. It is quite natural, that the computer can help, storing the text, producing listings of different formats, answering interactive queries e.t.c. Of course this much can be achieved using any text editor. The text editors and the computer aided specification systems differ in the latter's ability to understand the meaning (strictly defined syntax and semantics) of the stored data. This capability adds the valuable facility of automatic consistency checking to the features of the system.

In this paper we propose a general computer aided specification system model. A semantics definition valid for a wide class of specification languages is given based on this model.

I. THE ARCHITECTURE OF THE COMPUTER AIDED SPECIFICATION SYSTEM

Considering the text editor as a simple computer aided specification system its functioning can be described relatively simply (see *Figure 1*): the a specification is accepted in interactive mode and stored in a data file. There is possibility to change the stored data, to obtain different listings. The text editors support some consistency checking of the specification as well, providing different search facilities based upon formal criteria (all occurences of a given character string, e.t.c.).

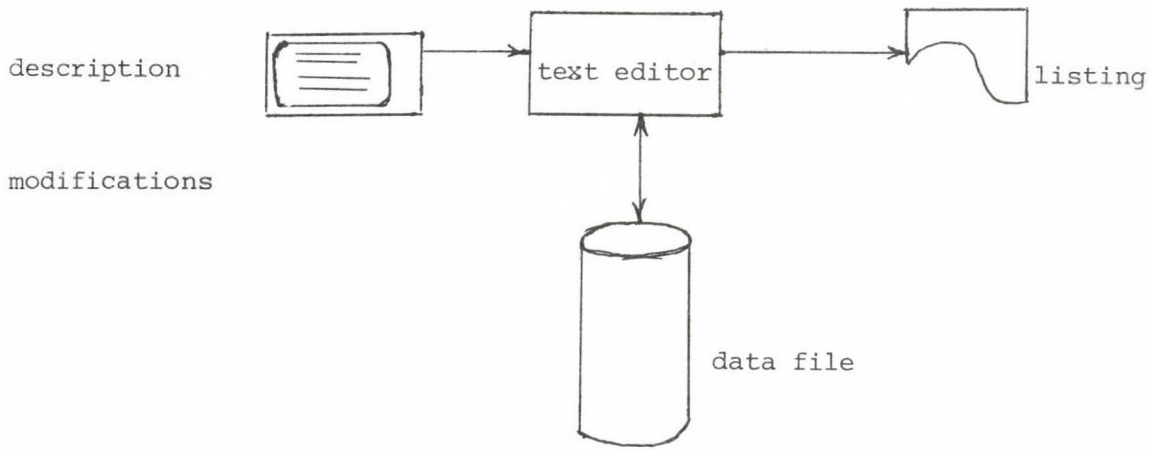


Figure 1.

The text editor

There is a principal difference between text editors and the specific computer aided specification systems: the former accepts any informal text, while the latter checks its input,

and accepts only correct sentences of a precisely defined machine analyzable language. This description language should be extremely user friendly, easy to read, self explaining (its users in general are not computer specialists) and at the same time should possess sufficient descriptive power.

These points are demonstrated by a tiny PSL [2] description fragment in *Figure 2*. The language is structured into sections (indicated by indentation in Figure 2). Each section consists of a section header (the first line of the section), which specifies the object we want to tell things about, and a section body containing statements referring to the section header. The meaning of the test can be easily understood, and it is quite clear, that is structure is simple enough to be computer analyzable.

```
process payroll system;
   uses payroll input to derive payroll output;
interface administration;
   generates payroll input;
   receives payroll summaries;
process syntax check;
   uses payroll input;
   derives payroll file;
```

Figure 2.

A PSL description fragment

The strict description language is advantageous from the point of view of the communication. It is not only the man-machine interface, which needs a formal language. In the communication among people some defense is required against misunderstanding each other's thought expressed in human language. We refer to the well known teamwork problems, or to the apparent inability of the user to explain the problem to be solved and

to understand and check the proposed solution without misinterpretation.

The formal languages - besides the offered advantages - have their own weak points. In applying a particular language to the solution of some real world problem, it is often hard to match the predefined concepts of the language with those naturally arising from the analysis of the problem. The description language must be general to be widely applicable, and at the same time should provide concepts, which are suitable to describe the essential special characteristics of any problem within the - preferably wide - range of applicability. This contradiction leads to the development of the two-level or meta specification systems [4].

This approach - recognising the above mentioned advantages and disadvantages - provides software, which instead of giving supply to only one predefined language, encourages the user to define his own language. After the definition a two-level specification system will provide all the usual advantages of the computer aided specification systems with the user defined - and therefore problem oriented - concepts.

This is similar to the abstract data type definition mechanism of some programming languages, where a type is defined by its name and characterized by interrelationships with other types (abstract operations) [3]. An object of a given type is capable of possessing those and only those relationships which are permitted for the type according to its specification. In this regard the fixed language specification systems can be compared with the programming languages without type definition facility, while the two-level systems with those programming languages, which provide it.

The architecture of the two-level specification system is shown in *Figure 3*. The meta level system provides facilites to define the specification language to be accepted by the description level system later on. This connection is realized by tables created by the definition interpreter and used by all the

modules of the description system. We note, that the description level in itself can be considered as one-level self-contained specification system, once the description language has been fixed.

As a concrete example we mention the SDLA system [5]. At the meta level the SDLA user can define:

- the conceptual framework,
- the syntax of statements used throughout the specifications,
- semantical constraints associated with the concepts.

Figure 4 shows a definition fragment. The defined language coincides with a subset of the PSL fragment of Figure 2.

```
concept process;
concept input;
concept output;
concept uses to derive (input, process, output);
        form process; uses input to derive output;
concept usage (input, process);
        form process: uses input;
```

Figure 4.
SDLA definition fragment

2. SPECIFICATION SYSTEM MODEL

First of all we note, that each specification system is a model in itself. A description language contains predefined concepts (for example in Figure 2 appear "process", "uses", e.t.c.), which are abstractions of real worls entities. Their existence is based upon the fact, that the different real world systems have enough in common to make possible a classification of their constituent parts into general types (concepts). The statements (concepts) of the PSL for example can

be used to describe any concrete information system, so they can be considered as general information system model.

The two-level specification system is an even higher level abstraction. It integrates the different specification systems into unique framework. The meta level can be considered as a specification system, modeling specification systems. Indeed, the "concept" concept of the SDLA is applicable to a wide variety of specification systems [6], as it graphs their essence - the use of abstract concepts (types) to describe the properties and the relationships of real world entities by a proper classification scheme.

The specification system can be considered as a special kind of database management system as well. It allows the user to classify, store, list and modify data via proper interface, while the software provides the technical details of storage and retrieval. A general database management system model was proposed by the ANSI/X3/SPARC committee in 1975 [7]. According to this model a database management system consists of the user interface (external scheme), details of storage (internal scheme), and the conceptual scheme, invisible from outside, but playing a most significant - if not allways explicitly stated - role in the functioning of the database management system. The conceptual scheme is an abstraction of the outside world to be modeled. The mapping between the elements of the external and the internal scheme is not direct, it is materialized by their referring to the objects of the conceptual scheme. Although the conceptual scheme may be invisible for the user, it is the very model, which determines the external and internal schemes. The conceptual scheme is the central element of the whole conception.

All three schemes have their corresponding counterpart in the computer aided specification systems. The external scheme corresponds to the specification language, the internal to the data management. The counterpart of the conceptual scheme is the method of modeling provided, the concepts available to describe the real world. In the case of the PSL these concepts are concrete object and relationship types and some semantical

constraints (see *Figure 5*).

```
type input;
type output;
type process;
type interface;
    :
relation generates (interfaces, input);
relation uses to derive (input, process, output);
relation uses (input, process);
    :
constraint uses to derive (input, process, output) implies
     uses (process, input) and derives (process, output);
```

Figure 5.

PSL conceptual scheme fragment

The definition of types is obvious (Figure 2 shows examples of their usage), but the formalization of the semantics is a harder task. For example the

```
process  P;
   uses I to derive O;
PSL description fragment, that is the
   uses to derive (I,P,O);
relationship automatically implies the
   process P;
      uses I;
      derives O;
fragment, that is the
      uses(P,I); derives (P,O);
```

relationships for any P,I and O objects. The "constraint" statement of Figure 5 describes this property of the PSL.

Another example of conceptual scheme is that of the SDLA.

On the meta level there is only one conceptual object type. Everything is expressed using

type concept (attribute 1,...,attribute n);

where all attributes are concepts as well. Without going into details we remark the proximity of this model to the relational data model [8].

It is also possible to define semantical constraints on meta level. The statement

concept $A(X_1,X_2,...,X_n)$ *implies* $B(X_i,...,X_i)$; $(k < n)$

for example is a generalization of the above mentioned PSL semantic constraint. Whenever on object of type A is created on the description level, the system automatically creates another object of type B. Semantic constraints of this type are called "implication" in the SDLA. While in the PSL it works only for a fixed relationship ("uses to derive"), on the SDLA meta level implication constraint can be defined between any two concepts.

3. SPECIFICATION LANGUAGE SEMANTICS

We can use the ANSI/SPARC model to describe the functioning of the computer aided specification system. According to the model, the processing of the arriving description may be considered as realization of two mappings - from the external scheme language (specification language) into the elements of the conceptual language (occurrences of conceptual objects), and then from the conceptual scheme language into data manipulation commands of the internal scheme. The query is the inverse of these two mappings, and the modification also can be described by them.

We define the semantics of the specification languages as a set of relationships between conceptual scheme objects. According to this definition only those relationships are

semantic, which involve actions executed between the realization of the two above mentioned mappings. For example, when a new description fragment arrives, the semantical constraints result in actions realized after mapping specification language statements into concept occurances, and before mapping these into data manipulation commands.

This definition implies the way of semantics specification. As any semantic constraint is a relationship between conceptual scheme objects, it should be defined in terms of the conceptual scheme.

There is no problem in the case of one-level specification systems. For example, the conceptual scheme of the PSL is a set of fixed types, therefore a routine executing the necessary actions may serve as definition (or realization of the definition) of a semantic constraint. The processing algorithm of the semantic constraint from Figure 5 is quite clear by itself, and the routines realizing it can be added to that part of the input analyzer which deals with the "uses to derive" relationship.

If we want to define semantics in a two-level specification system, we face additional difficulties. A semantics constraint statement can not be attached to a concrete object type, as it should operate with conceptual scheme objects (in case of the SDLA these are concepts). A formal tool, capable of defining semantics in general, not only for a given specification language, but for a wide family of specification languages is needed. As we have seen the SDLA's "implies" constraint is a generalization of a semantic property of the PSL's "uses to derive" statement. Its definition is correct - relationship between any two concepts - and the facility can be applied on meta level to specify semantic constraints in any defined description language.

There is a number of mathematically exact, but rather complicated, inpractical methods to describe general semantic constraints for the procedural (programming) language [9].

These methods don't seem to be applicable in case of description languages. It is a more realistic approach to generalize some of the semantic constraints of the existing one level specification systems based on the needs of the applications. We refer to the numerous SDLA publications ([4], [5], [6], [10], e.t.c.), which contain several examples of semantic constraint specification in concrete systems.

REFERENCES

[1] Boehm, B.W., Software Engineering. *IEEE Transactions on Computers*, C-25 (1976) 12, 1226-1241.

[2] Teichroew, D., E.A. Hershey; PSL/PSA: a Computer-aided Technique for Structure Documentation and Analysis of Information Processing Systems. *IEEE Transactions on Software Engineering*, SE-3 (1977) 1, 41-48.

[3] Liskov, B.H., A. Snyder, R. Atkinson and C. Shaffert; Abstraction mechanisms in CLU. *Communications of ACM*, August 1977.

[4] Demetrovics, J., E. Knuth, P. Rado; Specification Meta Systems. *Computer*, 15 (1982) 5, 29-35.

[5] Knuth, P. Rado, A. Toth; Preliminary Description of SDLA. *MTA SZTAKI Tanulmányok*, Budapest, 105/1980, 1-62.

[6] Knuth, E., P. Rado; Principles of Computer Aided System Description. *MTA SZTAKI Tanulmányok*, Budapest, 117/1981, 1-46.

[7] ANSI/X3/SPARC Study Group on Database Management Systems. *Interim Report*, 1975.

[8] Codd, E.F., Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, 4(1979) 4, 397-434.

[9] Donahue, J.E.; Complementary Definitions of Programming Language Semantics. *Springer Lecture Notes in Computer Science*,

[10] Knuth, E., F. Halasz, P. Rado; SDLA System Descriptor and Logical Analyzer. in Information System Design Methodologies: a Comparative Review (ed. T.W. Olle, et al.), *North-Holland*, Amstardam, 1982.

Ö S S Z E F O G L A L Á S

A LEIRÓ NYELVEK SZEMANTIKÁJÁRÓL

Radó P.

A cikk a számitógépes specifikációs rendszerek felépitését elemezve jut el a rendszerek egy széles osztályára alkalmazható modellhez. A modell alapján általános definiciót javasol a leiró nyelvek szemantikájára.

О СЕМАНТИКЕ ЯЗЫКОВ ОПИСАНИЯ

П. Радо

Анализ архитектуры систем спецификации приводит к общей модели применяемой на широкий класс таких систем. На базе этой модели предлагается общее описание семантики для языков описания.

 Közlemények 31/1984. 63-76

ПОСТРОЕНИЕ АДАПТИВНЫХ РЕКУРСИВНЫХ ФИЛЬТРОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ СГЛАЖИВАНИЯ И ПРОГНОЗИРОВАНИЯ В СИСТЕМАХ УПРАВЛЕНИЯ

ВАЛЕРИЙ САЛЫГА
НИКОЛАЙ КАРАБУТОВ
АЛЕКСАНДР ЧЕРНОГОРОВ
ИШТВАН ХАДРЕВИ

МОСКВОСКИЙ ИНСТИТУТ СТАЛИ И СПЛАВОВ

ВВЕДЕНИЕ

Цифровые методы обработки сигналов широко применяются в системах управления для решения задач сглаживания, фильтрации, прогнозироования, индентификации и управления. Так как системы управления технологическими объектами работают в условиях априорной неопределенности, то применение цифровых фильтров с фиксированными параметрами в данных условиях может оказаться малоэффективным. Поэтому в настоящее время большое распространение находят адаптивные цифровые фильтры, которые позволяют получать такую степень подавления помех, какую трудно, а иногда и невозможно, получить с помощью фильтров с фиксированными параметрами [1].

Цифровые фильтры принято делить на рекурсивные, или фильтры с бесконечной памятью, и нерекурсивные[2]. При построении адаптивных рекурсивных фильтров (АРФ) можно выделить два подхода. Один из них - непрямой - состоит из двух этапов. На первом этапе рассчитывается нерекурсивная система фильтрации с постоянными параметрами [3], на втором - осуществляется аппроксимация полученного эталонного решения адаптивным фильтром с бесконечной памятью [3-5]. Другой подход - прямой - состоит в непосредственном выборе параметров адаптивной рекурсивной системы. Примером такого устройства может служить адаптивный фильтр калмана [6].

В работе излагаются методы синтеза АРФ с помощью прямого метода Ляпунова, пути повышения качества работы полученной системы фильтрации, рассматривается возможность применения АРФ к сглаживанию нестационарных последовательностей. Полученные результаты детально рассмотрены в [7].

1. ПОСТРОЕНИЕ АДАПТИВНЫХ РЕКУРСИВНЫХ ФИЛЬТРОВ НА ОСНОВЕ ПРЯМОГО МЕТОДА ЛЯПУНОВА

Пусть в каждый момент времени измеряется аддитивная смесь

$$x_n = u_n + \xi_n, \qquad n=0,1,2,... \qquad (1)$$

где u_n - полезный сигнал, $n\Delta t$ - дискретное время, ξ_n - независимая случайная помеха, формируемая фильтром "авторегрессии-скользящего среднего" [8]:

$$\gamma_1(\nabla)\,\xi_n = \gamma_2(\nabla)\,w_n$$

где $\gamma_1(s)=1+d_1 s+d_2 s^2+\dots+d_k s^k$ - устойчивым полином степени k,

$\gamma_2(s)=1+b_1 s+b_2 s^2+\dots+b_r s^r$ - полином степени r, $r\le k$,

w_n - дискретный белый шум с $M\{w_n\}=0$, $M\{w_n^2\}=\sigma_w^2$,

$M\{\cdot\}$-знак математического ожидания. Тогда почти наверное для ξ_n справедливая оценка

$$M\{\xi^2\} = \sigma^2 < \infty \quad , \tag{2}$$

причем σ^2 неизвестна.

Оцениваемая последовательность u_n является детерминированной функцией времени с неизвестным законом изменения.

Считаем, что u_n и ξ_n разнесены по частоте, причем

$$\omega_u \ll \omega_\xi$$

Для выделения u_n из последовательности $\{x_n\}$, $n=0,1,2,\ldots$ будем применять рекурсивный фильтр вида

$$y_n = \sum_{i=1}^{i=m} \hat{a}_{i,n-1}\, y_{n-i} + \sum_{j=1}^{j=k} \hat{b}_{j,n-1}\, x_{n-j} \quad , \tag{3}$$

где $\hat{a}_{i,n-1}$, $\hat{b}_{j,n-1}$ - подстраиваемые параметры фильтра, y_n - выход фильтра, x_{n-j} - наблюдаемая последовательность в момент времени $n-j$.

Введем обозначения

$$\hat{A}^T_{n-1} = \left[\hat{A}^T_{1,n-1} \,\middle|\, \hat{B}^T_{n-1}\right] = \left[\hat{a}_{1,n-1}, \ldots, \hat{a}_{m,n-1}, \hat{b}_{1,n-1}, \ldots, \hat{b}_{k,n-1}\right]$$

$$Y^T_{n-1} = \left[Y^T_{1,n-1} \,\middle|\, X^T_{n-1}\right] = \left[y_{n-1}, \ldots, y_{n-m}, x_{n-1}, \ldots, x_{n-k}\right]$$

Тогда уравнение (3) перепишем в виде

$$y_n = \hat{A}^T_{n-1}\, Y_{n-1} \quad , \tag{4}$$

где T - знак транспонирования.

Сделаем следующее предположение: последовательность $\{u_n\}$ является выходом дискретной динамической системы

$$u_n = K_1^T (U_{n-1} - Y_{1,n-1}) + A^T Y_{n-1} , \qquad (5)$$

где $K_1 \in R^m$ - известный вектор, полином $\varphi(z)=1-k_1 z^{-1} -\ldots-k_m z^{-m}$ - является устойчивым, R - евклидово пространство, $A^T = [A_1^T \,|\, B^T]$ - вектор неизвестных параметров, $U_{n-1} \in R^m$.

Данное предположение позволяет свести задачу фильтрации x_n к задаче идентификации объекта (5) с помощью адаптивной модели (4). При этом (5) можно рассматривать как динамическую систему с эталонной моделью (вектор K_1).

Уравнение, описывающее работу системы фильтрации,имеет вид

$$e_n = K_1^T E_{1,n-1} + \Delta A_{n-1}^T Y_{n-1} , \qquad (6)$$

где $e_n \stackrel{def}{=} u_n - y_n$, $E_{1,n-1} = [e_{n-1} ,\ldots, e_{n-m}]^T$, $\Delta A_n \stackrel{def}{=} A - \hat{A}_n$.

Синтез адаптивных алгоритмов будем осуществлять с помощью функции Ляпунова

$$V_n = cM\{e_n^2\} + \tilde{\delta}^{-1} \|\Delta A_n\|^2 ,$$

где $c, \tilde{\delta} > 0, \|\Delta A_n\| = \Delta A_n^T \Delta A_n$. Так как мы налюдаем последовательность (1), то в дальнейшем под e_n будем понимать разность $x_n - y_n$.

Для устойчивости системы (6) необходимо, чтобы первая разность функции V_n удовлетворяла условию $\Delta V_n = V_{n+1} - V_n \leq 0$.
Вычисляя ΔV_n и дифференцируя по ΔA_n , получим адаптивный алгоритм подстройки вектора ΔA_n :

$$\Delta A_{n+1} = \Delta A_n - \tilde{\delta} c (K_1^T E_{1,n} Y_n + Y_n Y_n^T \Delta A_n) . \qquad (7)$$

Реализация процедуры (7) требует вычисления матрицы $Y_n Y_n^T$ и вектора ΔA_n , который нам неизвестен. Поэтому алгоритм (7) будем называть потенциальным. Из него, в частности, можно получить упрощен-

ный алгоритм

$$\Delta A_{n+1} = \Delta A_n - \gamma e_n Y_n , \qquad (8)$$

откуда следует алгоритм подстройки параметров АРФ (4):

$$A_{n+1} = A_n + \gamma e_n Y_n . \qquad (9)$$

Параметр γ вводится для обеспечения устойчивости процедуры (8) в условиях действия ненаблюдаемой помехи ξ_n. Схема адаптивного рекурсивного фильтра (4) с алгоритмом (9) показана на рис.1.

2. АДАПТИВНАЯ ФИЛЬТРАЦИЯ НЕСТАЦИОНАРНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Рассмотрим применение АРФ (4) для случая, когда $\{u_n\}$ является нестационарной последовательностью (параметры A, а может быть и K_1 в уравнении (5) зависят от времени).

Применение АРФ (4) в этом случае будет вносить временное запаздывание в получаемые оценки y_n , так как:

1) выход y_n является прогнозом по состоянию фильтра $Y_{1,n-1}$, вектору параметров $\hat{A}_{n-1}$ и входу X_{n-1} ;

2) АРФ как динамическая система имеет нелинейную фазовую характеристику и конечное время реакции на входное воздействие.

Для исключения указанного недостатка фильтра с бесконечной памятью в [9] предлагается пропускать полученную входную $\{x_n\}$ и выходную $\{y_n\}$ последовательности $(n=\overline{0,N_1})$ через двусторонний фильтр. Непосредственно данный подход к текущей фильтрации неприменим, так как требует формирования множества $G_n = \{Y_i \in R^{m+k}, i=\overline{0,n}\}$ и пропускания его через полученный к данному моменту времени АРФ в обратном порядке, что намного увеличивает время обработки и предъявляет повышенные требования к памяти

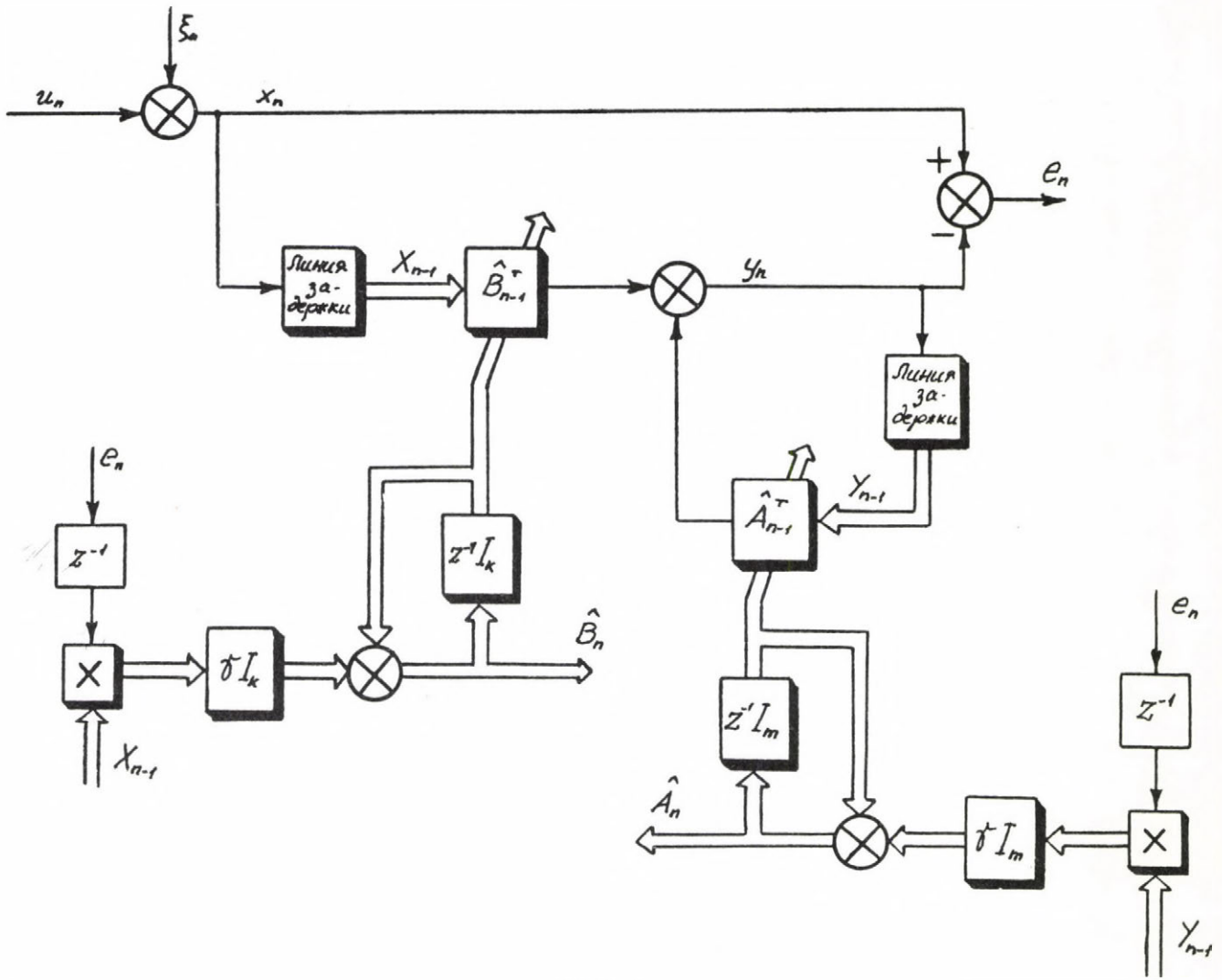

ξ_n
u_n
x_n
e_n
Линия задержки
X_{n-1}
$\hat{B}^T_{n-1}$
y_n
Линия задержки
Y_{n-1}
$\hat{A}^T_{n-1}$
z^{-1}
$z^{-1}I_k$
γI_k
$\hat{B}_n$
X_{n-1}
$z^{-1}I_m$
$\hat{A}_n$
$\gamma^* I_m$
z^{-1}
Y_{n-1}

РИС.1.

системы фильтрации.

В данной работе используется следующий подход, который назовем методом фильра точного прогноза (ФТП).

Опишем принцип построения ФТП первой степени.

Пусть в момент времени $n\in[0,N]$ получена оценка (прогноз) y_n последовательности u_n по вектору Y_{n-1} и параметрам $\hat{A}_{n-1}$. Состояние системы фильтрации в момент времени n будем характеризовать множеством

$$\Gamma_1 = \left\{ Y_{1,n-1}\in R^m,\ \hat{A}_{n-1}\in R^{m+k},\ n\in[0,N] \;\middle|\; y_n = \hat{A}_{n-1}^{T} Y_{n-1} \right.$$
$$\left. \text{при заданном входе}\quad X_{n-1}\in R^k \right\} \qquad (10)$$

Для получения точной оценки последовательности u_n с помощью АРФ (4), находящегося в момент времени $n\in[0,N]$ в состоянии Γ_1 введем преобразование Φ, имеющее вид

$$\tilde{Y}_{n-1} = \Phi\begin{bmatrix} Y_{1,n-1} \\ \hline X_{n-1} \end{bmatrix} = (zI_m \dot{+} I_k)\begin{bmatrix} Y_{1,n-1} \\ \hline X_{n-1} \end{bmatrix} = \begin{bmatrix} \tilde{Y}_{1,n-1} \\ \hline X_{n-1} \end{bmatrix}$$

где $\Phi\in R^{(m+k)\times(m+k)}$, $\dot{+}$ - знак прямой суммы матриц [10], $zy_{n-1}=y_n$, I_m и I_k - единичные матрицы соответствующих размерностей.

Тогда получение точной оценки нестационарной последовательности u_n в момент времени $n\in[0,N]$ состоит в повторном применении АРФ к вектору $\tilde{Y}_{n-1}$. При этом состояние системы фильтрации в момент времени $n\in[0,N]$ будет описываться множеством

$$\Gamma_2 = \left\{ \hat{A}_{n-1}\in R^{m+k},\ \tilde{Y}_{1,n-1}\in R^m,\ n\in[0,N] \;\middle|\; y_n = \hat{A}_{n-1}^{T} \tilde{Y}_{n-1} \right.$$
$$\left. \text{при заданном входе}\quad X_{n-1}\in R^k \right\} \qquad (11)$$

Таким образом, состояние адаптивного ФТП в момент $n\in[0,N]$ можно представить как объединение множеств (10), (11): $\Gamma=\Gamma_1\cup\Gamma_2$

Аналогично, для фильтра точного прогноза ℓ-й степени получаем

$$\Gamma = \bigcup_{i=1}^{i=\ell} \Gamma_i$$

В пространстве состояний ФТП первой степени описывается уравнением

$$\tilde{y}_n = \hat{A}^{\tau}_{1,n-1}\,\tilde{A}_{n-1}\,\bar{Y}_{n-1} + (\,\hat{A}_{1,n-1}\,I + 1\,)\,\hat{B}^{\tau}_{n-1}\,X_{n-1}\ , \qquad (12)$$

где

$$\bar{Y}^{\tau}_{n-1} = [\,y_{1,n-1}\ ,\dots,\ y_{m,n-1}\,] \in R^m\ ,\quad \bar{Y}_{n-1} = Y_{1,n-1}\ ,\quad I=[1,0,\dots,0]^{\tau} \in R^{m-1}\ ,$$

$$\tilde{A}_{n-1} = \left[\begin{array}{c|c} \hat{A}_{1,n-1} & \\ \hline I_{m-1} & 0 \end{array}\right]$$ -матрица размерности $m \times m$

$I_{m-1} \in R^{(m-1)\times(m-1)}$ - единичная матрица.

К этому уравнению необходимо еще добавить алгоритм адаптации(9).

Из уравнения (12) видно, что, если:

а) $|\hat{a}_{1,n-1}|<1$,

б) матрица $\tilde{A}_{n-1}$ устойчивая, то есть все ее собственные значения лежат внутри единичного круга $|\lambda_i(\tilde{A}_{n-1})| \leq 1\ \forall_i = \overline{1,m}$, то выход ФТП первой степени будет иметь меньшее временное запаздывание по отношению к u_n, чем выход фильтра (4). Это вытекает из следующего равенства

$$\varphi_{\text{ФТП}}(\omega) = \varphi(\omega) - \varphi_0(\omega)\ ,$$

где

$$\varphi(\omega) = \operatorname{arctg}\left\{\frac{\hat{B}^{\tau}_{n-1}\,L_k\hat{A}^{\tau}_{1,n-1}\,D_m + \hat{B}^{\tau}_{n-1}\,D_k\,(\,1 - \hat{A}^{\tau}_{1,n-1}\,L_m)}{\hat{B}^{\tau}_{n-1}\,L_k\,(\,1 - \hat{A}^{\tau}_{1,n-1}\,L_m) - \hat{B}^{\tau}_{n-1}\,D_k\hat{A}^{\tau}_{1,n-1}\,D_m}\right\}$$ -

-фазочастотная характеристика АРФ (4),

$$L^{\tau}_m = [\cos(\omega),\dots,\cos(m\omega)]\ ,\quad D^{\tau}_m = -[\sin(\omega),\dots,\sin(m\omega)]\ ,$$

$$L^{\tau}_k = [\cos(\omega),\dots,\cos(k\omega)]\ ,\quad D^{\tau}_k = -[\sin(\omega),\dots,\sin(k\omega)]$$

$\varphi_0(\omega)$ - фазочастотная характеристика элемента задержки $z:z^{-1}$ $y_n = y_{n-1}$

Для ФТП ℓ -й степени справедливо соотношение

$$\varphi_{\text{ФТП}}(\omega) = \varphi(\omega) - \ell \cdot \varphi_0(\omega) ,$$

из которого следует, что применение адаптивного фильтра точного прогноза ℓ -й степени позволяет уменьшить временное запаздывание, присущее АРФ, на ℓ шагов.

3. ПОВЫШЕНИЕ КАЧЕСТВА РАБОТЫ АРФ

Изложим способ улучшения качественных характеристик адаптивных рекурсивных фильтров, описанных в разделах 1-2. В основе данного подхода лежит метод окон [2,11], который широко применяется при проектировании цифровых нерекурсивных фильтров.

В общем виде уравнение линейного АРФ (4) можно записать в виде [7]

$$y_n = \hat{A}^T_{n-1} F Y_{n-1} \qquad (13)$$

где $F = F^A + F^B$, $F^A = \text{diag}(f^A_1(c^A_1,n,m) , \ldots , f^A_m(c^A_m,n,m))$,

$F^B = \text{diag}(f^B_1(c^B_1,n,k) , \ldots , f^B_k(c^B_k,n,k))$ -

-диагональные матрицы размерности $(m \times m)$ и $(k \times k)$ соответственно;

$f^A_i(c^A_i ,n,m)$, $f^B_j(c^B_j ,n,k)$ - действительные функции, зависящие от параметра C, текущего момента времени n и размерности векторов:

$Y_{1,n-1}$, X_{n-1} : $\dim Y_{1,n-1} = m$, $\dim X_{n-1} = k$.

На $f_i(\cdot)$ налагается условие

$$\| f_i(\cdot) \| \leq 1 \qquad \forall n \in [0,N] .$$

Подчеркнем принципиальное отличие, связанное с введением матрицы F в рекурсивные и нерекурсивные фильтры. В нерекурсивных системах матрица F

вводится для получения заданной точности аппроксимации идеального фильтра конечным нерекурсивным фильтром. При этом выбор матрицы F^{β} (так как $F^{\alpha} \equiv 0$) осуществляется путем удовлетворения тех или иных критериев в частотной области. В отличие от этого, матрица F в рекурсивных системах вводится для улучшения сглаживающих свойств, обеспечения нечувствительности выхода фильтра y_n к действующим возмущениям (помехе ξ_n). Поэтому те подходы и методы, которые применялись при выборе функции окна $f(.)$ в нерекурсивных фильтрах [11], оказываются малоэффективными в рекурсивных системах, что подтверждают результаты моделирования [7].

Временное окно $f(c,i,k)$, применяемое в нерекурсивных фильтрах, зависит от $\dim X_n$ и номера выборки x_{n-i} на интервале $[n-k,n]$ и записывается в виде

$$f(c,i,k) = c\,\tilde{f}(i,k),$$

где $C < C_0 = \text{const}$ - постоянное число, $\tilde{f}(.)$ - известная функция времени. В задачах адаптивной рекурсивной фильтрации функция $f(.)$, которую мы будем называть S-функцией, должна быть параметризована с точностью до параметра $C \in R^{m+k}$:

$$f_i^{\alpha} = f_i^{\alpha}(c_i, n, m) \quad \forall i = \overline{1,m},$$

$$f_j^{\beta} = f_j^{\beta}(c_j, n, k) \quad \forall j = \overline{1,k},$$

причем $c_{i(j)} = c_{i(j)}(n)$

Так как априорная информация о помехе ξ_n отсутствует, то при выборе вектора $C \in R^{m+k}$ могут быть предложены два подхода. Первый из них состоит в адаптации C_n по мере поступления x_n с целью сведения ошибки фильтрации к нулю. Другой путь выбора функции $f(.)$ связан с разделением множества $S = \{e_n \in R,\ n \in [0,N]\}$ на две области $S = S_1 \cup S_2$ и использованием функции

$$f(\cdot) = \begin{cases} f_1(\cdot), & e_n \in S_1 \\ f_2(\cdot), & e_n \in S_2 \end{cases} \qquad (14)$$

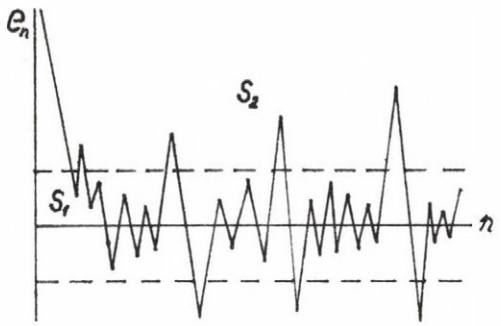


РИС.2.

Применяя аппарат R-функций [12], получаем выражения для S -функции (14)

$$f(\cdot) = f_1(\cdot) \wedge_\alpha f_2(\cdot) ,$$

где $\wedge_\alpha$ -R - конъюнкция.

Прежде чем привести результаты, показывающие влияние S - функций на качество работы АРФ, запишем (6), учитывая (13), в матричной форме

$$E_n = K E_{n-1} + \tilde{I} \Delta A_{n-1}^T F Y_{n-1} , \qquad (15)$$

где $E_{n-1}^T = [e_{1,n-1}, \ldots, e_{m,n-1}] \in R^m$, $e_{1,n-1} = e_{n-m}$, $\tilde{I} = [0,0,\ldots,0,1]$,

$$K = \left[\begin{array}{c|c} 0 & I_{m-1} \\ \hline K_m \ldots\ldots & K_1 \end{array}\right] \in R^{m \times m}$$

Считаем, что:

а) матрица К устойчивая, то есть все ее собственные числа лежат внутри единичного круга

$$|\lambda_i(k)| \le 1 \qquad \forall i = \overline{1,m} ; \qquad (16)$$

б) $$\|F\| \le \varrho < 1 \qquad (\|F\| = \max_i \lambda_i(F)) \qquad (17)$$

в) ограничение на действующее возмущение

$$M\left\{ |\Delta A_n^T Y_n| / Y_n \right\} \le \alpha \qquad \forall Y_n \neq 0, \quad n > 0 \qquad (18)$$

где $\alpha > 0$ - некоторая постоянная;

г) вектор в силу (2) ограничен в среднеквадратичном

$$M\left\{ \|Y\|^2 \right\} \le L(1 + \sigma^2) , \qquad (19)$$

где L - положительная константа.

Тогда из (15) следует, что

$$\sup \|E_n\| = \nu \qquad \text{почти наверное,} \qquad (20)$$

где ν - некоторая неслучайная константа зависящая от Eo.

Рассмотрим функцию $V_n \triangleq E_n^T P E_n$, удовлетворяющую условиям:

д) $$V_n = E_n^T P E_n > 0 \qquad (21)$$

- положительно определенная функция, допускающая бесконечно низший предел при $\|E\| \to \infty$, ($P = P^T > 0$, $P \in R^{m \times m}$) ;

е) во всем пространстве R^m выполняется следующее неравенство для

$$\Delta V_n = V_{n+1} - V_n :$$

$$\Delta V_n + \tau(E_n)[V_n - \mu] < 0 \qquad \forall E_n \neq 0,\ n \geq 0 \qquad (22)$$

для некоторой кусочно непрерывной функции

$$0 < \tau(E_n) < 1 \qquad \forall E_n \neq 0,\ n \geq 0$$

и ограничения (18)

Тогда справедлива следующая <u>теорема</u> [7]. Пусть выполняются условия (16)-(20) и существует функция V_n, удовлетворяющая (21)-(22). Тогда система (15), (8) диссипативна в целом и множество $G = \{E_n : M\{V_n\} \leq \leq \tau^{-1}(p^2 + \bar{I}^T P \bar{K} Q^{-1} \bar{K} P I) \cdot \varphi^2[\alpha + \gamma\nu L(1 + \sigma^2)]^2 = \mu\}$, где $p = \bar{I}^T P I$; τ выбирается так, чтобы матрица $\bar{K} = K/\sqrt{1-\tau}$ имела собственные числа строго внутри единичного круга;

$$\gamma_n \leq \gamma(\|E_0\|) = \gamma$$

матрица P определяется из уравнения

$$\bar{K}^T P \bar{K} - P = -\frac{1}{1-\tau} Q ,$$

$Q = Q^T > 0$ -произвольная матрица, является областью предельной ограниченности решений системы.

Диссипативность системы (6), (8) следует из следствия к приведенной теореме.

Следствие [7]. Система (6),(8) диссипативна в целом в области

$$G_1 = \{ E_n : M\{V_n\} \leq \mu_1 \},$$

где $\mu_1 = \mu(\rho)\Big|_{\rho=1}$

ЛИТЕРАТУРА

1. Уидроу Б., Гловер Дж.Р., Маккул Дж.М., Кауниц Дж. и др. Адаптивные компенсаторы помех. Принципы построения и применения. ТИИЭР, 63/1975/:12,69-99.

2. Рабинер Л., Голд Б. Теория и применение цифровой обработки сигналов. Москва, "Мир", 1978.

3. Cadrow J.A. Rekursive digital filter synthesis via gradient based algorithms.-IEEE TRANS.Un acoust.,speech and signal proces.,24/1976/:5, 349-355.

4. Парикх Д., Ахмед Н., Джонсон С.Р., Лейримон М.Г. Адаптивный алгоритм для фильтров с бесконечной импульсной характеристикой. ТИИЭР, 66/1978/:5,68-78.

5.Парикх Д., Ахмед Н. Метод последовательного регрессионного анализа применительно к адаптивной фильтрации. ТИИЭР, 66/1978/:12,83-85.

6. Фильтрация и стохастическое управление в динамических системах/под ред. К.Т.Леондеса. Москва, "Мир", 1980.

7. Карабутов Н.Н., Черногоров А.А. Синтез адаптивных цифровых фильтров для задач сглаживания и прогнозирования случайных последовательностей. Рукопись депонированная в ВИНИТИ 1 марта 1984 г., № 1197-84 Деп.

8. Бокс Д., Дженкинс Г., Анализ временных рядов. Прогноз и управление. Москва, "Мир", 1974.

9. Broome P.W. Diskrete ortonormal sequenses. - Journ. ACM,1965, V.12, № 2, pp. 151-168.

10. Маркус М., Минк Х. Обзор по теории матриц и матричных неравенств. Москва "Мир", 1972.

11. Харрис Ф., Использование окон при гармоническом анализе методом дискретного преобразования фурье. ТИИЭР,68/1089/:1,60-97.

12. Рвачев В.Л. Теория R - функций и ее приложения. Киев, "Наукова думка", 1982.

Ö S S Z E F O G L A L Á S

ADAPTIV REKURZIV SZÜRŐK KONSTRUÁLÁSA AZ IRÁNYITÁSI RENDSZEREKBEN ELŐFORDULO ELŐREBECSLÉSI ÉS SZINTÉZIS PROBLÉMÁKRA

V. Saliga, N. Karabutov, A. Chernogorov, I. Hadrevi

A cikkben a szerzők az adaptiv rekurziv szürők /ARSZ/ konstruálásának módszereit ismertetik használva a direkt Ljapunov módszert. Az ARSZ alkalmazását nem-stacionárius sorozatok "összeragasztásához" is tárgyalják. Az eredmények részletesebb tárgyalása a [7]-ben megtalálható.

THE CONSTRUCTION OF ADAPTIVE RECURSIVE FILTERS FOR THE SOLUTION OF PROBLEMS OF GLUEING AND PROGNOSIS IN CONTROL SYSTEMS

V. Saliga, N. Karabutov, A. Chernogorov, I. Hadrevi

In the paper the methods for the construction of adaptive recursive filters (ARF) is discussed using the direct method of Ljapunov. The possibility of the application of ARF to glueing non-stationary sequences is also studied. The results are discussed in details in ref. [7].

 Közlemények 31/1984. 77-94

SYSTEM DESIGN TECHNOLOGY

A. TÓTH

INORGA, Košice, Czechoslovakia

1. INTRODUCTION

Design and implementation of computerized information and control systems is a complicated process.

This process involves a very wide range of managerial and technical activities such as: problem analysis, user requirements specification, functional design, data structure design, computer-equipment selection, program development, user training, system testing, etc.

The way from the idea (i.e. requirements specification) to the final product (i.e. information or control system) often called as project life-cycle unfortunatelly is not straightforward:

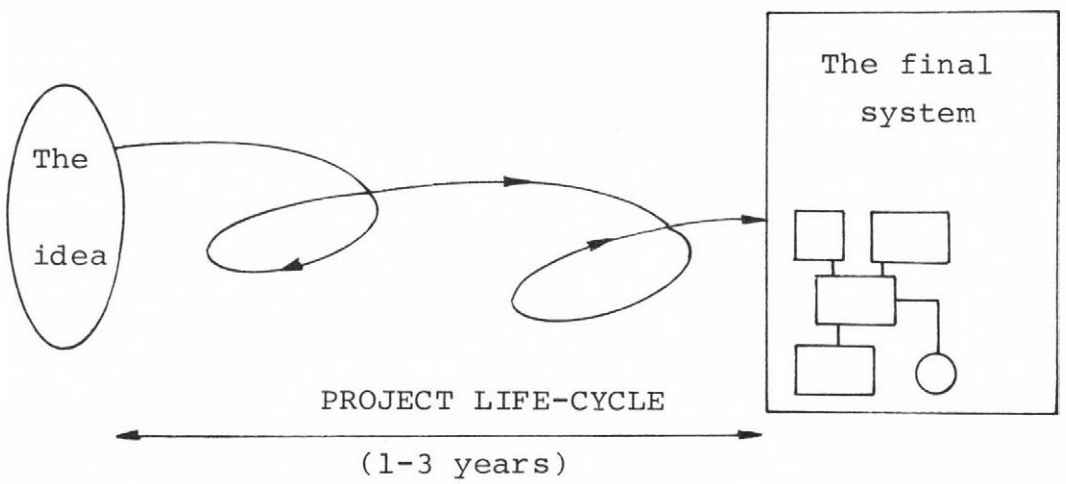


Fig.1: System development process

The succes of our projects depends on many facts. Let us mention only the most crucial ones of them:

*1 Does the developed system satisfy the functional requirements specified at the project initiation?
*2 Has the project met the original budget?
*3 Was it implemented within the given time schedule?

In the case of computerized projects the answers to these well-known questions steem from the following 'details':

What is the manpower and its professional level allocated to the project?
* To what extent is the company management and the potential users involved in the development process?
* The level of the project management itself?
* Which from the available methods and techniques are used for a particular system development activity?
* The level of project documentation?
* The level of user training?
* Do the end-users know how to use and operate the system developed for them?
* etc.

Really there are too many things we should take into account and combine them properly into a goal oriented development process. Unfortunatelly should only...

The usual case is that some of these details are neglected due to shortage of time or lack of project development experience:

we often use unefficient and/or incompatible methods in design and programming; the results of the work are incompletely docomented and certain details are known only by some 'key professionals' who can quit in the middle of the job; the project progress is not systematically evaluted so nobody knows what has been done and has to be done; after 1-2 months the users are already 'out of game' and are waiting for hopeful results; eventually, if these usually delayed results missmatch the user needs the project has to be reworked again.

Besides wasting of money and time we have to realize a dep-

ressive consequence of this ad-hoc system development approach:

There is no garantes that these mistakes will not happen again in the next project. Until there is no structured and recorded knowledge neither about the system developed nor about the development methods used this job remains

'a secret art of some experienced masters'.

In such situation young or less experienced fellows have very little chance to learn alone from the case studies of imlemented projects. They have to work in the shadow of their masters for long time as observers if they wanto to acquire this system development knowledge

This is a serious problem also in the developing countries where there is generally a lack of experienced computer professionals.

SDT (System Desing and Development Technology) offers an opportunity to resolve this problem.

2. THE SDT APPROACH

In first aproximation SDT is a structured set of proved methods combined into an activity-network for system development process.

Immediatelly one can object each project must have a specific development strategy (and activity network) so there is no general approach. This objection is true until we percieve the systems development as a heuristic process.
However, after a deeper analysis, we can 'discover' many common technical and managerial features among seemingly quite different, types of projects. For example:

* the rules of activity planning and project progress monitoring
* how to prepare a functional or data structure specification
* how to document a program algorithm, etc.

All such project independent features are incluede in the sc called SDT-skeleton. This a 250 page guide we can consider as a common model for system development. On the case study of a production control project it illustrates the usage of SDT approach. The main components of this modular document are:

AS - System development activity structure
AL - Activity list
AD - Detailed activity descriptions
AN - Activity network
WM - Optional set of working methods
WMT - methods for technical activities
WMP - methods for project management activities
DS - Project documentation standards
DSP - project progress documentation standards
DSF - final system documentation standards
DSS - sample progress and final documentation
CS - Computer-aided design facility
GL - Guidelines for SDT implementation

This skeleton is a structured knowledge-base for project planning:

* The activity list (AL) specifies WHAT has to be done
* The detailed descriptions (AD) supported by an optional set of working methods (WM) specify HOW to carry out each activity. Documentation standards (DS) specify HOW to document the result (output) of each activity and the final system developed
* The activity network (AN) determines WHEN to carry out an activity

The computer-aided design fasility (CS) is an optional support for SDT-users. This software-packege can interactively maintain all development documentation in a common project data base.

Now let as have a brief overview what is inside the SDT-skeleton:

2.1 AL - Activity list

This is a three-level hierarchical list of activities:

- development phases
- subphases and
- project steps

We have picked out some parts from the sample production control project activity list:

Project phases:

First level:

A - Problem analysis
B - Functional design
C - Data structure design
D - Process structure design
E - System implementation

subphases of B:

2nd level:

BOP - Standards training for phase B
B1 - Functional structure specification
B2 - Input/output specification
B3 - Functional feasibility test
. . . . .
B9P - Project progress monitoring

3rd level:

step for B1:

B11 - Determine time horizons
B12 - Specify functions for horizon
B13 - Describe the functions
(AD)
. . . . .

steps for B9P:

B9P1 - Check the completness of B
B9P2 - Evaluate progress in B
B9P3 - Prepare schedule for phase C
. . . . . .

Comments: - codes containing P denote project management activites.
- an average project-life-cycle consists of about 120 project steps.
- the description of B12 we can find in section AD (Fig.2 on the next page)

2.2 AD - Activity descriptions

For each activity there is a standardized activity description sheet:

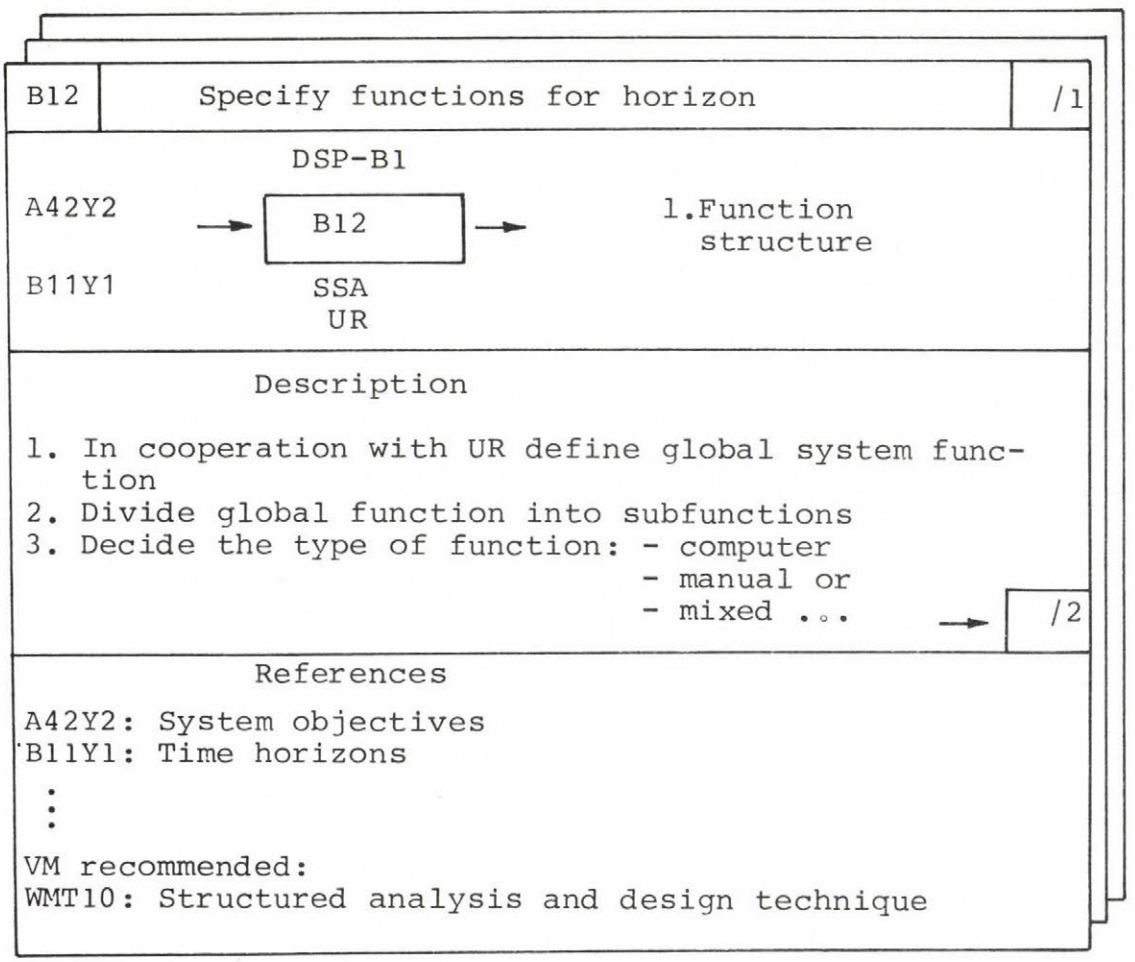


Fig.2: Activity sheets

Explanation:

* This step should carry out a senior systems analyst (SSA) in cooperation with a user representative (UR)
  They will need input documents A32Y1, B11Y1 and their result will be identified by B12Y2 (Function structure)
  This output should be docemnted according to documentation standard DSP-B1 (described in DSP component)
  The description part of the sheet can continue on the next page.

* Progress document coding: A42Y2 is the 2nd output(Y) of step 2 subphase 4 phase A.

2.3 AN - Activity network

Depending on the scale of the project this is a 2-3 level time-based precedence graph representing the activities and their scheduled duration:

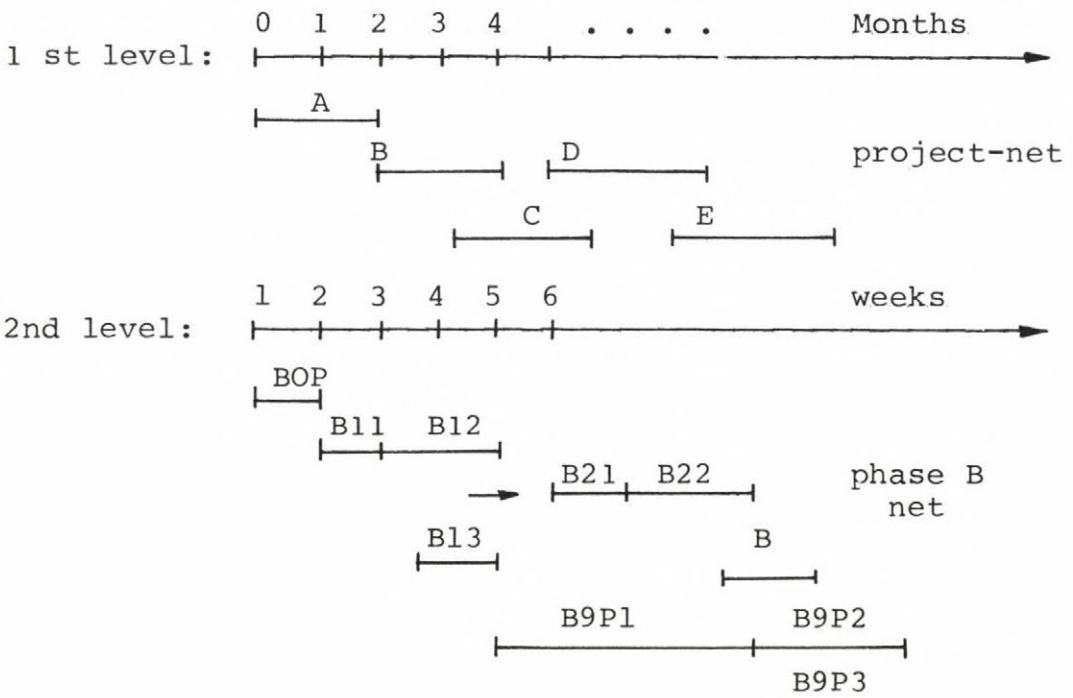


Fig.3: AN-samples

2.4 WM - Optional set of working methods

Is a collection of proved technical and project management methods and techniques. For example:

for technical activities:

WMT01: Information flow diagrams the ISAC method
WMTO2: System an program flowcharts
. . . .
WMT10: Structured analysis and design technique
WMT11: The HIPO method
..... etc.,

for managarial activities:

WMP01: Job assignment and evaluation
WMP02: Bar-charts
WMP03: Time-based networks
WMP04: Project man-power planning

These methods we can find also in a wide range of publications. But here in WM one can find them together (the most relevant ones) and already properly 'chained' to project activities: on the activity-sheets there are recommendations for WM selection.

2.5 DS - Project documentation standards

As it has been already mentioned in the Problem statement the quality of the documentation is a key condition for project success.

In SDT the documentation has two main functions:

* a medium for exact inter-team communications during system development
* a comprehensive information source for understanding and operating the system developed

For that a documentation standard is a common prescription for;

- how to record the result of work and
- how to percieve its content by other persons.

The project progress standards (DSP) are structured according to activities (AL,AD) the final system documentation standards (DSF) according to system structure.

Now let us see what is the standard for output B12Y1 included in DSP-B1 (see activity sheet, page 6):

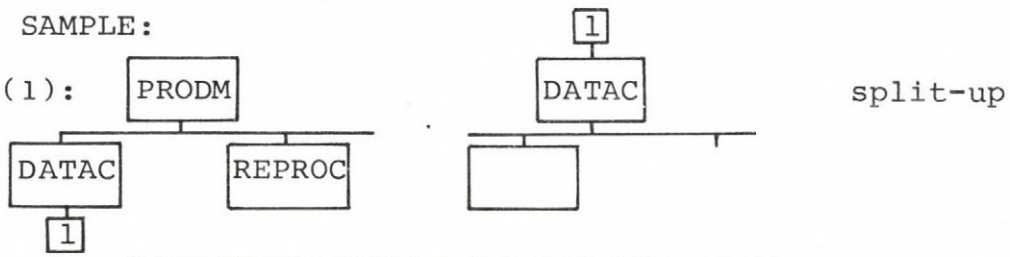
In Fig. 4A there is a prescription for B12Y1 output specification. This is a conventional standard for manual-mode design. If we decide for computer-aided design mode (see pages 10-13) then this prescription should be converted to rules of a specification language - shown on Fig.4B. In this case, however there is no need to draw hierarchical diagrams: the computer will draw and insert it into your progress and/or final documentation.

.....

DOCUMENT: B12Y1 - Function structure
COMPONENTS:
(1) Hierarchical diagram
(2) Function structure description

SAMPLE:

(1):



(2):

IDENT	NAME	TYPE	...
PRODM	Production monitoring	MIXED	
DATAC	Shop-floor data collection	MIXED	
REPROC	Report processing	COMP	
...	....		

EXPLANATION:
(1): ..... Identifier not longer than 8 letters
.....
(2): .....

Fig.4A: Project progress documentation standard sample - (conventional design)

```
FMODUL : CONCEPT 'Functional module'
           ATTRIBUTES:
fname  :     TEXT40 'Module name'
       :     MTYPE  'Module type'
           LEGAL OWNERS: HORIZON, FMODUL
fm     :   FORMAT:
             <ident>:MODUL<fname>TYPE<ftype>
           SAMPLE:
             -- ...
             -- (see screein image on Fig.6)
```

Fig.4B: Project progress documentation standard sample - (computer-aided design mode)

2.6 CS - Computer-aided design facility

The common feature of any conventional system development it produces a great amount of paperwork which has to be shared simultaneously among several professional groups:

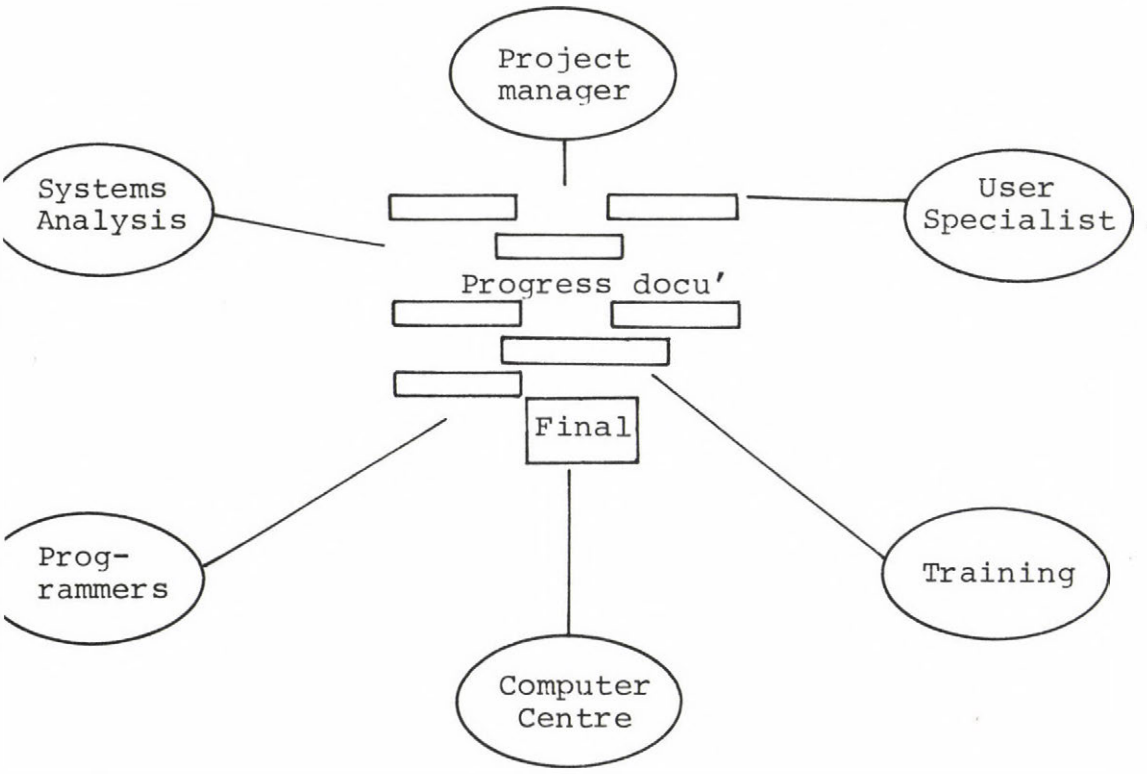


Fig.5: Conventional system design

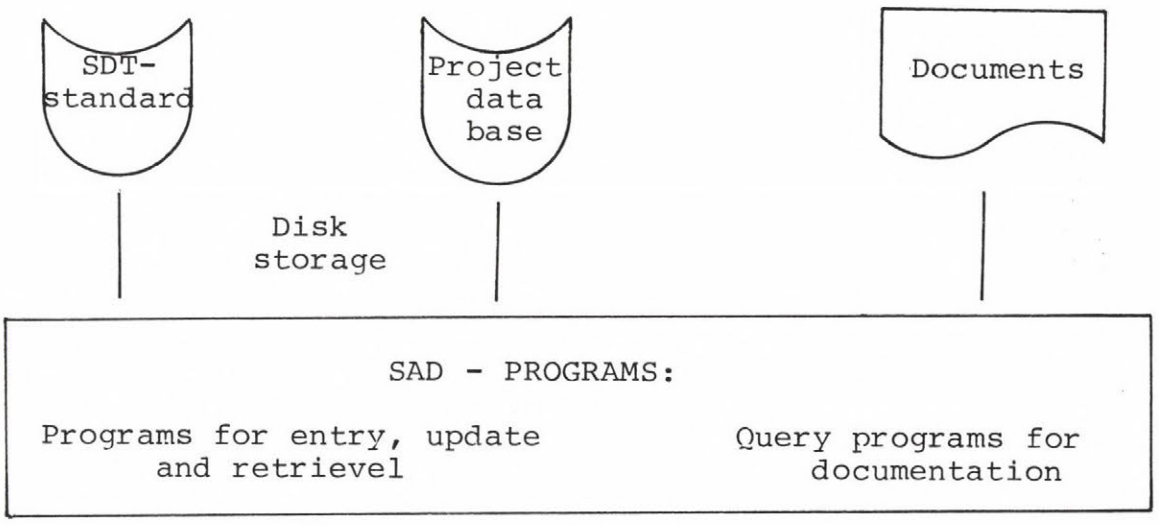
Disadvantages of this situation are well-known:

cost and time consuming clerical work; to modify reports or drafts is very difficult; the recorded results are often inconsistent; we have to search for some details in many documents; the final documentation has to be compiled from progress drafts manually and in many versions for: computer operators, users, training, etc.

The solution: in SDT you can utilize the computer to execute this clerical work!

The program support which provides these services is called:

SAD (Softvare for Advanced Design)



SAD-GUIDE
SDT-INEX

Fig.6:
Computer-aided design facility

Using SAD the outputs of project activities can be simultaneously entered and/or up-dated through on-line display terminals into a common project data base. In Fig.6 a part of B12Y1 activity output entry is illustrated.

The SDT standards here are stored also on disk storage so the SAD automatically can verify the correctness of the specifications entered before storing them into the project data base. This ensures the contents and structure of the project data base is consistent with the standards. These standards (denoted as BS2 on the screen, Fig.6) can be selected and then entered interactively as a set of expression rules at the project initiation phase. Really be means of these rules we formulate our specification language tailored to the 'dialect' of particular project phases.

For early design phases (A,B) we can select natural language-like constructs and for the final ones computer-oriented language contructs. Here is an example for a real-time program specification (phase D):

```
ocdc    :PROG'On-line converter data collection'
        :. PROGRAM INPUTS:
cbuf    :. . BUFFER'Host control buffer'
chspec  :. . . GROUP'Channel specification'
        :. PROGRAM OUTPUTS:
mbuf    :. . BUFFER'Measured values'
chnum   :. . . DATA'Channel number' PICT octal
valuem  :. . . DATA'Value measured' PICT floar
        :. COMMON DATA:
        :. . EVENT flag35 = 'Cbuf queued for satellite'
        :PROGRAM ALGORITHM:
ocdc1   :. RECEIVE cbuf
        :. . WAIT FOR flag35
        :. . CLEAR flag35
ocdc2   :. DOUNTIL'All channels processed'
        :. . SELECT chspec
        :. . . CASE 10
        :. . . . DO'Measure temperature'
        :. . . CASE 15
        :. . . . DO'Measure 02-flow rate'
        :. SEND mbur
        :END ocdc
```

Fig.7: Program specification sample

The design of any specification language by SAD-language definition facility is a relatively simple job:

- we define sentence types, (e.g.Program heading, etc.)
- some keywords, (PROG,BUFFER, etc.)
- attributes types (text,data,evenet,number, etc.)
- and finally specifiy feasible relations of these sentences (e.g. the RECEIVE sentence can be only part of PROGRAM ALGORITHM or a DO-type sentence)

So, you can define your own project-oriented design language!

For any case the DSP kit of SDT skeleton, contains a sample language for each project phase of a production control project. Addition to this standard kit there are also available further language sets from a wide spectrum of previous SDT applications. From these sources new SDT users can easily compile their own language or directly take an existing one.

Now let us summarize the benefits of this computer-aided design approach for:

* project management staff:
  - using S (Show), T (Trace), G(Graph) commands we can obtain an up-to-date project progress documentation on arbitrary object and level of details.
  - using the SAD query programs we can obtain immediatelly final or semifinal system documentation from the project data base; we can specify the format and contents of documentation required for different user levels.
* system developers:
  - the result of my work I can directly and exactly specify in A (Auto-guide) mode where the computer step by step asks for the details of my activity output
  - if I want to add or extent a specification already stored, using R (Refer), F (Format) or I (Insert) commands I can easily do it.

- if I want to change, rearrange or delete some parts of my outputs I use the U (Update), C (Change), M (Move) or D (Delete) commands.
- if I need some details on results of my colleanges I can retrieve them by S,T,G commands.
- if I forget some dialog command I type only H (Help)

* user representatives:

- we have got always up-to-data and comprehensive documentation for systems operation and for 'end-users' training
- if some system change happens we only run the document processing programs and we obtain a new version within a couple of minutes.

The above statements are typical responses of SDT/SAD users. Really comparing with the conventional style of develpment this

new facility can reduce the project costs and time up to 40% and significantly increases the quality of documentation.

For this reason this facility addition to SDT is highly recommended for project staff having access to a computer with terminal network. SAD is now running on IBM mainframes on PDP11/family and on some personal computers. To provide easy portability it was developed in FORTRAN IV programming language (the personal version in BASIC) with sensible core-memory requirements (56 KB).

2.7 GL - Guidelines for SDT implementation

SDT implementation involves a set of preparatory activities providing application of the SDT skeleton for a given project. Generally we have to complie a new skeleton let say SDT-x which should incorporate all the pecularities of our new project. Such a project dependent part is for example the Activity network (AN).

As it is shown in Fig.8 for each SDT-x skeleton component we have to decide how to create it:

*1 take it from an existing skeleton without modification?
*2 take if from an existing skeleton with modification?
*3 create a completely new one?

An existing skeleton can be either the master SDT or some dedicated SDT-1, SDT-2,... skeletons from previous SDT-driven company projects.

Existing skeletons:

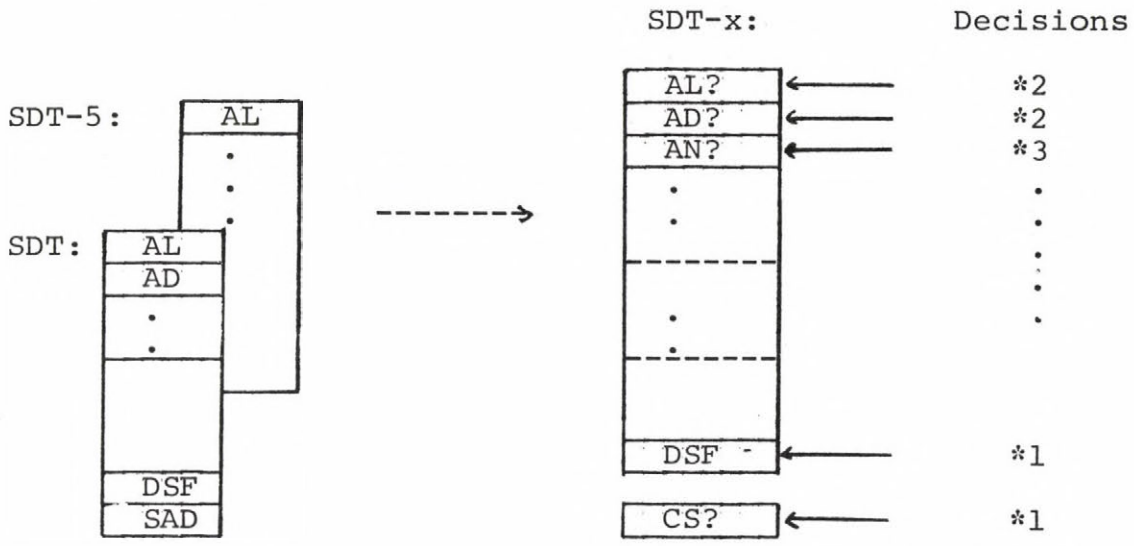


Fig.8: SDT implementation (example)

For example if we are going to prepare SDT-x for a Blast Furnace process control project we potentionally should utilize the SDT master skeleton and an (SDT-5) skeleton of Rolling Mill production control project implemented before.

In ideal case, if we keep our skeletons on computer storage a new SDT-x guide we can compile easily using a text editor.

Despite the SDT implementation is principally and technically a comprehensive process assistance of an SDT-specialist at least for the first implementation is recommemded.

Depending on project size and skeletons available takes about 0,5-3 months. The implementation team configuration recommended:

Project manager (for coordination)
Senior systems analyst
Chief programmer
SDT-specialist

3. CONCLUSIONS

The goal of this report was to provide an overview on SDT approach which gives an oporutnity to promote the current systems development practice from art to technology.

The challenge is not new: the traditional technical disciplines (as engineering or building design) have made this move decades before.

For computerized systems development this is not a moment too soon: while the capabilities of new computer hardware increase by orders of magnitude, the productivity of people creeps forward by a few percentage points a year.

The challenge is great; the reward greater.

4. FOOTNOTES

[1] SDT has been developed by Industrial Management and Automation Institute (INORGA), Czechoslovakia - Kosice during 1979-83.

[2] SDT has been implemented and verified on about 16 domestic projects and on two UNDP/UNIDO projects:

DP/CZE/77/005 (Czechoslovakia)
and DP/EGY/13/002 (EGYPT)

5. REFERENCES RECOMMENDED

[1] Teichroev D., Hershey E.A.III, PSL/PSA a computer aided Technique for Structured Documentation and Analysis of Information Processing Systems, IEEE Tranactions on SE-J, (1977).

[2] E.Knuth,F.Halász.,P.Radó: SDLA - System Descriptor and Logical Analyzer, Proc. Information Systems Design Methodolosies, North Holland 1982, pp. 143-171.

[3] Demetrovics,J., Knuth E., Radó P.: Specification meta System, IEEE on Computers, May 1982. pp.29-35.

[4] Tóth A.: SAD - Software for Computer-aided System Description on Minicomputers, IFIP TC2 Working Conference on System Description Methodolgies, Kecskemét, 1983.

[5] Békéssi A., Demetrovich J.: Contribution to the Theory of Data Base Relations, Discrete Mathematics 27 (1979) 1-10.

[6] Sobik F., Sommerfeld E.: A Graph Theoretical Approach to the Characterization of Classes of Structured Objects, Computers and Artifical Intelligence, 3 (1984), No.3., 235-247

[7] Georgescu I: A Catesorial Approach to Knowledge-based Systems, Computers and Artificial Intelligence, 3 (1984), No.2, 105-113.

Ö S S Z E F O G L A L Á S

RENDSZER TERVEZÉS TECHNOLÓGIA

Tóth Attila

A számitógépes információs és irányitási rendszerek tervezése bonyolult folyamat, amely a müszaki és irányitási tevékenységek széles skáláját ölei fel: probléma analizis, felhasználói követelményspecifikáció, funkcionális tervezés, adatszerkezet tervezés, számitógép kiválasztás, programfejlesztés, felhasználók oktatása, renszertesztelés, stb.

Az SDT bevált módszerek strukturált készlete, amelyek egymásba kapcsolódva egy olyan folytonos tevékenységhálót képeznek amely magába foglalja a rendszerfejlesztési projekt életciklusát.

ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ СИСТЕМ

Аттила Тот

Проектирование информационных и управляющих систем для ЭВМ является сложным процессом, охватывающим широкий диапазон деятельности: анализ проблем, спецификация требований пользователей, функциональное проектирование, проектирование структуры данных, выбор ЭВМ, развитие программ, обучение пользователей, проверку систем и т.п.

Система CDT является структурным набором проверенных методов, которые образуют непрерывную сеть деятельностей, охватывающую цикл жизни проекта выработки систем.

 Közlemények 31/1984, 95-113

ÜBER N-FACHE DEKOMPOSITION EINER RELATION IM CODDSCHEN RELATIONENMODELL

LE TIEN VUONG
Institut für Kybernetik und
Rechentechnik, Hanoi, Vietnam

Abstract

This paper introduces a concept: n-ary decomposition of a relation over a set of attributes. The inference rules -System of n-ary Decompositionstructure of this relation and some important properties of a full family of all decompositions have been shown. The concept of antiroot is introduced as a tool for describing the families of decompositions. The necessary and sufficient condition for a relation to be decomposable into n projections is provided in general case.

1. Einleitung

Das Konzept der Dekomposition einer Relation ist sehr wichtig im Prozess des Datenbankentwurfs. Eine Relation $R(U)$ auf einer Menge der Attributen $U=\{A_1,...,A_n\}$ wird oft in kleinere Relationen $R_1(X_1),...,R_n(X_n)$ mit $X_i \subseteq U$, $\bigcup_{i=1}^{n} X_i = U$ dekomponiert. Bei [COOD 70], [CODD 71] wird dieser Dekompositionsprozess durch die Normalisierung realisiert. Sie basiert auf den Abhängigkeiten zwischen Attributen einer Relation. Die Dekomposition wird informationsverlustsfrei genannt, wenn aus ihren Projektionen durch die natürlichen Verbundoperation die Relation R auf U wiederaufgebaut werden kann.

In der Praxis ist die Bedingung für die informationsverlustfreie Dekomposition der Relation sehr streng. Auf Grund der funktionalen und mehrwertigen Abhängigkeit haben [FAGI 77], [ARDE 79] eine notwendige und hinreichende Bedingung für die binaere Dekomposition einer Relation in 2 Projektionen bewiesen.

In [NICO78], [MEMA 79], [LE 83] haben die Autoren für die Dekomposition einer Relation in 3 Projektionen ohne Informationsverlust auf Grund der gegenseitigen Abhängigkeit gezeigt. Mit Hilfe des verlustfreien Verbundes haben [ABU 79], [BEVA 79], [RISS 77] die Dekomposition einer Relation in mehr als 3 Projektionen betrachtet.

In dieser Arbeit werden die Resultate von [ARDE 79], [LE 83] verallgemeinert. Eine Relation R auf U wird in n (>3) Projektionen ohne Informationsverlust zerlegt. Diese Dekomposition wirn n-fache Dekomposition (Abk. n-Dekompostion) der Relation genannt. Die notwendige und hinreichende Bedingung für die Existenz der n-Dekomposition einer Relation R auf U sowie die Eigenschaften dieser Dekomposition werden gezeigt.

2. Grundbegriffe

Es sei $U = \{A_1,\ldots,A_n\}$ eine endliche Menge, deren Elemente A_i, $i=\overline{1,n}$ Attribute sind. Jedes Attribut A_i hat einen entsprechenden Werte-bereich (Domain) $dom(A_i)$. Im folgenden werden die Buchstaben $A,B,\ldots$ fuer die einzelnen Attribute und $X,Y,\ldots$ für die Menge von Attributen benutzt. Eine Relation R auf U ist eine Untermenge des Kartesischen Produktes der zugehöriger Wertebereiche bzw. Wertemengen

$$R \subseteq dom(A_1)x\ldots x\ dom(A_n).$$

Es wird also eine Relation mit $R(A_1,\ldots,A_n)$ oder $R(U)$ bezeichnet. Es sei t ein Tupel von R. Dann wird mit $t[A_i]$ der Wert von t für das Attribut A_i bezeichnet. Die Projektion von t in $X\subseteq U$ wird mit $t[X]$ geschrieben. Sie ist eine Abbildung aller Attribute aus X in ihre Wertebereiche und wird X-Werte on t genannt.

Die funktionale Abhängigkeit (Abk. FA) $X\to Y$ ist in einer Relation R erfüllt, wenn aus $t_1[X]=t_2[X]$ für 2 beliebige Tupel $t_1,t_2\in R$ auch $t_1[Y]=t_2[Y]$ folgt (vgl./ARMS 74/). Eine

mehrwertige Abhängigkeit (Abk. MWA) $X \twoheadrightarrow Y$ ist in einer Relation $R(U)$ erfüllt, wenn für jedes Paar von Tupeln $t_1, t_2 \in R$ mit $t_1[X]=t_2[X]$ immer ein Tupel $t \in R$ existiert, so dass $t[XUY]=t_1[XUY]$ und $t[U\setminus(XUY)]=t_2[U\setminus(XUY)]$ gelten. Die gegenseitige Abhängigkeit (Abk. GA) $g(X,Y,Z)$ gilt in einer Relation R mit $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$ und $XUYUZ=U$, wenn für je 3 beliebige Tupel $t_1, t_2, t_3 \in R$ mit $t_1[X]=t_2[X]$, $t_2[Y]=t_3[Y]$, $t_3[Z]=t_1[Z]$ immer ein Tupel $t \in R$ existiert, so dass $t[X]=t_1[X]$, $t[Y]=t_2[Y]$, $t[Z]=t_3[Z]$ gelten (vgl. [NICO 78], [MEMA 79]).

In der vorgelegten Arbeit werden nur 2 relationale Operationen, und zwar Projektion und natürlicher Verbund benutzt. Es wird im folgenden die Vereinigung von 2 Mengen XUY in Form XY vereinbart.

3. n-fache Dekomposition

In diesem Abschnitt wird die informationsverlustfreie Dekomposition einer Relation auf der Menge der Attribute in n Projektionen $(n>3)$ betrachtet. Der Begriff der binaeren Dekomposition und der ternaeren Dekomposition (vgl.[ARDE 79], 83]) [LE 83]) wird wie folgt verallgemeinert.

Defintion 3.1

Es sei R eine Relation auf U. Eine n-fache Dekomposition von R ist ein n-Tupel $(X_1,\ldots,X_n)$ der Untermengen von U mit $X_i \subseteq U$, $\bigcup_{i=1}^{n} = U$, so dass für beliebige Tupel $t_i \in R$, $i=\overline{1,n}$

$$t_i[X_i \cap X_j] = t_j[X_i \cap X_j],\ i \neq j,\ i,j=\overline{1,n}$$

gelten, und ein Tupel $t \in R$ existiert, wobei $t[X_i]=t_i[X_i]$ $i=\overline{1,n}$ ist.

Es werden die Menge $W=\bigcup_{i \neq j} S_{ij}$ mit $S_{ij}=X_i \cap X_j$ die Wurzel der n-Dekomposition, $Z_i=X_i \setminus W$ X_i-Zweig und $W_i=X_i \cap W$ X_i-Wurzel

$i=\overline{1,n}$ der Dekomposition genannt. $X_i, i=\overline{1,n}$ sind Komponenten. Alle möglichen n-Dekompositionen einer Relation werden in eine Familie der Dekompositionen zusammengefasst. Diese Familie wird Dekompositionsstruktur $\mathcal{D}$ genannt. Dann werden die Eigenschaften der Dekompositionsstruktur der Relation R auf U im folgenden Satz formuliert.

Satz 3.1

Es sei $\mathcal{D}$ eine Familie der n-Dekompositionen einer Relation R auf U. Dann erfüllt $\mathcal{D}$ die folgenden Bedingungen:

D1. $(\emptyset,\ldots,\emptyset,U)\in\mathcal{D}$

D2. Wenn $(X_1,\ldots,X_n)\in\mathcal{D}$ ist, sind auch $(X_{\pi(1)}, X_{\pi(2)},\ldots,X_{\pi(n)})\in\mathcal{D}$, wobei $\pi:\{1,2,\ldots,n\}\to\{1,2,\ldots,n\}$ eine Permutation ist.

D3. Wenn $(X_1,\ldots,X_n)\in\mathcal{D}$, $X_i\subseteq Y\subseteq U$, $i=\overline{1,n}$ ist, dann gilt $(X_1,\ldots,X_{i-1},Y,X_{i+1},\ldots,X_n)\in\mathcal{D}$.

D4. Wenn $(X_1,\ldots,X_n)\in\mathcal{D}, X_i\subseteq X_j$, $i\neq j$ sind, ist auch $(X_1,\ldots,X_{i-1},\emptyset,X_{i+1},\ldots,X_j,\ldots,X_n)\in\mathcal{D}$.

D5. Wenn $(X_1,\ldots,X_n),(Y_1,\ldots,Y_n)\in\mathcal{D}$ mit $Y_1\cap Y_i$, $i=\overline{2,n}$ $Y_i\cap Y_j\subseteq X_i\cap X_j$, $i\neq j$, $i,j\geq 2$ sind, dann gilt $(X_1\cap Y_1, Y_2,\ldots,Y_n)\in\mathcal{D}$.

Beweis

Die Bedingungen D1 bis D4 folgen ohne Schwierigkeit aus der Definition 3.1. Hier wird nur D5 bewiesen.

Für D5 werden die Bedingungen aus der Definition 3.1 geprüft.

a. Da $Y_1\cap Y_i=X_i$, $i=\overline{2,n}$ sind, gilt dann

$$(X_1\cap Y_1)\cup\bigcup_{i=2}^{n} Y_i=(X_1\cup\bigcup_{i=2}^{n} Y_i)\cap(\bigcup_{i=1}^{n} Y_i)\supseteq(\bigcup_{i=1}^{n} X_i)\cap(\bigcup_{i=1}^{n} Y_i)=U.$$

b. Da $(X_1,\ldots,X_n)\in\mathcal{D}$ ist, existieren $t_i\in R, i=\overline{1,n}$ und $t'\in R$ mit $t_i[X_i\cap X_j]=t_j[X_i\cap X_j]$, $i\neq j$, $i,j=\overline{1,n}$ und $t'[X_i]=t_i[X_i]$, $i=\overline{1,n}$. Da $Y_1\cap Y_i=X_i$, $Y_i\cap Y_j\subseteq X_i\cap X_j$, $i\neq j$, $i,j=\overline{2,n}$

sind, sind für $(Y_1,\ldots,Y_n)\in\mathcal{D}$ zugleich $t',t_i\in R$, $i=\overline{2,n}$ und $t\in R$, die die Definition 3.1 erfüllen. Tatsächlich gelten $t'[Y_1\cap Y_j]=t_j[Y_1\cap Y_j]$, $j=\overline{2,n}$ und $t_i[Y_i\cap Y_j]=t_j[Y_i\cap Y_j]$, $i\neq j$, $i,j=\overline{2,n}$, und $t[Y_1]=t'[Y_1]$, $t[Y_i]=t_i[Y_i]$, $i=\overline{2,n}$. Zusammen mit $t_1[X_1]=t'[X_1]$ $t[Y_1]=t'[Y_1]$ gilt, das auf $X_1\cap Y_1$ $t=t'=t_1$ ist und alle Bedingungen der Definition 3.1 erf werden. So ist

$$(Y_1\cap Y_1,Y_2,\ldots,Y_n)\in\mathcal{D}\ .$$

Folgerung 3.1

D6. Ist eine der X_i, $i=\overline{1,n}$ gleich U, dann $(X_1,\ldots,X_n)\in\mathcal{D}$.

D7. Wenn $(X_1,\ldots,X_n)\in$ ist, gilt dann auch $(X_1,\ldots,X_{i-1},\emptyset,X_{i+1},\ldots,X_i\cup X_j,\ldots,X_n)\in\mathcal{D}$.

Beweis

D6 werden D1 und D3 und für D7 die Bedingungen D2, D3 und D4 zum Beweis benutzt. Daraus ergibt sich die Behauptung.

Um das im Satz 3.1 angegebene Eigenschaften-System als vollständiges System zu zeigen, wird eine vollständige Familie der n-Dekompositionen (analog wie vollständige Familie der funktionalen Abhängigkeiten (vgl. [ARMS 74])) wie folgt definiert:

Definition 3.2

Es sei U eine Menge der Attribute. Eine Familie $\mathcal{D}$ von allen n-Tupeln $(X_1,\ldots,X_n)$ der Untermengen von U mit $\bigcup_i X_i=U$, wobei sie alle Bedingungen D1 bis D5 erfüllen, wird eine vollständige Familie der n-Dekompositionen auf U genannt.

Nach der Definition 3.2 kann der Satz 3.1 wie folgt umformuliert werden: Es sei $\mathcal{D}$ eine Familie aller n-Dekompositionen der Relation R auf U. Dann ist $\mathcal{D}$ eine vollständige Familie der n-Dekompositionen auf U.

Damit einige wichtige Eigenschaften der vollständigen Familie der n-Dekompositionen untersucht werden können, wird der "duale" Begriff, nämlich die "Nichtdekomposition", Eingeführt. Dann gilt der folgende Satz.

Satz 3.2

Es sei $\mathcal{D}$ eine vollständige Familie der n-Dekompositionen der Relation R auf U. Dann gelten folgende Eigenschaften:

ND1. Wenn $(X_1,\ldots,X_n)\notin\mathcal{D}$, $\bigcup_{i=1}^{n} X_i=U$ sind, sind alle $X_i, i=\overline{1,n}$ keine ineinander geschachtelten Untermengen von U.

ND2. Wenn $\bigcup_{i=1}^{n} X_i=U$, $i=\overline{1,n}$, $(X_1,\ldots,X_nY)\notin\mathcal{D}$, $Y\subseteq U$ sind, so ist $(X_1,\ldots,X_n)\notin\mathcal{D}$.

ND3. Wenn $\bigcup_{i=1}^{n} X_i=U$, $i=\overline{1,n}$, $(X_1,\ldots,X_n)\notin\mathcal{D}$ ist, existiert eine Untermenge $S\subseteq U$, wobei $X_i\cap X_j\subseteq S$, $i\neq j$, $i,j=\overline{1,n}$, und mindestens 2 Untermengen von $\{X_1,\ldots,X_n\}$, nämlich X_p, X_q mit $X_p\nsubseteq S$, $X_q\nsubseteq S$ sind. Umgekehrt gilt fuer jedes $(Y_1,\ldots,Y_n)$ mit $\bigcup_{i=1}^{n} Y_i=U$, $Y_i\cap Y_j\subseteq S$, $i\neq j$, $i,j=\overline{1,n}$ und mindestens zwei Untermengen $Y_p, Y_q \subseteq U$ mit $Y_p\nsubseteq S$, $Y_q\nsubseteq S$ immer $(Y_1,\ldots,Y_n)\notin\mathcal{D}$.

Beweis

ND1 ist ohne Schwierigkeit einzusehen.

Zu ND2:

Es seien $\bigcup_{i=1}^{n} X_i=U$ und $\forall X_i\neq\emptyset$. Dann gilt entweder $(X_1,\ldots,X_n)\in\mathcal{D}$ oder $(X_1,\ldots,X_n)\notin\mathcal{D}$. Angenommen ist es $(X_1,X_2,\ldots,X_n)\in\mathcal{D}$. Aus D3 folgt $(X_1,\ldots,X_nY)\in\mathcal{D}$, wobei $Y\subseteq U$ ist. Dies ist ein Widerspruch zur Voraussetzung. Deshalb muss $(X_1,\ldots,X_n)\notin\mathcal{D}$ sein.

Zu ND3:

Es wird die Familie $\mathcal{B}$ von den Mengen $S\subseteq U$ betrachtet, so dass $X_i\cap X_j\subseteq S$, $i\neq j$, $i,j=\overline{1,n}$, und $(X_1S,\ldots,X_nS)\notin\mathcal{D}$ sind.

Im Spezialfall ist auch $X_i \cap X_j = S$ in der Familie. Wegen der Endlichkeit der Menge U kann eine Menge $S' \in \mathfrak{B}$ gewaehlt werden, die keine Untermenge von irgendeiner anderen in $\mathfrak{B}$ ist., d.h. S ist eine maximale Untermenge. Die X_i-Wurzel, $\overline{i=1,n}$ können wie möglich weiter vergrössert werden, bis die Resultate noch immer nicht zu $\mathfrak{D}$ gehoeren zu bleiben. Jetzt muss gezeigt werden, dass S' die Bedingung ND3 erfüllt.
O.B.d.A. wird vorausgesetzt, dass $X_i \subseteq S'$, $i=1,\overline{n-2}$, d.h. $n-2$ erste Untermengen X_i in S' enthalten sind. Weiter sei $X_{n-1} \subseteq S'$ Dann gilt $X_n S' = U$. So gilt nach D6 $(X_1,\ldots,X_n S') \in \mathfrak{D}$. Das ist ein Widerspruch. Das bedeutet, dass nicht gleichzeitig $n-1$ Untermengen von $\{x_1,\ldots,x_n\}$ in S' enthalten sind. Daraus folgt $X_{n-1} \nsubseteq S'$. Ganz analog wird auch für $X_n \nsubseteq S'$ bewiesen. Umgekehrt seien $(Y_1,\ldots,Y_n)$ mit $Y_i \cap Y_j \subseteq S'$, $i \neq j$, $\bigcup_{i=1}^{n} Y_i = U$. Es existieren mindestens zwei Untermengen von $\{Y_1,\ldots,Y_n\}$, nämlich $Y_k \nsubseteq S'$ und $Y_m \nsubseteq S'$. Dann muss gezeigt werden, dass $(Y_1,\ldots,Y_n) \notin \mathfrak{D}$ ist. Angenommen $(Y_1,\ldots,Y_n)$, die die obigen Bedingungen erfuellt, aber $(Y_1,\ldots,Y_n) \in \mathfrak{D}$. Es muss gezeigt werden, dass die Maximalität von S' verletzt wird. Zuerst wird dies für den Fall. $Y_i \cap Y_j = S'$, $i \neq j$ bewiesen, dann folgt mit Hilfe von ND2 der allgemeine Fall.

Aus $(Y_1,\ldots,Y_n) \in \mathfrak{D}$ folgt mit mehreren Anwendungen von D3, D2:

$$(X_1Y_1,\ldots,X_nY_n) \in \mathfrak{D},$$
$$(X_1Y_2,\ldots,X_nY_{n-1}) \in \mathfrak{D},$$
$$\cdots\cdots\cdots$$
$$(X_1Y_n,\ldots,X_nY_1) \in \mathfrak{D}.$$

Aus den letzten n Ausdrücken und aus $(Y_1,\ldots,Y_n) \in$ folgen mit mehreren Anwendungen von D5 und D2

$$(X_1Y_1 \cap X_2,\ldots,X_nY_n \cap Y_{n-1}) \in \mathfrak{D},$$
$$\cdots\cdots\cdots\cdots\cdots$$
$$(X_1Y_n \cap Y_{n-1},\ldots,X_nY_1 \cap Y_2) \in \mathfrak{D}.$$

Aus diesen n Ausdruecken und mit Anwendungen von D3 und den Bedingungen $Y_i \cap Y_j = S'$, $i \neq j$ gilt nach den ausfuehrlichen Umrechnungen aller Komponenten

$$(X_1 S', \ldots, X_n S') \in \mathcal{D} .$$

Das ist ein Widerspruch zur Definition der Menge S'. So muss

$$(Y_1, \ldots, Y_n) \in \mathcal{D}$$

sein. Zusammen mit ND2 gilt dann für auch allgemeine Fall.

Es ist notwendig, eine notwendige und hinreichende Bedingung fuer die Existenz einer n-Dekomposition einer Relation zu finden. Zu diesem Zweck wird der Begriff der Antiwurzel der vollständigen Familie der gen Familie der Dekomposition einge-Dekomposition eingeführt (vgl. [ARDE 79], [LE 83]). Die Antiwurzel wird wie folgt definiert:

Definition 3.3

Es sei $\mathcal{D}$ eine vollständige Familie der n-Dekompositionen auf U. Die nichttriviale Antiwurzel von $\mathcal{D}$ ist die Menge $S \subseteq U$ mit $card(U \setminus S) \geq 2$, wobei für jedes $(X_1, \ldots, X_n)$, $X_i \subseteq U$, $i = \overline{1,n}$, $\bigcup_{i=1}^{n} X_i = U$, die Bedingungen $X_i \cap X_j \subseteq S$, $i \neq j$, für mindestens zwei Mengen, z.B. $X_p \not\subseteq S$, $X_q \not\subseteq S$, $p,q \in \{1,2,\ldots n\}$ gelten, dann ist $(X_1, \ldots, X_n) \notin \mathcal{D}$.

Alle Untermengen $S \subseteq U$, die die obigen Bedingungen mit $card(U \setminus S) < 2$ erfüllen, werden triviale Antiwurzeln genannt.

$\mathcal{A}$ bezeichnet die Familie der nichttrivialen Antiwurzeln von $\mathcal{D}$ auf U. Dann gilt der

Satz 3.3

$\mathcal{D}$ kann vollstaendig mit ihrer Familie $\mathcal{A}$ beschrieben werden. Das bedeutet:

Jedes $(X_1,\ldots,X_n)$ mit $\bigcup_{i=1}^{n} X_i=U$, $X_i \subseteq U$ gehört genau dann $\mathfrak{D}$, wenn für jedes $S\in\mathfrak{A}$ mit $X_i \cap X_j \subseteq S$, $i\neq j$, $i,j=1,n$ mindestens $n-1$ dieser Untermengen (von $\{X_1,\ldots,X_n\}$) in S enthalten sind.

Beweis

Die Behauptung gilt analog zu dem folgenden: $(X_1,\ldots,X_n)$ $\notin\mathfrak{D}$ ist genau dann erfüllt, wenn eine $S\in\mathfrak{A}$ existiert, wobei $X_i\cap X_j \subseteq S$, $i\neq j$, $i,j=\overline{1,n}$, von denen mindestens zwei nicht in S enthalten sind.

Dies wird mit ND3 und der Definition 3.3 beweisen.

4. Beziehungen zwischen Abhängigkeitsstruktur und n-Dekomposition

Die Beziehungen zwischen den Datenabhängigkeitsarten und binären bzw. ternären Dekompositionen wurden in [RISS 77], [ARDE 79], [LE 83] diskutiert.

Es ist wohlbekannt, dass die Menge $\mathfrak{A}$ der nichttrivialen Antiwurzeln der Dekompositionsstructur von R auf U eine Untermenge der abgeschlossenen Menge $\mathfrak{B}$ der Attribute bzgl. der funktionalen Abhängigkeitsstructur von R ist. (Im Armstrong-Sinne, vgl. [ARMS 74]), d.h. $\mathfrak{A}\subseteq\mathfrak{B}$ (vgl. [ARDE 79], [LE 83]). Die Eigenschaften der Menge $\mathfrak{B}$ bzw. die Anzahl der maximalen Elemente der zugehoerigen FD-Structur F können ausführlich in [ARMS 74], [BÉKÉ 80] gefunden werden. Es ist nicht schwierig zu zeigen, dass alle nichttriviale Antiwurzeln der n-Dekompositionsstructur $\mathfrak{D}$ der Relation R auf U eine abgeschlossene Menge bilden.

Mit Hilfe der Resultate im obigen Abschnitt kann jede n-Dekompositionsstructur $\mathfrak{D}$ der Relation R durch ihre entsprechende Menge der Antiwurzeln $\mathfrak{A}$ beschrieben und aufgebaut werden.

Das Ziel dieses Abschnittes besteht darin, einige Eigenschaften der Antiwurzeln der n-Dekompositionen zu studieren und den Spezialfall der n-Dekomposition mit leeren Wurzeln zu entwickeln.

Es gilt der folgende

Hilfssatz 4.1

Es sei $(X_1,\ldots,X_n)\in\mathcal{D}$ eine n-Dekomposition der Relation R auf U. Dann sind alle Komponenten X_i, $i=\overline{1,n}$ die abgeschlossenen Mengen bzgl. FA-Structur dieser Relation.

Beweis

Es sei $(X_1,\ldots,X_n)\in\mathcal{D}$. Nach dem Satz 3.3 existiert eine Menge $S\subseteq U$ mit $X_i\cap X_j\subseteq S$, $i\neq j$, $i,j=\overline{1,n}$ und mindestens $n-1$ Untermengen X_i mit $X_i\subseteq S$. O.B.d.A. wird $X_i\subseteq S$, $i=\overline{1,n-1}$ vorausgesetzt. Für den Beweis ist es ausreichend $X_i\cap X_j=S$ und ND2 zu benützen. Zuerst wird für X_1 beweisen, dann gilt es auch für all X_i, $i=\overline{1,n}$.

Es muss gezeigt werden, dass $X_1\not\rightarrow A$ ist, wobei A ein Attribut aus U, $A\notin X_1$ ist. Dann muss $A\in X_j\setminus X_1$ bei festem $j\in\{2,\ldots,n\}$ sein.

Es seien zwei beliebige Tupel $t_1,t'\in R$ ausgewählt, so dass $t_1[X_1]=t'[X_1]$ aber $t_1[A]\neq t'[A]$ sind. Da S abgeschlossen und $X_i\cap X_j=S$ ist, können zwei Tupel $t_1,t_j\in R$ (für bestimmtes j) ausgewählt werden, wobei $t_1[X_1\cap X_j]=t_j[X_1\cap X_j]$ und $t_1[A]\neq t_j[A]$, $A\in X_j\setminus X_1$ sind.

Wegen $(X_1,\ldots,X_n)\in\mathcal{D}$ gilt für beliebige $t_i\in R$, $i=\overline{1,n}$ und $t'\in R$: $t_i[X_i\cap X_j]=t_j[X_i\cap X_j]$ und $t'[X_i]=t_i[X_i]$.
Bei festem j gelten auch $t'[X_j]=t_j[X_j]$ und $t'[A]=t_j[A]$.
Dann gilt

$$t'[X_1]=t_1[X_1] \text{ aber } t_1[A]\neq t'[A]=t_j[A].$$

Das bedeutet, dass X_1 abgeschlossen ist. Ganz analog gelten für

alle X_i, $i=\overline{1,n}$.
Zusammen mit ND2 gilt die Behauptung des Hilfssatzes.

Satz 4.2

Es seien R eine Relation auf U, S eine abgeschlossene Menge bzgl. der FA-Structur von R, so dass S nicht durch den Durchschnitt von zwei von S verschiedenen abgeschlossenen Mengen generiert wird. Es sei $card(U\backslash S)\geq 2$. Dann ist S eine Antiwurzel der n-Dekompositionsstructur von R.

Beweis

Die Behauptung wird mit Hilfe des Hilfssatzes 4.1 bewiesen.

Es ist bekannt, dass die Familie der abgeschlossenen Mengen bzgl. der FA-Struktur für die Durschschnittsoperation abgeschlossen itst, aber die Familie der Antiwurzeln nicht. Es gilt also nur: Wenn S_1, S_2 mit $S_1 \subseteq U$, $S_2 \subseteq U$ und $S_1 \cup S_2 \neq U$ abgeschlossen sind, dann ist $S_1 \cap S_2$ eine Antiwurzel (vgl. [ARDE 79], [LE 83]).
Für die Beziehung zwischen Funktionalen bzw. mehrwertigen Abhängigkeiten und n-Dekompositionsstructur der Relation R gelten die folgenden Verallgemeinerungen.

Satz 4.3 ([BEVA 79], [LE 83])

Wenn eine Untermenge in einem X_i-Zweig der n-Dekomposition von einer mit X_i-Zweig disjunkten Untermenge funktional abhängig ist, dann ist sie auch von der entsprechenden X_i-Wurzel abhängig.

Beweis

Der Satz kann wie folgt umformuliert werden:
Es seien $(X_1,\ldots,X_n)\in\mathcal{D}$, $S\rightarrow Y$, $S\subseteq U$, $S\cap Z_{X_i}=\emptyset$, $Y\subseteq Z_{X_i}$
dann gilt $W_{X_i}\rightarrow Z_{X_i}$ (vgl. Absch.3.).
Der Beweis läuft ganz analog wie in [LE 83].

Satz 4.4 ([MEMA 79])

Es seien R eine Relation auf U, $X_i \subseteq U$, $\bigcup_{i=1}^{n} X_i = U$. $(X_1,\ldots,X_n)$ gehört genau dann zur n-Dekompositionsstructur $\mathfrak{D}$ der Relation R auf U, wenn $(X_1 \cap W,\ldots,X_n \cap W)$ zu $\mathfrak{D}_w$ gehört und $W_i \twoheadrightarrow X_i \setminus W$ ist, wobei $\mathfrak{D}_w$ die n-Dekompositionsstructur der Relation auf der Menge $W = \bigcup_{i \neq j} (X_i \cap X_j)$, $W_i = X_i \cap W$ ist.

Beweis

Angenommen $(X_1,\ldots,X_n) \in$, $\bigcup_{i=1}^{n} X_i = U$, $W = \bigcup_{i \neq j} (X_i\ X_j)$, $i,j = \overline{1,n}$, $W_i = X_i \cap W$. Dann existieren $t_i \in R$, $i = \overline{1,n}$ und $t \in R$, wobei

$$t_i[X_i \cap X_j] = t_j[X_i \cap X_j],$$
$$t[X_j] = t_i[X_i]$$

sind.

Da $W_i \cap W_j = (X_i \cap W) \cap (X_j \cap W) = (X_i \cap X_j) \cap W \subseteq X_i \cap X_i$, und $W_i \subseteq X_i$ sind, gilt zugleich

$$t_i[W_i \cap W_j] = t_j[W_i \cap W_j] \quad \textit{und}$$
$$t[W_i] = t_i[W_i].$$

Deshalb gilt die n-Dekomposition der Relation auf der Menge $W = \bigcup_{i=1}^{n} W_i = \bigcup_{i=1}^{n} (X_i\ W)$. Mit anderen Worten:

$$(X_1 \cap W,\ldots,X_n \cap W) \in \mathfrak{D}_w.$$

Auf Grund der binären Dekomposition [ARBE 79] gilt für $(X_1,\ldots,X_n) \in \mathfrak{D}$:

$$(X_1, \bigcup_{i=1}^{n} X_i, \emptyset,\ldots,\emptyset) \in \mathfrak{D} \quad \textit{und}$$

$$(\bigcup_{i=2}^{n} X_i)\ X_1 \twoheadrightarrow X_1 \quad \text{(vgl.[FAGI 77], [ARDE 79]).}$$

Wegen $(\bigcup_{i=2}^{n} X_i)\cap X_1 = W\cap X_1 = W_1$ und $Z_1 = X_1\setminus W \subseteq X_1$ gilt dann $W_1 \twoheadrightarrow Z_1$ oder $W_1 \twoheadrightarrow X_1\setminus W$.

Es gilt analog für alle $i, i=\overline{1,n}$, d.h. $W_i \twoheadrightarrow X_i\setminus W$.
Umgekehrt sei angenommen, dass $(X_1,\ldots,X_n)$ mit
$\bigcup_{i=1}^{n} X_i = U$ und $W = \bigcup_{i\neq j} (X_i\cap X_j)$, $W_i = X_i\cap W$, $Z_i = X_i\setminus W_i$, $(W_1,\ldots,W_n)\in \mathcal{D}_w$
und $W_i \twoheadrightarrow Z_i$ gelten, so muss gezeigt werden, dass $(X_1,\ldots,X_n)\in\mathcal{D}$ ist.

Tatsächlich existieren $t_i\in R$, $i=\overline{1,n}$ und $t\in R$ mit

$$t_i[W_i\cap W_j] = t_j[W_i\cap W_j] \quad \textit{und}$$
$$t[W_i] = t_i[W_i].$$

Dann folgt wegen

$$W_i\cap W_j = X_i\cap X_j \quad W = X_i\cap X_j$$
$$t_i[X_i\cap X_j] = t_j[X_i\cap X_j]. \quad (+)$$

Aus $W_1 \twoheadrightarrow Z_1$ gelten nämlich für $t_1, t_k\in R$:

$$t_k[W_1] = t_1[W_1] \quad \textit{oder}$$
$$t_k[W\cap X_1] = t_1[W\cap X_1] \quad \textit{und} \quad t'\in R$$
$$t'[W_1Z_1] = t_1[W_1Z_1] \quad \textit{oder} \quad t'[X_1] = t_1[X_1] \quad \textit{und}$$
$$t'[W_1(U\setminus X_1)] = t_k[W_1(U\setminus X_1)].$$

Es gilt also für alle $i=\overline{1,n}$

$$t'[X_i] = t_i[X_i] \quad (++)$$

Aus (+) und (++) ergibt sich die Behauptung.

Wie oben gezeigt, die Familie der n-Dekompositionen einer Relation wird durch die Familie der entsprechenden Antiwurzeln charakterisiert. Hier wird der Spezialfall mit der leeren Wurzel, d.h. die Menge W=∅, betrachtet.

Gegeben ist eine vollständige Familie der n-Dekompositionen auf U. Sie wird in mehrere Teile X_i, $i=\overline{1,n}$ zerlegt. Auf jedem X_i wird eine neue Familie der Dekompositionen $\mathcal{D}_{X_i}$ wieder definiert. Es müssen neue Relationen R_i auf X_i konstruiert werden. Dann gilt der folgende

Satz 4.5

Es sei $\mathcal{D}$ eine vollständige Familie der Dekompositionen auf U und $(X_1,\ldots,X_n)\in\mathcal{D}$, wobei $X_i\cap X_j=\emptyset$, $i\neq j$ $i,j=\overline{1,n}$, $\bigcup_{i=1}^{n} X_i=U$ sind. Dann sind

$$\mathcal{D}_{X_i}=\{(V_1\,X_i,\ldots,V_n\,X_i)/(V_1,\ldots,V_n)\in\mathcal{D}\},\quad i=\overline{1,n}$$

die vollständigen Familien der Dekompositionen auf X_i.

Beweis

D1 bis D4 sind ohne Schwierigkeiten zu zeigen.

Zu D5:

Es seien für jedes $i=\overline{1,n}$ $(X_{i1},\ldots,X_{in})\in\mathcal{D}_{X_i}$, $(X'_{i1},\ldots,X'_{in})\in\mathcal{D}_{X_i}$ mit $X'_{i1}\cap X'_{ik}=X_{ik}$, $k=\overline{2,n}$

$X'_{ik}\cap X'_{il}\subseteq X_{ik}\cap X_{il}$, $i,1,=\overline{2,n}$, $i\neq 1$, $\bigcup_{k=1}^{n} X_{ik}=\bigcup_{k=1}^{n} X'_{ik}=X_i$.

Es muss gezeigt werden, dass $(X_{i1}\cap X'_{i1},X'_{i2},\ldots,X'_{in})\in\mathcal{D}_{X_i}$ ist.

Es existieren $(V_1,\ldots,V_n),(V'_1,\ldots,V'_n)\in\mathcal{D}$ mit

$$X_{i1}=V_1\cap X_i,\ldots,X_{in}=V_n\cap X_i\ ,$$
$$X'_{i1}=V'_1\cap X_i,\ldots,X'_{in}=V'_n\cap X_i\ .$$

Nach D3 kann $\bar{X}_i=U\setminus X_i$ zur Wurzel der Dekomposition in vergrössert werden. Das ist ganz ähnlich mit den Voraussetzungen wie

$$V_1=X_{i1}\bar{X}_i,\ V_2=X_{i2}\bar{X}_i,\ldots,V_n=X_{in}\bar{X}_i,$$

$$V_1' = X_{i1}'\bar{X}_i,\ V_2' = X_{i2}'\bar{X}_i, \ldots, V_n' = X_{in}'\bar{X}_i.$$

Dann gelten

$$V_1' \cap V_j' = X_{i1}'\bar{X}_i \cap X_{ij}'\bar{X}_i = \bar{X}_i(X_{i1}' \cap X_{ij}') = \bar{X}_i X_{ij} = V_j,\ j=\overline{2,n}$$

$$V_k' \cap V_l' = X_{ik}'\bar{X}_i \cap X_{il}'\bar{X}_i = \bar{X}_i(X_{ik}' \cap X_{il}') \subseteq V_k\ V_l\ ,\ k,l=\overline{2,n}.$$

Das bedeutet, dass $(V_1\ V_1', V_2', \ldots, V_n') \in \mathcal{D}$ gilt, dann gilt nach der Definition von $\mathcal{D}_{X_i}$ auch

$$((V_1 \cap V_1') \cap X_i,\ V_2' \cap X_i, \ldots, V_n' \cap X_i) \in \mathcal{D}_{X_i}$$

oder

$$(X_{i1} \cap X_{i1}', X_{i2}', \ldots, X_{in}') \in \mathcal{D}_{X_i}\ ,\ \text{fuer alle}\ \ i=\overline{1,n}\ .$$

Daraus folgt die Behauptung des Satzes.

Satz 4.6

Es seien n vollständige Familien der Dekompositionen $\mathcal{D}_{X_i}$ auf X_i, $i=\overline{1,n}$, wobei $\bigcup_{i=1}^{n} X_i = U$, $X_i\ X_j = \emptyset$, $i \neq j$, $i,j=\overline{1,n}$ sind. Dann ist

$$\mathcal{D} = \{(\bigcup_i X_{i1}, \ldots, \bigcup_i X_{in}) / (X_{i1}, \ldots, X_{in}) \in \mathcal{D}_{X_i}\ ,\ i=\overline{1,n}\}$$

eine vollständige Familien der Dekomposition auf U, die aus $\mathcal{D}_{X_i}$, $i=\overline{1,n}$ mit $(X_1, \ldots, X_n) \in \mathcal{D}$, $X_j = \bigcup_i X_{ij}$, $j=\overline{1,n}$, $X_i \cap X_j = \emptyset$, $i \neq j$, $i,j=\overline{1,n}$, $\bigcup_i X_i = U$ besteht.

Beweis

D1 bis D4 ist nicht schwierig zu prüfen.

Zu D5:

Angenommen $(V_1,\ldots,V_n),(V_1',\ldots,V_n')\in\mathfrak{D}$ mit $V_1'\cap V_i'=V_i$ $V_i'\ V_j'\subseteq V_i\cap V_j$, $i\neq j$, $i,j=\overline{2,n}$. Dann sind fuer X_i, $i=\overline{1,n}$

$$(V_1\cap X_i,\ldots,V_n\cap X_i)\in\mathfrak{D}_{X_i},$$

$$(V_1'\cap X_i,\ldots,V_n'\cap X_i)\in\mathfrak{D}_{X_i}$$

Seien $V_k\cap X_i=X_{ik}$, $V_k'\ X_i=X_{ik}'$, für festes i, $i\in\{1,\ldots,n\}$, $k=\overline{1,n}$. Dann gelten

$$(X_{i1},\ldots,X_{in})\in\mathfrak{D}_{X_i},$$

$$(X_{i1}',\ldots,X_{in}')\in\mathfrak{D}_{X_i}.$$

Es gelten also weiter

$$X_{i1}'\cap X_{ik}' = X_i\cap(V_1'\cap V_k') = X_i\cap V_k = X_{ik},\quad k=\overline{2,n}$$

$$X_{ik}'\cap X_{il}' = X_i\cap(V_k'\cap V_l')\subseteq X_{ik}\cap X_{il},\quad k\neq l,\ k,l=\overline{2,n}\ .$$

Die obigen Ausdruecken bedeuten, dass alle Bedingungen aus D5 für $\mathfrak{D}_{X_i}$ erfüllt sind. So sind

$$(X_{i1}\cap X_{i1}',X_{i2}',\ldots,X_{in}')=((V_1\cap X_i)\cap(V_1'\cap X_i),\ldots,V_n'\cap X_i)\in\mathfrak{D}_{X_i}.$$

Nach der Definition von $\mathfrak{D}$ gilt

$$((\bigcup_i X_{i1})\cap(\bigcup_i X_{i1}'),\ \bigcup_i X_{i2},\ldots,\bigcup_i X_{in}) =$$

$$(\bigcup_i[(V_1\cap X_i)\cap(V_1'\cap X_i)],\ldots,\bigcup_i(V_n'\cap X_i)) =$$

$$((V_1\cap V_1')\cap(\bigcup_i X_i,\ldots,V_n'\cap(\bigcup_i X_i))=$$

$$(V_1\cap V_1',V_2',\ldots,V_n')\in\mathfrak{D}\qquad \textit{wegen}\quad \bigcup_i X_i=U.$$

Ausserdem gelten noch

$$(\emptyset,\ldots,X_i,\emptyset,\ldots,\emptyset)\in\mathfrak{D}_{X_i},\quad i=\overline{1,n},\quad \text{dann gilt}$$

$$(X_1,\ldots,X_n)=(X_1 U\emptyset U..U\emptyset,\emptyset U X_2 U\emptyset U..U\emptyset,\ldots,\emptyset U\ldots U X_n)\in\mathcal{D}$$

So gilt die Behauptung.

5. Schlussfolgerung

In dieser Arbeit wurde der Begriff n-fachen Dekomposition einer Relation auf der Menge der Attributen eingefuehrt. Dieser Begriff ist äquivalent zu der n-Join Abhängigkeit. Die vollständige Familie aller möglichen n-Dekompositionen wurde mit Hilfe der entsprechenden Antiwurzeln betrachtet, die im Armstrong-Sinne abgeschlossen sind. Die notwendige und hinreichende Bedingung für ihre Existenz wurden im verallgemeinerten Fall gezeigt. Ein Spezialfall gezeigt. Ein Spezialfall der Dekomposition mit leeren Wurzel ist auch betractet. Mit diesen Resultaten ist es möglich, den Prozess des Datenbankentwurfes verzu verbessern.

Der Autor bedankt sich recht herzlich bei .
Prof. J. Demetrovics und Dr. A. Békéssy für die wertvollen Hinweise und Hilfe zu dieser Arbeit bzw. für das Lesen des Manuskriptes.

Literatur

ABU 79 : A.V.Aho,C.Beeri, J.D.Ullman: The Theory of Join in Relational Databases, ACM TODS 4,3,Sept 1979,279-314

ARMS 74 : W.W.Armstrong: Dependency structure of Data Base Relationsihps, Inf. Pro. 74, 580-583.

ARDE 79 : W.W.Armstrong, C.Delobel: Decompositions and functional Dependenceis in Relation, Université de Montreal 1979 Pub. 271

BEVA 79 : C.Beeri, M.Y.Vardi: On the Properties of total Join Dependencies, Nov. 1979.

BÉKÉ 80 : A.Békéssy, J.Demetrovics, L.Hannák, P.Frankl, Gy. Katona: On the number of maximal dependencies in a Data Basa Relation of fixed order, Disc., Math. 30, 1980, 83-88

CODD 70 : E.F.Codd: A relational Model of Data for large schared Data Bank, C.ACM 13, 6, (1970) 377-387

CODD 71 : E.F.Codd: Further normalization of the Data Base relational model, IBM-Res. RJ 909, AUG. 1971

FAGI 77 : R.Fagin: Multivalued dependencies and a new Normal Form for relational Databases, ACM TODS 2,3 (1977), 262-278

LE 83 : Le Tien Vuong: Untersuchung zur ternaeren Dekomposition einer Relation und zur anwndung der unscharfen Mengen im CRM, Diss. TU-Dresden, 1983

MEMA 79 : A.O. Mendelzon, D.Maier: Generalized mutual dependencies and the deomposition of Data Base Relations, Pro. 5th. Conf. VLDB, Rio de Janeiro, oct. 1979, 75-82.

NICO 78 : J.M.Nicolas: Mutual Dependencies and some results on undencomposable Relations, Proc. 4th. Conf. VLDB, Berlin, sept. 1978, 360-367

RISS 77 : J.Rissaenen: Independent components of Relations ACM TODS 2,4 (1977) 317-325 .

Ö S S Z E F O G L A L Ó

A CODD-FÉLE RELÁCIÓS-MODELLBEN LÉVŐ RELÁCIÓ N-SZERES DEKOMPOZICIÓJÁRÓL

Le Tien Vuong

A cikkben a szerző levezeti az n-szeres dekompozició fogalmát és a dekompoziciók teljes családjának bizonyos fontos tulajdonságaira mutat rá. Megadja a dekomponálhatóság szükséges és elégséges feltételét.

ON THE N-ARY DECOMPOSITION OF A RELATION IN THE CODD'S RELATION-MODEL

In the paper the concept of n-ary decomposition of a relation over a set of attributes is introduced. Some important properties of a full family of all decompositions are shown. The necessary and sufficient condition for a relation to be decomposable into n projections is given.

ОБ N-АРНОМ РАЗЛОЖЕНИИ СООТНОШЕНИЯ В РЕЛЯЦИОННОЙ МОДЕЛИ КОДДА

Ле Тиен Вуонг

Автор вводит понятие n-арного разложения и доказывает некоторые важные свойства полного семейства разложений. Он задает необходимое и достаточное условие разложимости.

 Közlemények 31/1984 115-151

О СУЩЕСТВЕННО МИНИМАЛЬНЫХ ТС-КЛОНАХ НА ТРЕХЭЛЕМЕНТНОМ МНОЖЕСТВЕ

Деметрович Я., Мальцев И.А.

Введение

Пусть A-конечное множество, f-n - местная операция на A. Операция f удовлетворяет термальному условию, если для любого i, $0 \le i \le n$, и любых $x,y,a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_n,b_1,\ldots,b_{i-1},b_{i+1},\ldots,b_n$ из A

из

$$f(a_1,\ldots,a_{i-1},x,a_{i+1},\ldots,a_n)=f(b_1,\ldots,b_{i-1},x,b_{i+1},\ldots b_n)$$

следует

$$f(a_1,\ldots,a_{i-1},y,a_{i+1},\ldots,a_n)=f(b_1,\ldots,b_{i-1},y,b_{i+1},\ldots,b_n).$$

Операции, удовлетворяющие термальному условию, будем называть ТС-операциями.

Множество всех операций на множестве A обозначим через O_A. Предитеративной алгеброй Поста над множеством A называется алгебра $P^*_A = \langle O_A,\zeta,\tau,\Delta,*\rangle$ [5] типа/1,1,1,1,2/ со следующим образом определенными операциями:

$$(\zeta f)(x_1,x_2,\ldots,x_n)=f(x_2,x_3,\ldots,x_n,x_1),$$

$$(\tau f)(x_1,x_2,\ldots,x_n)=f(x_2,x_1,x_3,\ldots,x_n),$$

$$(\Delta f)(x_1,x_2,\ldots,x_n)=f(x_1,x_1,x_2,\ldots,x_{n-1}),$$

$$(f*g)(x_1,x_2,\ldots,x_{n+m-1})=f(g(x_1,\ldots,x_m),x_{m+1},\ldots,x_{n+m-1}).$$

Клонами называются подалгебры алгебры P^*_A, содержащие операцию $e^2_2(x_1,x_2)=x_2$. Клон, состоящий только из ТС-операций, называется ТС-клоном.

Термальное условие является обобщением некоторых очевидных свойств унарных операций и линейных операция в векторных прост-

ранствах. Оно играет существенную роль при изучении некоторых классов многообразий. Более подробную информацию читатель найдет в работах Бермана и Маккензи [3] и Тэйлора [12].

Каждый ТС-клон на множестве A содержится в некотором максимальном ТС-клоне. На двухэлементном множестве имеется один максимальный ТС-клон, на трехэлементном - два, на множестве из четырех элементов имеется уже 25 максимальных ТС-клонов. При $2<|A|<\omega$ существует ТС-клон, имеющий счетное число подклонов [8], число подклонов каждого максимального ТС-клона не более чем счетно [3].

Операция f из O_A, имеющая **n** переменных, существенно зависит от своей i-той переменной, если в A найдутся такие $a_1,\ldots,a_{i-1}$, $a_{i+1},\ldots,a_n,b,c$, что

$$f(a_1,\ldots,a_{i-1},b,a_{i+1},\ldots,a_n)\neq f(a_1,\ldots,a_{i-1},c,a_{i+1},\ldots,a_n).$$

Операцией называется существенно m-местной, если она существенно зависит ровно от m переменных. Операции, существенно зависящие более чем от одной переменной, называются существенно многоместными.

Клон называется существенно минимальным, если он содержит существенно многоместные операции, но все его собственные подклоны состоят только из существенно одноместных операций. /Мачида [10]/. Минимальным называется клон, имеющий только один собственный подклон - подклон E , порождаемый операцией e_2^2.

В дальнейшем везде предполагается, что $A=\{0,1,2\}$, сложение всегда ведется по mod 2, если не оговорено противное. На A имеется два максимальных ТС-клона: L и B. Клон L состоит из линейных операций, т.е. операций, представимых в виде $a_o+a_1x_1+\ldots+a_nx_n$, сложение и умножение ведется по mod 3. Решетка подклонов клона L конечна и описана Деметровичем и Бадьинским [1, 2]. Клон B состоит из одноместных операций и операций, представимых в виде $f_o(f_1(x_1)+\ldots+f_n(x_n))$, где слежение ведет-

ся по mod 2, а операции $f_o, f_1, \ldots, f_n$ одноместные. Впервые этот клон упомянут в работе Бурле [4]. Описание решетки подклонов клона Бурле дает полное решение следующей проблемы: найти все ТС-клоны на трехэлементом множестве. Авторами решена более скромная проблема: построена решетка всех подклонов клона, образованного операциями, принадлежащими клону Бурле, и принимающими значение 0 и 1, и операциями из E. Вид этой решетки позволяет сделать заключение, что решетка подклонов клона Бурле устроена сложнее известной решетки Поста клонов на двухэлементном множестве [11, 14]. С другой стороны, в качестве следствия получено полное описание минимальных и существенно минимальных подклонов клона Бурле. Отметим, что некоторые общие свойства клона Бурле изучались в работах [6-9].

1. Строение фундаментальной решетки $L(F_\infty^{\zeta\Psi})$

Восемь одноместных функций, принимающих значение 0 и 1, и одна функция, принимающая три значения играют существенную роль в дальнейших рассуждениях /таблица 1/.

x	ζ	Ψ	γ	δ	$\bar{\zeta}$	$\bar{\Psi}$	c_o	c_1	λ
0	0	0	0	1	1	1	0	1	1
1	1	1	0	1	0	0	0	1	0
2	0	1	1	0	1	0	0	1	2

Таблица 1.

Для простоты иногда будем писать $\gamma+\gamma+\gamma$, $\zeta+\gamma$ и т.п. вместо $\gamma(x_1)+\gamma(x_2)+\gamma(x_3)$, $\zeta(x_1)+\gamma(x_2)$ и 0, 1 вместо c_o, c_1.

Клон, образованный операциями из клона Бурле, принимающий значения 0, 1, и операциями из E, обозначим через Z. Через L(K) обозначим решетку подклонов клона K. Для элементов решетки L(Z) иногда будут вводиться специальные обозначения, а иногда будут указываться отличные от e_2^2 элементы базиса /операция e_2^2 входит в каждый базис/. Поскольку клон вместе с каждой операцией f содержит также все операции, получающиеся из f с добав-

лением и изъятием несущественных переменных, в большинстве случаев будет предполагаться, что все переменные рассматриваемых операций существенные.

Операция λ отображает множество A на себя. Сопоставляя каждой операции f из Z операцию

$$f^{\alpha}(x_1,\ldots,x_n)=\lambda(f(\lambda^{-1}(x_1),\ldots,\lambda^{-1}(x_n))),$$

получим отображение $\alpha: f \to f^{\alpha}$, которое является автоморфизмом клона Z. Операцию f^{α} будем называть двойственной к операции f. Подклоны клона Z назовем двойственными, если один переводится в другой автоморфизмом α. Очевидно, операции Ψ, δ, $\bar{\Psi}$, c_1 двойственны операциям ζ, γ, $\bar{\zeta}$, c_0.

Пусть $J^0_{\ell 0}$ - клон, порождаемый операциями e^2_2 и $\sum_{i=1}^{\ell}\gamma(x_i)$. Так как $\gamma(0)=\gamma(1)$, то очевидно, что клон $J^0_{\ell 0}$ не содержит операций, существенно зависящих более чем от ℓ переменных. Он содержит лишь операции вида $\sum_{i=1}^{m}\gamma(x_i)$, $m \le \ell$, операции, получающиеся из указанных добавлением фиктивных переменных, константу c_0 и операции из E. Получаем возрастающую цепочку клонов $E \subset J^0_{00} \subset J^0_{10} \subset \ldots$, пределом которой является клон $J^0_{\infty 0}$, содержащий все суммы $\gamma+\ldots+\gamma$ /Рис. 1/.

К операции $\sum_{i=1}^{\ell}\gamma(x_i)$ двойственной является операция $1+\sum_{i=1}^{\ell}\gamma(x_i)$. Получаем цепочку двойственных клонов $E \subset J^1_{00} \subset J^1_{01} \subset \ldots$ с пределом $J^1_{0\infty}$. /Обозначая через J^0_{00} (J^1_{00}) клон, порождаемый операциями e^2_2 и c_0 (c_1). /

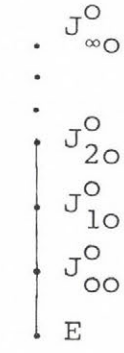


Рис. 1.

Обозначим через $J_{\ell m}$ клон, порождаемый совместно элементами клонов $J^0_{\ell 0}$ и J^1_{0m}. Ввиду свойства операций γ и δ очевидно, что $J_{\ell m}$ содержит лишь те операции, которые входят либо в $J^0_{\ell 0}$, либо в J^1_{0m}, т.е. $J_{\ell m}$ является простым объединением

клонов $J^{o}_{\ell o}$ и J^{1}_{om}. Получаем решетку $L(W)$ /рис. 2/.

Пусть $e = \Delta e^2_2$, E_ζ и E_Ψ - клоны, порождаемые элементами e^2_2, ζ и e^2_2, Ψ соответственно. Так как операции e, ζ, Ψ оставляют элементы множетсва $\{0, 1\}$ неподвижными, $\gamma(e) = \gamma$, $\delta(e) = \delta$, $\gamma(\zeta) = \gamma(\Psi) = c_o$, $\delta(\zeta) = \delta(\Psi) = c_1$, то множества вида $K_1 \cup K_2 \cup K_3$, где $K_1 \in \{J^{o}_{\ell o}, J^{1}_{om}, J_{\ell m}\}$, K_2, $K_3 \in \{E_\zeta, E_\Psi\}$ являются клонами. Получаем решетку $(W_{\zeta,\Psi})$ /рис. 3/. Как частный случай получаем решетку подполугрупп полугруппы, порождаемой операциями ζ, Ψ, γ, δ, e /рис. 4/.

Каждый клон, содержащий $\bar{\zeta}$, содержит также $\zeta = \bar{\zeta} * \bar{\zeta}$. Клон $J_{\ell m}$ порождается операциями $f(x_1,\ldots,x_\ell) = \sum_{i=1}^{\ell} \gamma(x_i)$ и $g(x_1,\ldots,x_m) = 1 + \sum_{i=1}^{m} \gamma(x_i)$. Пусть $n = \max(\ell, m)$, F^{ζ}_n - клон, порождаемый элементами f, g, $\bar{\zeta}$. Так как $\bar{\zeta}(f(x_1,\ldots,x_\ell)) = 1 + f(x_1,\ldots,x_\ell)$, $\bar{\zeta}(g(x_1,\ldots,x_m)) = 1 + g(x_1,\ldots,x_m)$, то $F^{\zeta}_n \supset J^{\zeta}_{nn}$.

Так как $F^{\zeta}_n \setminus J^{\zeta}_{nn}$ содержит лишь операции, отличающиеся от $\bar{\zeta}$ несущественными переменными, то между клонами F^{ζ}_n и J^{ζ}_{nn} нет промежуточного. Двойственным к F^{ζ}_n является клон F^{Ψ}_n. Объединение дает $F^{\zeta\Psi}_n$. Пределом цепочки $F^{\zeta\Psi}_o \subset F^{\zeta\Psi}_1 \subset$ является $F^{\zeta\Psi}_\infty$. Фрагмент решетки $(F^{\zeta\Psi}_\infty)$ показано на рис. 5. Подрешеткой этой решетки является решетка подполугрупп всех одноместных операций клона Z /рис. 6/.

2. Решетка подклонов клона Z

Оставшаяся часть решетки $L(Z)$ содержит конечное число элементов и имеет менее регулярное строение. Пусть L_ζ - клон, порождаемый операциями e^2_2 и $\zeta(x) + \zeta(y) + \zeta(z)$, и пусть $f_n(x_1,\ldots,x_n) = \zeta(x_1) + \ldots + \zeta(x_n)$. Так как $\zeta * f_n = f_n$, $f_n * f_m = f_{n+m-1}$, $\Delta f_n = f_{n-2}$, то L_ζ содержит E, операции f_1, f_3, f_5,..., и все операции, отличающиеся от указанных несущественными переменными. Единственным максимальным подклоном клона L_ζ является E_ζ, порождаемый $\{\zeta, e^2_2\}$. Двойственным к L_ζ является клон L_Ψ.

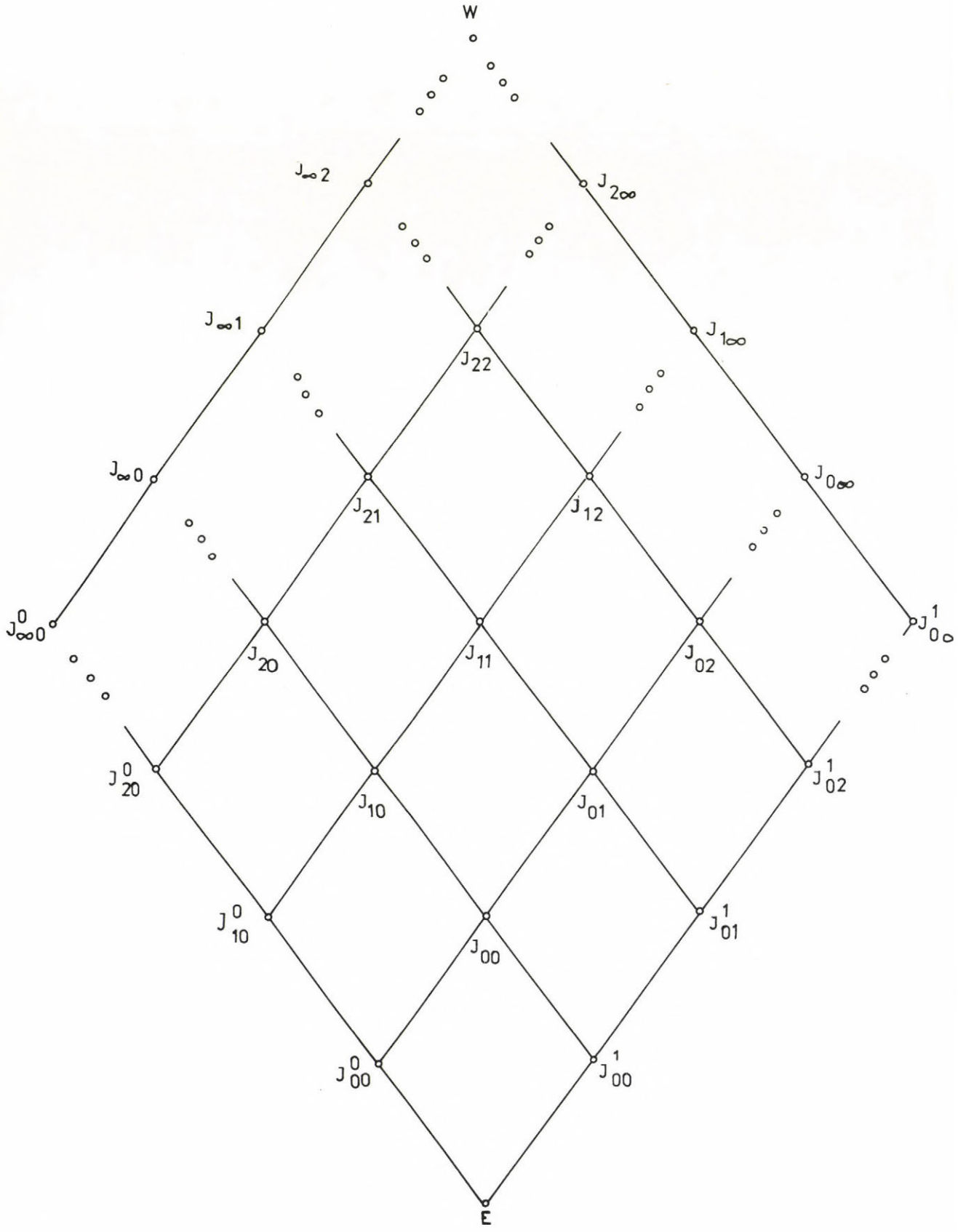


Рисунок 2.
/Решётка L(W)/

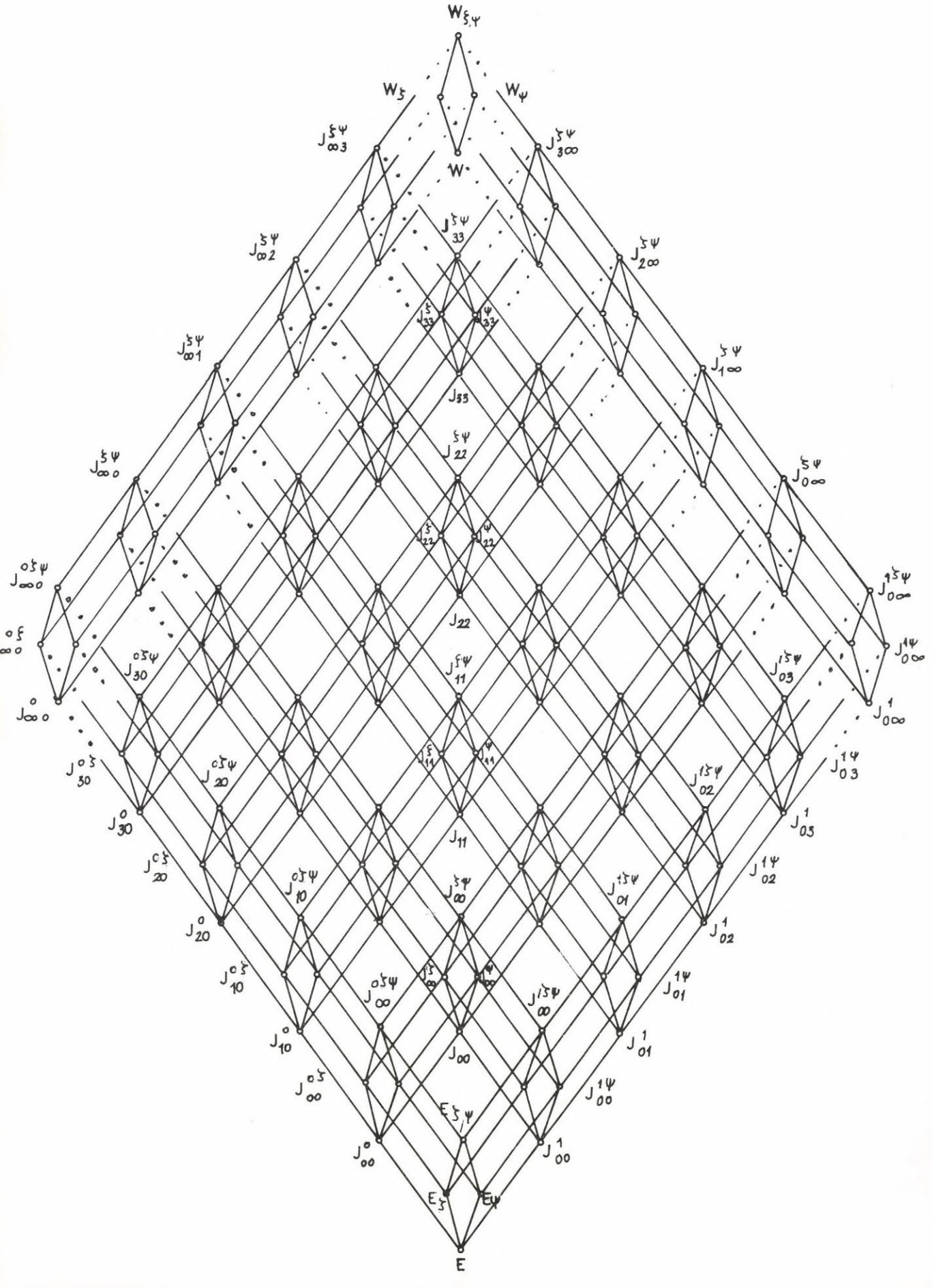


Рисунок 3

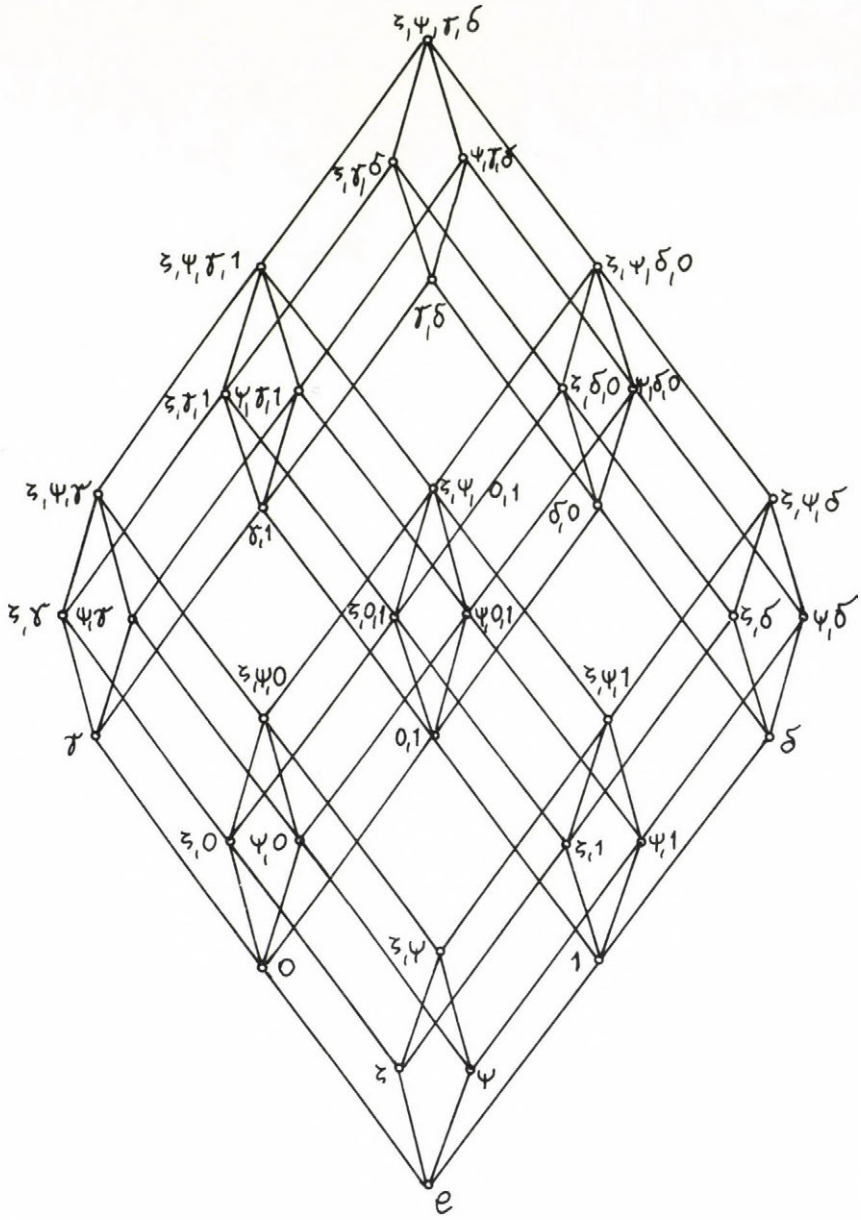
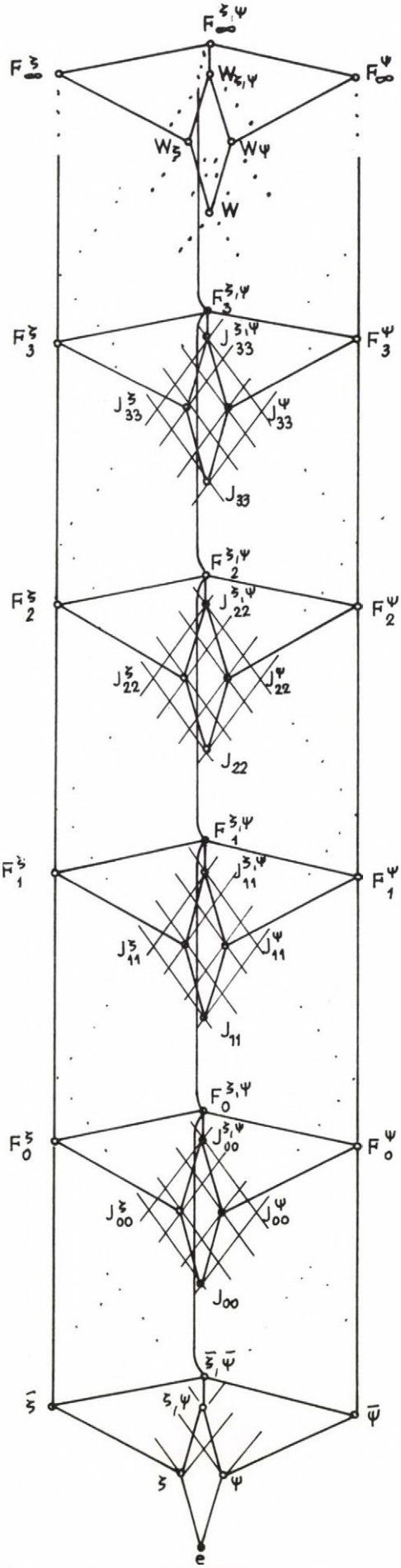

ξ,ψ,γ,δ
ξ,γ,δ
ψ,γ,δ
ξ,ψ,γ,1
ξ,ψ,δ,0
γ,δ
ξ,γ,1
ψ,γ,1
ξ,δ,0
ψ,δ,0
ξ,ψ,γ
ξ,ψ,0,1
ξ,ψ,δ
γ,1
δ,0
ξ,γ
ψ,γ
ξ,0,1
ψ,0,1
ξ,δ
ψ,δ
ξ,ψ,0
ξ,ψ,1
γ
0,1
δ
ξ,0
ψ,0
ξ,1
ψ,1
0
ξ,ψ
1
ξ
ψ
e

Рисунок 4.


$F_{\infty}^{\xi,\psi}$
F_{∞}^{ξ}
F_{∞}^{ψ}
$W_{\xi,\psi}$
W_{ξ}
W_{ψ}
W
$F_3^{\xi,\psi}$
F_3^{ξ}
F_3^{ψ}
$J_{33}^{\xi,\psi}$
J_{33}^{ξ}
J_{33}^{ψ}
J_{33}
$F_2^{\xi,\psi}$
F_2^{ξ}
F_2^{ψ}
$J_{22}^{\xi,\psi}$
J_{22}^{ξ}
J_{22}^{ψ}
J_{22}
$F_1^{\xi,\psi}$
F_1^{ξ}
F_1^{ψ}
$J_{11}^{\xi,\psi}$
J_{11}^{ξ}
J_{11}^{ψ}
J_{11}
$F_0^{\xi,\psi}$
F_0^{ξ}
F_0^{ψ}
$J_{00}^{\xi,\psi}$
J_{00}^{ξ}
J_{00}^{ψ}
J_{00}
$\bar{\xi},\bar{\psi}$
$\bar{\xi}$
$\bar{\psi}$
ξ,ψ
ξ
ψ
e

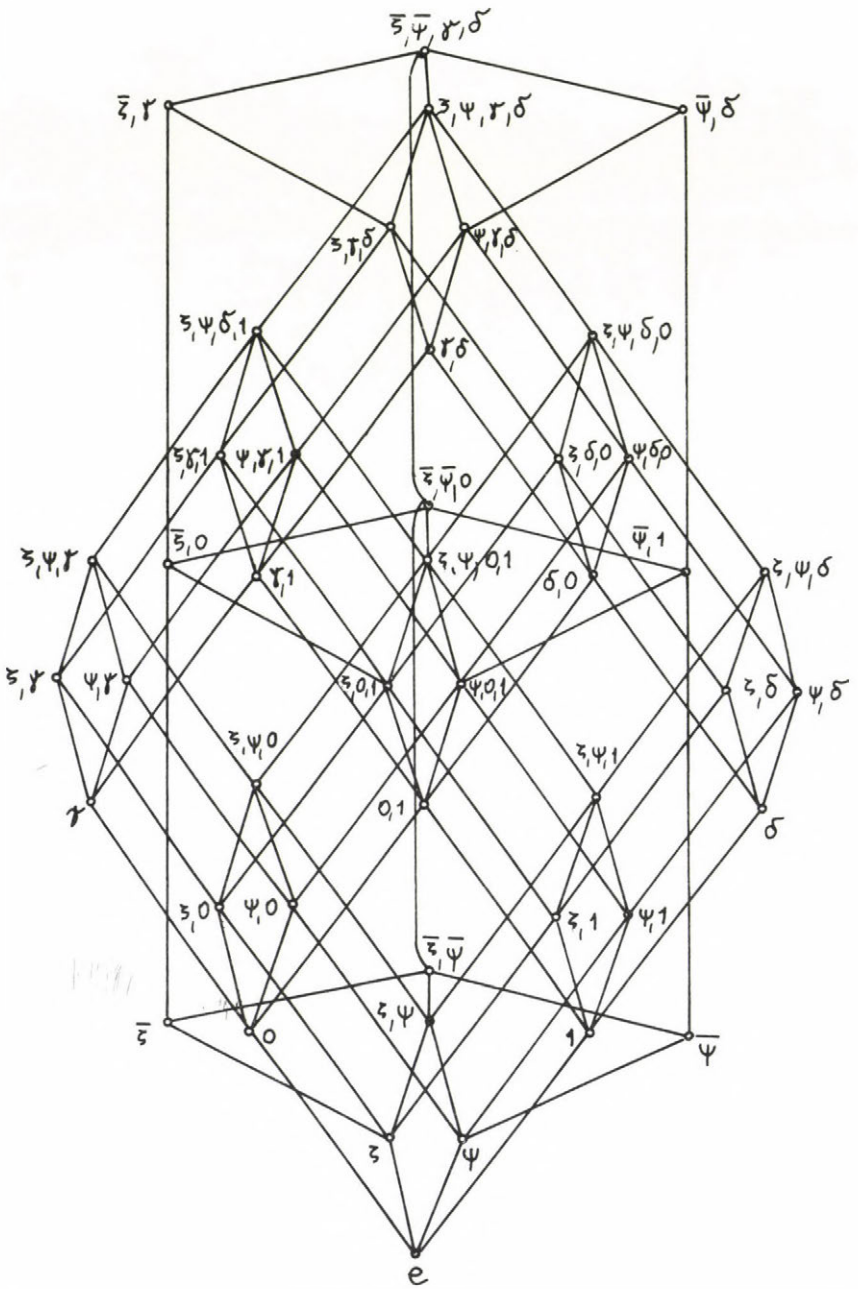

ξ,ψ,γ,δ
ξ,γ
ξ,ψ,γ,δ
ψ,δ
ξ,γ,δ
ψ,γ,δ
ξ,ψ,δ,1
ξ,ψ,δ,0
γ,δ
ξ,δ,1
ψ,γ,1
ξ,δ,0
ψ,δ,0
ξ,ψ,0
ξ,0
ψ,1
ξ,ψ,γ
ξ,ψ,δ
ξ,ψ,0,1
δ,0
γ,1
ξ,γ
ψ,γ
ξ,0,1
ψ,0,1
ξ,δ
ψ,δ
ξ,ψ,0
ξ,ψ,1
γ
0,1
δ
ξ,0
ψ,0
ξ,1
ψ,1
ξ,ψ
ξ
0
1
ψ
ξ
ψ
e

Рисунок 6.

Клон S_ζ, порождаемый операцией $f_2(x,y)=\zeta(x)+\zeta(y)$, содержит операции $f_3=f_2*f_2$ и $c_o=\Delta f_2$, поэтому S_ζ содержит все f_i, $i=1,2,\ldots$. Максимальных подклонов имеется два: L_ζ и клон с базисом $\{\zeta,c_o,e_2^2\}$. Двойственным является клон S'_ψ, порождаемый операцией $1+\psi+\psi$, двойственной к $\zeta+\zeta$.
Клон L'_ζ порождается операциями e_2^2, f_3 и $1+f_3$. Для любых m и n справедливо $f_m*(1+f_n)=1+f_{m+n-1}=(1+f_n)*f_m$, $(1+f_m)*(1+f_n)=f_{m+n-1}$, $\Delta(1+f_n)=1+f_{n-2}$. Учитывая указанные ранее свойства операции f_m видно, что L'_ζ содержит операции $f_1,f_3,\ldots,\bar{\zeta}=1+f_1$, $1+f_3,\ldots$. Максимальных подклонов два: L_ζ и клон с базисом $\{\bar{\zeta}, e_2^2\}$. Двойственным является клон L'_ψ.

Клон S'_ζ порождается базисом e_2^2, $1+f_2$. Так как $(1+f_2)*(1+f_2)=f_3$, то он содержит операции $f_1,f_3,\ldots,1,1+f_2,1+f_4,\ldots$. Максимальных подклонов два: L_ζ и клон с базисом $\{e_2^2, c_1, \zeta\}$. Двойственный клон - S_ψ, порождаемый операцией $\psi+\psi$.

Операции f_2, $1+f_2$ порождают клон U_ζ. Очевидно, U_ζ содержит все элементы клонов S_ζ, L'_ζ и S'_ζ, которые являются его максимальными подклонами. Двойственным является клон U_ψ.

Пусть $g_n(x_1,..,x_n)=\gamma(x_1)+\ldots+\gamma(x_n)$. Для любой операции $h\in Z$ справедливо $\zeta*h=h$, $\psi*h=h$, $\gamma*h=0$. Так как $(\zeta+g_n)*(\zeta+g_m)=\zeta+g_{n+m-1}$, $(\psi+g_n)*(\psi+g_m)=\psi+g_{n+m-1}$, $\Delta(\zeta+g_n)=\psi+g_{n-1}$, $\Delta(\psi+g_n)=\zeta+g_{n-1}$, то клон $L_{\zeta+\gamma}$ порождаемый базисом $\{e_2^2, \zeta+\gamma\}$, содержит лишь операции e, $\zeta+g_n$, $\psi+g_n$, $n=0,1,\ldots$, и операции, отличающиеся от указанных несущественными переменными. Максимальный подклон один, имеет базис $\{e_2^2, \zeta, \psi\}$. Клон самодвойственный.

Клон $L_{\zeta,\psi}$ порождается базисом $\{e_2^2, f_3, \bar{f}_3\}$, где $\bar{f}_n(x_1,..,x_n)=$

$= \psi(x_1)+\ldots+\psi(x_n)$. Так как

$$\Delta(\gamma+\gamma)=\Delta(\zeta+\zeta)=\Delta(\psi+\psi)=0,\ \Delta(\zeta+\gamma)=\psi,\ \Delta(\psi+\gamma)=\zeta, \qquad /1/$$

то $L_{\zeta,\psi}$ содержит суммы вида $\sum_{i=1}^{m}\gamma(x_{ji})+\sum_{i=1}^{n}\zeta(x_{\ell i})+\sum_{i=1}^{k}\psi(x_{ti})$, где сумма n+k нечетная. Максимальными подклонами являются клоны L_ζ, L_ψ, $L_{\zeta+\gamma}$. Клон самодвойственный.

Клон $L_{\zeta+\delta}$ порождается операцией $\zeta+\delta=1+\zeta+\gamma$. Так как $(\zeta+\delta)*(\zeta+\delta)=\zeta+\gamma+\gamma=\zeta+g_2$, то в нем содержатся все операции клона с базисом $\zeta+\gamma$. Так как

$$\Delta(\zeta+\delta)=\bar{\psi},\ \Delta(\psi+\delta)=\bar{\zeta},\ \Delta(\bar{\zeta}+\delta)=\psi,\ \Delta(\bar{\psi}+\delta)=\zeta,\ \Delta(\bar{\zeta}+\gamma)=\bar{\psi},\ \Delta(\bar{\psi}+\gamma)=\bar{\zeta}, \qquad /2/$$

то клон содержит также операции $\bar{\zeta}+g_n$, $\bar{\psi}+g_n$, $n=0,1\ldots$. Указанные выше равенства, а также равенства $\bar{\zeta}*(\bar{\zeta}+\gamma)=1+\bar{\zeta}+\gamma$, $\bar{\zeta}*(\bar{\zeta}+\delta)=1+\bar{\zeta}+\delta=\bar{\zeta}+\gamma$ показывают, что клон является максимальным в рассматриваемом клоне. Другой максимальный клон - клон с базисом $\{e_2^2,\ \bar{\zeta},\ \bar{\psi}\}$. Действительно, добавление к этому базису любой существенно многоместной функции из клона $L_{\zeta+\delta}$ дает базис клона $\zeta+\delta$. Клон самодвойственный.

Клон $L'_{\zeta,\psi}$ порождается базисом $\{e_2^2,\ 1+\zeta+\psi+\psi\}$. Обозначая через Δ^n результат n-кратного применения операции Δ, получаем $(1+\zeta+\psi+\psi)*(1+\zeta+\psi+\psi)=\zeta+\psi+\psi+\psi+\psi$, $\Delta^3(\zeta+\psi+\psi+\psi+\psi)=\gamma+\psi$, $(1+\zeta+\psi+\psi)*(\gamma+\psi)=1+\gamma+\psi+\psi+\psi$, $\Delta^2(1+\gamma+\psi+\psi+\psi)=1+\gamma+\psi=\delta+\psi$, поэтому $U'_{\zeta,\psi}$ содержит все операции из клона $L_{\zeta+\delta}$ в том числе ζ и ψ. Далее, $(1+\zeta+\psi+\psi)*\psi=1+\psi+\psi+\psi$, $\Delta^2(\zeta+\psi+\psi+\psi+\psi)*\psi=\psi+\psi+\psi$, поэтому $L'_{\zeta,\psi}\supset L'_\zeta$. Аналогично получаем двойственное включение $L'_{\zeta,\psi}\supset L'_\psi$. Так как $L'_{\zeta,\psi}$ содержит операции $f^3=\zeta+\zeta+\zeta$, $\bar{f}^3=\psi+\psi+\psi$, образующие базис клона $L_{\zeta,\psi}$, то $L'_{\zeta,\psi}\supset L_{\zeta,\psi}$. Ввиду равенств /1/ и /2/ клон $L'_{\zeta,\psi}$ содержит все суммы вида

$$\sum_{i=1}^{n_1}\gamma(x_{ji})+\sum_{i=1}^{n_2}\zeta(x_{\ell i})+\sum_{i=1}^{n_3}\bar{\zeta}(x_{mi})+\sum_{i=1}^{n_4}\psi(x_{pi})+\sum_{i=1}^{n_5}\bar{\psi}(x_{ti}) \qquad /3/$$

где $n_2+n_3+n_4+n_5=2k+1$. Клон не содержит γ, поэтому не совпадает с Z.

Найдем теперь максимальные подклоны клона $L'_{\zeta,\Psi}$. Каждая операция f из $L'_{\zeta,\Psi} \setminus L_{\zeta+\delta}$ является суммой вида /3/, в которой по крайней мере три согласных отличны от γ. Отождествляя нужное число раз переменные в парах однотипных слагаемых и в парах γ, ζ, δ, $\bar{\zeta}$, $\bar{\gamma}$, Ψ, $\bar{\delta}$, $\bar{\Psi}$, мы получим одну из следующих операций:

$$\zeta+\zeta+\zeta,\ 1+\zeta+\zeta+\zeta=\bar{\zeta}+\zeta+\zeta,\ \zeta+\zeta+\Psi,\ \bar{\zeta}+\zeta+\Psi,$$
$$\zeta+\Psi+\Psi,\ \bar{\zeta}+\Psi+\Psi,\ \Psi+\Psi+\Psi,\ \bar{\Psi}+\Psi+\Psi. \qquad /4/$$

Каждая из этих операций вместе с принадлежащими клону $L_{\zeta+\delta}$ операциями ζ, $\bar{\zeta}$, Ψ, $\bar{\Psi}$, порождает операцию $1+\zeta+\Psi+\Psi$. Тем самым доказано, что клон $L_{\zeta+\delta}$ является максимальным в $L'_{\zeta,\Psi}$.

Множество $L'_{\zeta,\Psi} \setminus L_{\zeta,\Psi}$ содержит операции вида

$$1+\sum_{i=1}^{n_1}\gamma(x_{ji})+\sum_{i=1}^{n_2}\zeta(x_{\ell i})+\sum_{i=1}^{n_3}\Psi(x_{mi}), \qquad /5/$$

где $n_2+n_3 = 2k+1$. Отождествляя переменные в однотипных слагаемых, а также в парах γ, ζ; γ, Ψ, мы получим одну из операций $1+\zeta=\bar{\zeta}$, $1+\Psi=\bar{\Psi}$. $L_{\zeta,\Psi}$ содержит операцию $\zeta+\Psi+\Psi$. Так как $(\zeta+\Psi+\Psi)*\bar{\zeta}=\bar{\Psi}*(\zeta+\Psi+\Psi)=1+\zeta+\Psi+\Psi$, то $L_{\zeta,\Psi}$ максимален в $L'_{\zeta,\Psi}$.

Множество $L'_{\zeta,\Psi} \setminus L'_{\zeta}$ содержит операции вида /3/, в которых одно из слагаемых, есть либо Ψ, либо $\bar{\Psi}$.

Возьмем одну из таких операций, и пусть i-тое слагаемое есть либо Ψ, либо $\bar{\Psi}$. Отождествляя все переменные, кроме x_i, получим одну из следующих операций:

$$\Psi,\ \bar{\Psi},\ \gamma+\Psi,\ 1+\gamma+\Psi.$$

Так как $\Delta^2((1+\gamma+\Psi)*(1+\gamma+\Psi))=\bar{\Psi}$, $\bar{\Psi}*\bar{\Psi}=\Delta^2((\gamma+\Psi)*(\gamma+\Psi))=\Psi$, то каждая из них порождает Ψ. Эта операция вместе с принадлежащей L'_{ζ} операцией $1+\zeta+\zeta+\zeta$ порождает $1+\zeta+\Psi+\Psi$, и тем самым клон $L'_{\zeta,\Psi}$.

Аналогично доказывается максимальность в $L'_{\zeta,\Psi}$ двойственного к

L'_ζ клона L'_ψ .

Рассмотрим теперь произвольную систему операций клона $L'_{\zeta,\psi}$, не содержащуюся целиком в клонах L'_ζ , L'_ψ , $L_{\zeta,\psi}$, $L_{\zeta+\delta}$. Эта система операций содержит

(i) сумму вида /3/, в которой по крайней мере три слагаемых отличны от γ;

(ii) сумму вида /5/;

(iii) сумму вида /3/, в которой одно из слагаемых есть либо ψ, либо $\bar{\psi}$;

(iv) сумму вида /3/, в которой одно из слагаемых есть либо ζ, либо $\bar{\zeta}$.

Выше было показано, что из операции со свойством (ii), отождествлениями можно получить либо $\bar{\zeta}$, либо $\bar{\psi}$, из суммы (iii) - ψ, из суммы (iv) - ζ, из суммы (i) - одну из операций /4/. Любая из операций системы /4/ вместе с $\bar{\zeta}$, $\bar{\psi}$ порождает $1+\zeta+\psi+\psi$. Таким образом клон $L'_{\zeta,\psi}$ не имеет максимальных подклонов, отличных от L'_ζ , L'_ψ , $L_{\zeta,\psi}$, $L_{\zeta+\delta}$.

Клон H_ζ порождается операциями $\{\zeta+\gamma,\ c_o,\ e_2^2\}$. Так как $\zeta+\gamma$ порождает клон $L_{\zeta+\gamma}$, содержащий суммы вида $\zeta+\gamma+\ldots+\gamma$, $\psi+\gamma+\ldots+\gamma$, то очевидно, что H_ζ содержит любые суммы $\gamma+\ldots+\gamma$, ζ и ψ, т.е. все операции из $J_{\infty o}^{\zeta\psi}$ /рис. 3/. Множество $H_\zeta \backslash J_{\infty o}^{\zeta\psi}$ содержит суммы $\zeta+\gamma+\ldots+\gamma$, $\psi+\gamma+\ldots+\gamma$ с числом слагаемых больше единицы, поэтому $J_{\infty o}^{\zeta\psi}$ - максимальный подклон клона H_ζ. Множество $H_\zeta \backslash L_{\zeta+\gamma}$ содержит все операции из $J_{\infty o}^{\zeta\psi}$, не содержащиеся в клоне, порождаемом базисом ζ, ψ, e_2^2, т.е. операции $g_n=\gamma+\ldots+\gamma$. Каждая из этих операций при отождествлении всех переменных дает c_o, поэтому клон $L_{\zeta+\gamma}$ максимален в H_ζ. Любая система операций клона H_ζ, не содержащаяся целиком в клонах $J_{\infty o}^{\zeta\psi}$, $L_{\zeta+\gamma}$, содержит f_n и либо $\zeta+g_m$, либо $\psi+g_m$, $m\geq 1$, и потому порождает H_ζ. Так как клон H_ζ конечно порожденный, то других максимальных подклонов клон H_ζ не имеет. Двойственным к H_ζ является клон H_ψ.

Клон $S_{\zeta,\psi}$ порождается базисом $\{e_2^2, \zeta+\psi\}$. Отождествляя $y(\zeta+\psi)*(\zeta+\psi)=$

$=\zeta+\Psi+\Psi$ две последние переменные, получим ζ, поэтому в $S_{\zeta,\Psi}$ содержатся операции $\zeta+\zeta$ и $\zeta+\Psi+\zeta$. Из последней получаем Ψ, и из $\zeta+\zeta$ и Ψ получаем $\Psi+\Psi$. Таким образом $S_{\zeta,\Psi}$ содержит S_ζ и S_Ψ. Легко заметить, что $S_{\zeta,\Psi}$ содержит суммы вида

$$\sum_{i=1}^{n_1}\gamma(x_{\ell i})+\sum_{i=1}^{n_2}\zeta(x_{pi})+\sum_{i=1}^{n_3}\Psi(x_{qi}) \quad . \qquad /6/$$

Отсюда заключаем, что $S_{\zeta,\Psi}$ содержит также $L_{\zeta,\Psi}$ и H_ζ.

Докажем, что клоны S_ζ, S_Ψ, $L_{\zeta,\Psi}$ и H_ζ являются максимальными в $S_{\zeta,\Psi}$. Множество $S_{\zeta,\Psi}\backslash S_\zeta$ содержит суммы вида /6/, в которых либо $n_1>0$, либо $n_3>0$. Пусть $f\in S_{\zeta,\Psi}\backslash S_\zeta$.

а/ $n_1>0$. Фиксируем одно из слагаемых вида γ, и отождествляем переменные у остальных. Получим одну из операций γ, $\zeta+\gamma$, $\Psi+\gamma$. Подставляя $c_o\in S_\zeta$ в две последние операции, также получим γ. Так как $\zeta+\zeta+\zeta\in S_\zeta, ((\zeta+\zeta)*\gamma)=\Psi+\zeta$, то f вместе с S_ζ порождает $S_{\zeta,\Psi}$.

б/ $n_3>0$. Фиксируем одно из слагаемых вида Ψ, и отождествляем переменные у остальных. Получим одну из операций Ψ, $\gamma+\Psi$, $\Psi+\Psi$, $\zeta+\Psi$. Подставляя c_o в три последние операции и отождествляя переменные, получим во всех случаях Ψ. Операции Ψ и $\zeta+\zeta$ порождают $S_{\zeta,\Psi}$.

Множество $S_{\zeta,\Psi}\backslash L_{\zeta,\Psi}$ содержит суммы вида /6/, в которых $n_2+n_3=2k$. Пусть $f\in S_{\zeta,\Psi}\backslash L_{\zeta,\Psi}$.

а/ $k=0$. Отождествляя у f все переменные, получим c_o. Операция $(\zeta+\zeta+\Psi)*c_o=\zeta+\Psi$ порождает $S_{\zeta,\Psi}$.

б/ $k\geq 1$. Фиксируем два слагаемых из множества $\{\zeta,\Psi\}$ отождествляем остальные переменные. В результате получим сумму

$$\sum_{i=1}^{n_1}\gamma(x_{\ell i})+\sum_{i=1}^{n_2}\zeta(x_{pi})+\sum_{i=1}^{n_3}\Psi(x_{qi})+\mu_1(x_1)+\mu_2(x_2), \qquad /7/$$

в которой все $x_{\ell i}$, x_{pi}, x_{qi} равны x_3 $\{\mu_1, \mu_2\}\subseteq\{\zeta, \Psi\}$. Возможны следующие варианты:

i). n_2 и n_3 четны. Тогда сумма /7/ дает одну из операций $\zeta+\zeta$, $\zeta+\Psi$, $\gamma+\zeta+\zeta$, $\gamma+\zeta+\Psi$, $\gamma+\Psi+\Psi$.

Операция $\zeta+\Psi$ порождает $S_{\zeta\Psi}$. В остальных случаях отождествление переменных дает либо γ, либо c_o. Так как $\gamma*\gamma=c_o$, то во всех случаях имеем c_o. Операции c_o, $\zeta+\zeta+\Psi$ порождают $S_{\zeta,\Psi}$.

ii). n_2 и n_3 нечетны. Сумма /7/ принимает либо вид

$$\mu_1(x_1)+\mu_2(x_2)+\zeta(x_3)+\Psi(x_3), \text{ либо вид}$$

$$\mu_1(x_1)+\mu_2(x_2)+\zeta(x_3)+\Psi(x_3)+\gamma(x_3).$$

Так как $\zeta(x)+\Psi(x)=\gamma(x)$, то получаем варианты $\zeta+\zeta+\gamma$, $\zeta+\Psi+\gamma$, $\Psi+\Psi+\gamma$, $\zeta+\zeta$, $\Psi+\Psi$, $\zeta+\Psi$, уже рассмотренные ранее.

Множество $S_{\zeta,\Psi}\backslash H_\zeta$ содержит суммы вида /6/, в которых либо $n_2\geq 2$, либо $n_3\geq 2$. Пусть $n_2\geq 2$. Фиксируем два слагаемых вида ζ и отождествляем переменные у остальных. Получим одну из операций $\zeta+\zeta$, $\zeta+\zeta+\zeta$, $\zeta+\zeta+\Psi$, $\zeta+\zeta+\gamma$.

Так как $c_o\in H_\zeta$, то во всех случаях порождается $\zeta+\zeta$. Так как $\zeta+\zeta$ порождает S_ζ, то мы приходим к рассмотренному ранее случаю. Случай $n_3\geq 2$ аналогичен рассмотренному.

Каждая система C операции, не содержащихся целивоа в S_ζ, S_Ψ, $L_{\zeta\Psi}$ и H_ζ содержит

(i) сумму вида /6/, в которой $n_3\geq 1$;
(ii) сумму вида /6/, в которой $n_2\geq 1$;
(iii) сумму вида /6/, в которой $n_2+n_3=2k$, $k\geq 0$;
(iv) сумму вида /6/, в которой $n_2+n_3\geq 2$.

Существет два вида сумм /6/, удовлетворяющих (iii):

а/ $n_2+n_3=0$, тогда это либо $\gamma+\ldots+\gamma$, либо c_o. В любом случае порождается c_o.

б/ $n_2+n_3\neq 0$. Фиксируем любые два слагаемых из множества $\{\zeta,\Psi\}$, и отождествляем переменные у остальных. В результате получим одну из операций

$$\zeta+\Psi,\ \zeta+\zeta,\ \Psi+\Psi,\ \zeta+\zeta+\gamma,\ \zeta+\Psi+\gamma,\ \Psi+\Psi+\gamma.$$

Последние три при отождествлении переменных x_2 и x_3 дают $\zeta+\Psi$ и $\Psi+\zeta$ соответственно. Операция $\zeta+\Psi$ порождает $S_{\zeta,\Psi}$, операции $\zeta+\zeta$ и $\Psi+\Psi$ порождают S_ζ и S_Ψ соответственно. Так как система содержит также операцию (i) и (ii), не принадлежащие S_ζ и S_Ψ, то в двух последних случаях также порождается $S_{\zeta,\Psi}$.

Таким образом можно считать, что C порождает c_o. Зафиксировав в сумме (iv) два слагаемых из множества $\{\zeta,\Psi\}$ и поставив в остальные c_o, придем к рассмотренному выше случаю б/. Видим, что максимальными подклонами клона $S_{\zeta\Psi}$ являются лишь S_ζ, S_Ψ, $L_{\zeta,\Psi}$ и H_ζ. Клоном, двойственным к $S_{\zeta,\Psi}$ является $S'_{\zeta,\Psi}$, порождаемый операцией $1+\zeta+\Psi$.

Клон Z состоит из операций, принадлежащих E , и операций, представимых в виде $u_1(x_1)+\ldots+u_n(x_n)\ (n\geq 1)$, где операции u_i однолистные. Легко заметить, что в качестве $u_1,\ldots,u_n$ можно брать операции, принадлежащие клону Z, поэтому каждая операция клона Z представима в виде суммы

$$\sum_{i=1}^{n_o} c_o(x_{ji})+\sum_{i=1}^{n_1} c_1(x_{\ell i})+\sum_{i=1}^{n_2}\gamma(x_{mi})+\sum_{i=1}^{n_3}\delta(x_{ui})+\sum_{i=1}^{n_4}\zeta(x_{pi})+\sum_{i=1}^{n5}\Psi(x_{qi})+$$

$$+\sum_{i=1}^{n_6}\bar{\zeta}(x_{\tau i})+\sum_{i=1}^{n_7}\bar{\Psi}(x_{si}).$$

Эта сумма упрощается, если учесть следующие свойства слагаемых. Несущественные переменные можно опустить. Так как сложение ведется по mod2, то в сумму войдет не более одного слагае-

мого c_1. Так как $\delta=1+\gamma$, $\bar{\zeta}=1+\zeta$, $\bar{\Psi}=1+\Psi$, то, если функция отлична от c_1, слагаемое c_1 можно не писать. Из этих же равенств заключаем, что в сумму достаточно включать лишь одно слагаемое из $\bar{\zeta}$, $\bar{\Psi}$, δ. Таким образом любая отличная от 0 и 1 операция из Z с точностью до несущественных переменных представима в виде суммы

$$\sum_{i=1}^{n_1}\gamma(x_{mi})+\sum_{i=1}^{n_2}\zeta(x_{pi})+\sum_{i=1}^{n_3}\Psi(x_{qi})+\alpha_1\cdot\delta(x_{u1})+\alpha_2\bar{\zeta}(x_{u2})+\alpha_3\bar{\Psi}(x_{u3}),\ /8/$$

в которой $\{\alpha_1,\alpha_2,\alpha_3\}=\{0,1\}$, $\alpha_1+\alpha_2+\alpha_3\leq 1$.

Клон $H_{\zeta\Psi}$

Докажем теперь, что клоны $S_{\zeta\Psi}$, $S'_{\zeta\Psi}$, U_ζ, U_Ψ, $H_{\zeta\Psi}$, $L'_{\zeta\Psi}$ являются максимальными в Z. Для доказательства нам нужна некоторая система порождающих клона Z. В качестве такой системы можно взять e_2^2, $\zeta+\Psi$, $\bar{\zeta}$. Действительно, $\zeta+\Psi$ порождает все суммы вида /6/. Подставляя нужное число раз $\bar{\zeta}$, получим сумму /8/.

Множество $Z\setminus S_{\zeta\Psi}$ содержит суммы /8/, в которых $\alpha_1+\alpha_2+\alpha_3=1$, и c_1. Пусть $f\in Z\setminus S_{\zeta\Psi}$. Так как e_2^2 и $\zeta+\Psi$ уже содержатся в $S_{\zeta\Psi}$, нам нужно получить только $\bar{\zeta}$. Если $f\neq c_1$, то f есть сумма /8/, в которой для некоторого $i\in\{1,2,3\}$, $\alpha_i\neq 0$. Подставив вместо каждой отличной от u_i переменной $c_o\in S_{\zeta\Psi}$, получим одну из операций δ, $\bar{\zeta}$, $\bar{\Psi}$. Подставляя в любую из них c_o, получим c_1.

Таким образом можно предполагать, что $f=c_1$. Композицией операций $\zeta+\zeta\in S_{\zeta\Psi}$ и c_1, получаем $1+\zeta=\bar{\zeta}$.

Множество $Z\setminus U_\zeta$ содержит суммы /8/, в которых либо $n_1\neq 0$, либо $n_3\neq 0$, либо $\alpha_1\neq 0$, либо $\alpha_3\neq 0$.

a/ $n_1\neq 0$. Фиксируем одно слагаемое γ и в остальные подставляем $c_o\in U_\zeta$. Получим либо γ, либо $1+\gamma=\delta$. Далее, $\Delta((\zeta+\zeta+\zeta)*\gamma)=\Psi+\zeta=\Delta((1+\zeta+\zeta+\zeta)*\delta)$. Операции e_2^2 и $\bar{\zeta}=1+\zeta$ уже

принадлежит U_ζ.

б/ $n_3 \neq 0$. Фиксируя слагаемое Ψ и подставляя в остальные c_o, получим либо Ψ, либо $\Psi+1=\bar{\Psi}$. Далее, $\bar{\Psi}*\bar{\Psi}=\Psi$, $\zeta+\zeta \in U_\zeta$, $(\zeta+\zeta)*\Psi=\Psi+\zeta$.

в/ $\alpha_3 \neq 0$. Подставляя вместо каждой отличной от x_{u3} переменной c_o, получим $\bar{\Psi}$. Попадаем в условие пункта б/.

Множество $Z \setminus F_\infty^{\zeta\Psi}$ содержит суммы /8/, в которых $n_2+n_3+\alpha_2+\alpha_3 \geq 2$. Пусть $f \in Z \setminus F_\infty^{\zeta\Psi}$. Фиксируя в f два слагаемых из множества $\{\zeta, \Psi, \bar{\zeta}, \Psi\}$ и подставляя в остальные $c_o \in F_\infty^{\zeta\Psi}$, получим одну из операций $\zeta+\zeta$, $\zeta+\zeta+1$, $\zeta+\Psi+1$, $\zeta+\Psi$. Операции ζ, Ψ, $\bar{\zeta}$, $\bar{\Psi}$ принадлежат $F_\infty^{\zeta\Psi}$, $\Psi+\zeta=(\zeta+\zeta)*\Psi=(\zeta+\zeta+1)*\bar{\Psi}$, $\zeta+\Psi=(\zeta+\Psi+1)*\bar{\zeta}$.

Множество $Z \setminus L'_{\zeta\Psi}$ содержит суммы /8/, в которых $n_2+n_3+\alpha_2+\alpha_3 = 2k$. Пусть $f \in Z \setminus L'_{\zeta\Psi}$. Так как $\bar{\zeta} \in L'_{\zeta\Psi}$, нужно получить только $\zeta+\Psi$.

а/ $k=0$. Отождествив все переменные, получим $\mu \in \{\gamma, \delta, c_o, c_1\}$; γ и δ дают c_o и c_1; $(\zeta+\zeta+\Psi)*c_o=(1+\zeta+\zeta+\Psi)*c_1=\zeta+\Psi$, что и требовалось.

б/ $k \neq 0$. Фиксируя два слагаемых из множества $\{\zeta, \Psi, \bar{\zeta}, \bar{\Psi}\}$, и отождествляя переменные у остальных, получим одну из операций $\zeta+\zeta$, $\zeta+\Psi$, $\zeta+\zeta+1$, $\zeta+\Psi+1$, $\zeta+\Psi+1$, $\gamma+\zeta+\zeta$, $\gamma+\zeta+\Psi$, $\delta+\zeta+\zeta$, $\delta+\zeta+\Psi$ /см. доказательство максимальности клона $L_{\zeta\Psi}$ в $S_{\zeta\Psi}$/. Клон $L'_{\zeta\Psi}$ содержит ζ, Ψ, $\bar{\zeta}$, $\bar{\Psi}$, $\Psi+\zeta=(\zeta+\zeta)*\Psi=(\zeta+\zeta+1)*\bar{\zeta}=\Delta(\gamma+\zeta+\zeta)=$ $=(\Delta(\delta+\zeta+\zeta))*\bar{\Psi}$, $\zeta+\Psi=(\zeta+\Psi+1)*\bar{\zeta}=(\Delta(\gamma+\zeta+\Psi))*\zeta=(\Delta(\delta+\zeta+\Psi))*\bar{\zeta}$.

Клоны $S'_{\zeta\Psi}$ и U_Ψ двойственны клонам $S_{\zeta\Psi}$ и U_ζ и поэтому также максимальны в Z. Осталось доказать, что клон Z не имеет максимальных подклонов, отличных от уже рассмотренных. Пусть - система операций клона Z, не содержащаяся целиком в клонах $S_{\zeta\Psi}$, $S'_{\zeta\Psi}$, $S'_{\zeta\Psi}$, U_ζ, U_Ψ, $F_\infty^{\zeta\Psi}$, $L'_{\zeta\Psi}$. Система C содержит

(i) или сумму вида /8/, в которой $\alpha_1+\alpha_2+\alpha_3=1$, или c_1;
(ii) сумму вида /8/, в которой $n_1 \neq 0 \vee n_3 \neq 0 \vee \alpha_1 \neq 0 \vee \alpha_3 \neq 0$;

(iii) сумму вида /8/, в которой $n_2+n_3+\alpha_2+\alpha_3\geq 2$;

(iv) сумму вида /8/, в которой $n_2+n_3+\alpha_2+\alpha_3=2k$;

(v) сумму вида /8/, в которой или $n_2+n_3=2k+1$, $\alpha_1=1$; или $n_2+n_3=2k$, $\alpha_1=0$, $\alpha_2+\alpha_3=1$, или $n_2+n_3=2k$, $k\neq 0$, $\alpha_1+\alpha_2+\alpha_3=0$;

(vi) сумму вида /8/, в которой $n_1\neq 0 \vee n_2\neq 0 \vee \alpha_1\neq 0 \vee \alpha_2\neq 0$.

Сумма (i) дает при отождествлении всех переменных одну из операций $\bar{\zeta}$, $\bar{\Psi}$, δ, $\bar{\zeta}+\gamma$, $\bar{\Psi}+\gamma$, c_1. Так как $\Delta(\bar{\zeta}+\gamma)=\bar{\Psi}$, $\Delta(\bar{\Psi}+\gamma)=\bar{\zeta}$, $\delta*\delta=c_1$, то получаем либо $\bar{\zeta}$, либо $\bar{\Psi}$, либо c_1.

а/ Пусть получили c_1. Если в сумме (v) $n_2+n_3=2k+1$, $\alpha_1=1$, или $n_2+n_3=2k$, $\alpha_2+\alpha_3=1$, то, фиксируя три отличных от δ, γ слагаемых и подставляя в остальные c_1, получим одну из операций

$$\zeta+\zeta+\zeta+1,\ \zeta+\zeta+\Psi+1,\ \zeta+\Psi+\Psi+1,\ \Psi+\Psi+\Psi+1,\ \zeta+1,\ \Psi+1.$$

Подставляя c_1, из каждой можно получить либо $\bar{\zeta}$, либо $\bar{\Psi}$. Если в сумме (v) $n_2+n_3=2k$, $k\neq 0$, $\alpha_1=\alpha_2=\alpha_3=0$, то, фиксируя два отличных от γ и δ слагаемых, и подставляя в остальные c_1, получим одну из операций $\zeta+\zeta$, $\zeta+\Psi$, $\Psi+\Psi$, каждая из которых в свою очередь дает либо $\bar{\zeta}$, либо $\bar{\Psi}$. Имея $\bar{\zeta}$ или $\bar{\Psi}$ и c_1, получаем c_o.

Зафиксируем в сумме (iii) два отличных от γ и δ слагаемых и подставим в остальные c_o, получим одну из операций

$$\zeta+\zeta,\ \zeta+\Psi,\ \Psi+\Psi,\ \zeta+\zeta+1,\ \zeta+\Psi+1,\ \Psi+\Psi+1.$$

Последние три операции при подстановке в $\bar{\zeta}$ или $\bar{\Psi}$ дают три первые. Так как $\zeta+\Psi$ вместе с любой из операций $\bar{\zeta}$, $\bar{\Psi}$ порождает Z, то будем предполагать, что мы получили $\zeta+\zeta$ и $\bar{\zeta}$.

Если в сумме (vi) $n_1\neq 0$, то фиксируя слагаемое γ и подставляя в остальные c_o, получим либо γ, либо δ. в первом случае $\Delta(((\zeta+\zeta)*(\zeta+\zeta))*\gamma)=\Psi+\zeta$, во втором $(((\zeta+\zeta)*(\zeta+\zeta))*\bar{\zeta})*\delta=\gamma+\zeta+\zeta$, $\Delta(\gamma+\zeta+\zeta)=\Psi+\zeta$.

б/ Пусть из (i) мы получили $\bar{\zeta}$. Если в (iv) k=0, то, отождествив все переменные, получим $\mu\in\in\{\gamma,\delta,c_o,c_1\}$. Так как $\gamma*\gamma=c_o$, $\delta*\delta=c_1$, то всегда имеем либо c_o, либо c_1. Так как $\bar{\zeta}*c_o=c_1$, то попадаем в условия пункта а/.

Если в (iv) k≠0, то, фиксируя два отличных от γ и δ слагаемых, и отождествляя переменные у остальных, получим одну из операций

$$\zeta+\zeta,\ \zeta+\Psi,\ \zeta+\zeta+1,\ \zeta+\Psi+1,\ \gamma+\zeta+\zeta,\ \gamma+\zeta+\Psi,\ \delta+\zeta+\zeta,\ \delta+\zeta+\Psi.$$

Далее,

$$\bar{\zeta}*(\zeta+\zeta+1)=\zeta+\zeta,\ \bar{\zeta}*(\zeta+\Psi+1)=\zeta+\Psi,\ \Delta(\gamma+\zeta+\zeta)=\Psi+\zeta,\ \Delta(\gamma+\zeta+\Psi)=\Psi+\Psi,$$

$$(\delta+\zeta+\zeta)*\bar{\zeta}=1+\zeta+\zeta,\ (\delta+\zeta+\Psi)*\bar{\zeta}=1+\zeta+\Psi.$$

Операции $\zeta+\Psi$ и $\Psi+\Psi$ вместе с $\bar{\zeta}$ дают базис Z, поэтому предполагаем, что мы получили $\zeta+\zeta$. Однако $\Delta(\zeta+\zeta)=c_o$, $\bar{\zeta}*c_o=c_1$, и мы снова попадаем в условия пункта а/. Тем самы доказано, что Z не имеет максимальных подклонов, отличных от шести ранее названных.

Рис. 7 показывает взаимное расположение клонов, не показанных на рис 3 и 5.

В целом проведенные рассуждения образуют доказательство следующего утверждения.

Теорема 1. Подклоны клона Z образуют решетку, изображенную на рис. 8.

3. Клон L и его подклоны.

Теперь рассмотрим клон L всех линейных операций на множестве A, т.е. операций, представимых в виде

$$a_o+a_1x_1+\ldots+a_nx_n, \qquad /1/$$

где + и . - сложение и умножение по mod.3.

Из определения следует, что одноместные операции из L представимы в виде $a_o+a_1x_1$ и поэтому либо принимают все три зна-

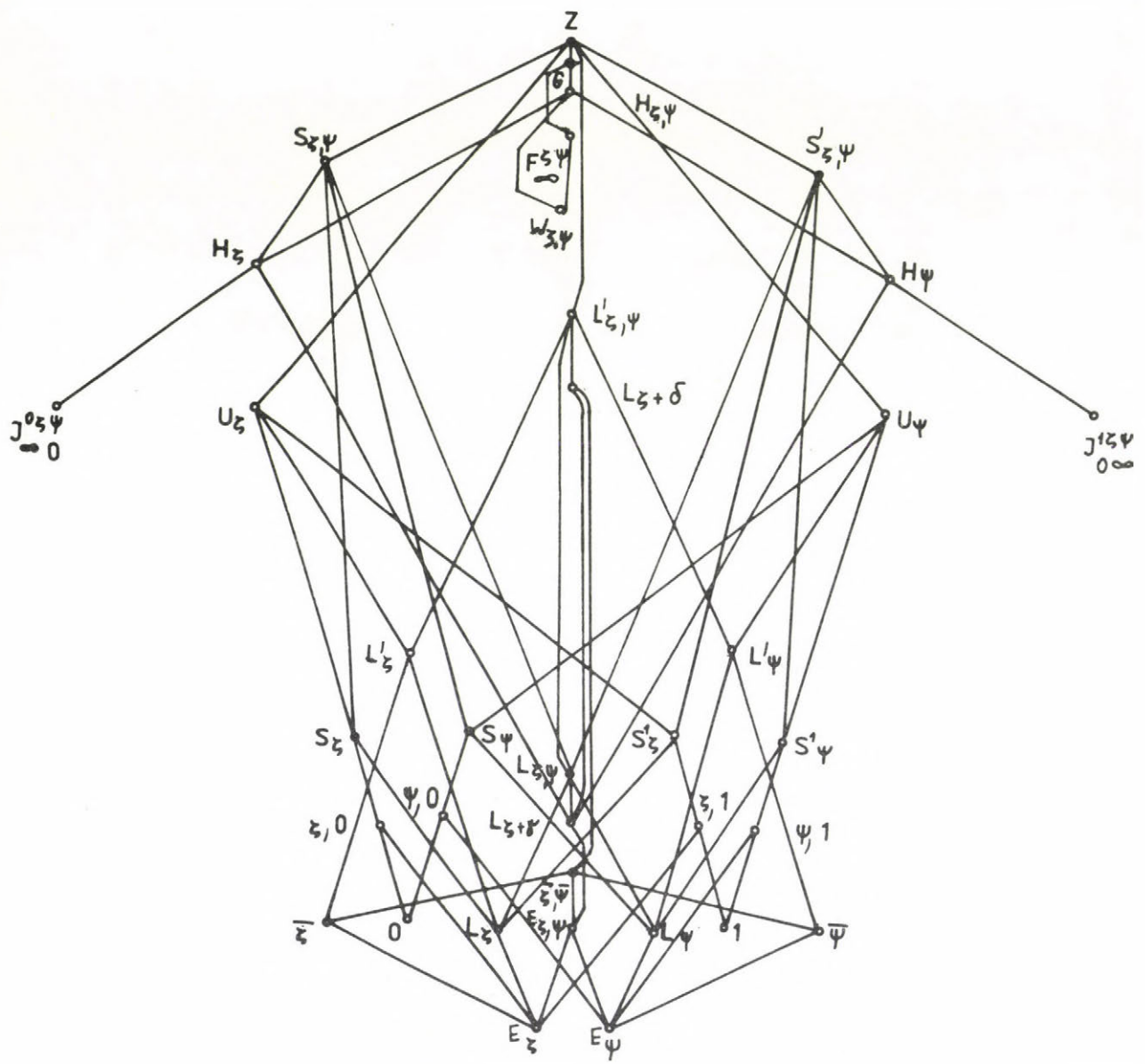

Z
G
$H_{\zeta,\psi}$
$S_{\zeta,\psi}$
$F^{\zeta\psi}_{\infty}$
$S'_{\zeta,\psi}$
$W_{\zeta,\psi}$
H_ζ
H_ψ
$L'_{\zeta,\psi}$
$L_{\zeta+\delta}$
$J^{0\zeta\psi}_{0}$
U_ζ
U_ψ
$J^{1\zeta\psi}_{0\infty}$
L'_ζ
L'_ψ
S_ζ
S_ψ
S'_ζ
S'_ψ
$L_{\zeta,\psi}$
$\psi, 0$
$\zeta, 1$
$\zeta, 0$
$L_{\zeta+\delta}$
$\psi, 1$
$\bar{\zeta}$
0
L_ζ
$E_{\zeta,\psi}$
L_ψ
1
$\bar{\psi}$
E_ζ
E_ψ

Рисунок 7.

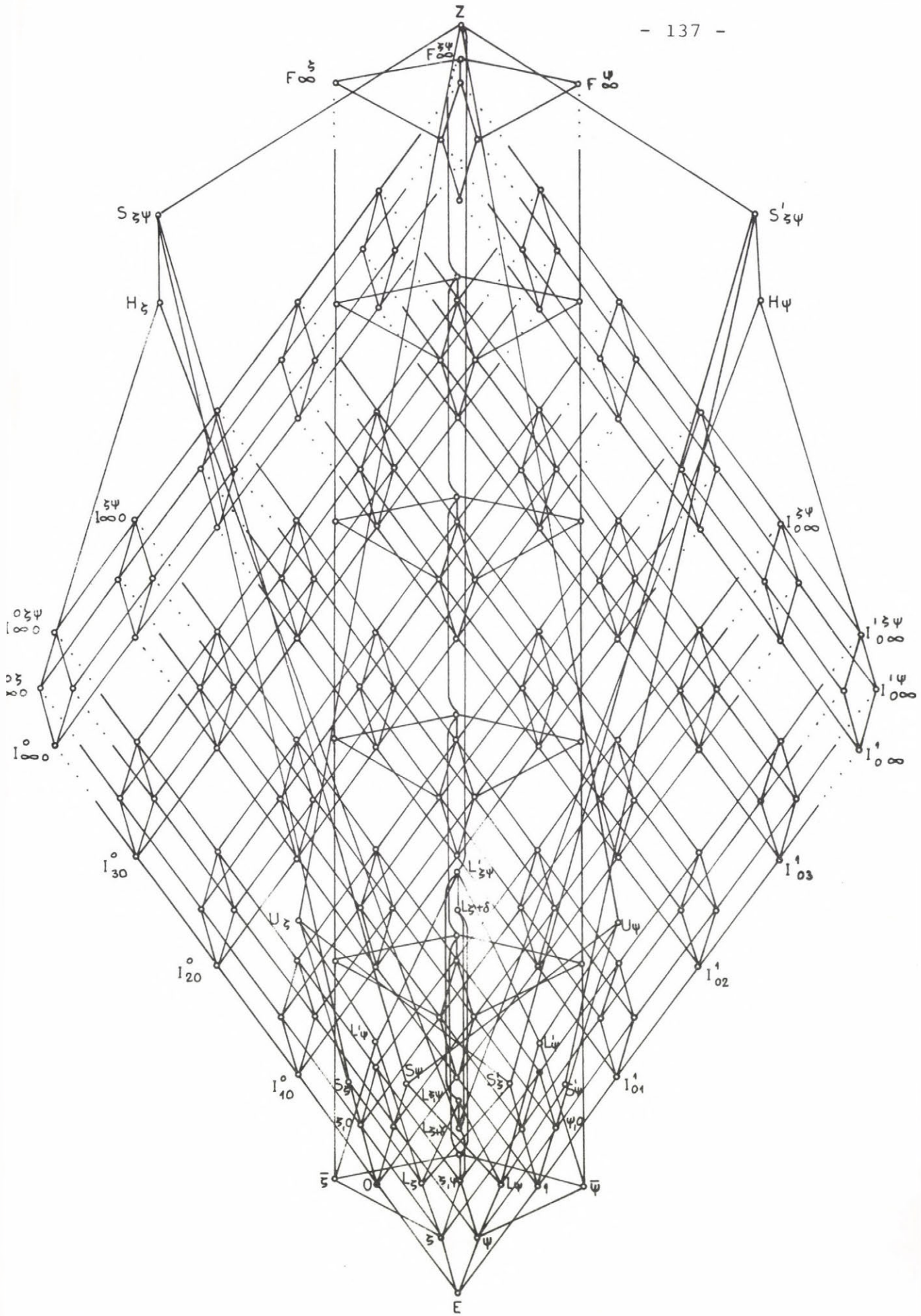

Z
F^{ξ}_{∞}
$F^{\xi\psi}_{\infty\infty}$
F^{ψ}_{∞}
$S_{\xi\psi}$
$S'_{\xi\psi}$
H_{ξ}
H_{ψ}
$I^{\xi\psi}_{\infty\infty 0}$
$I^{\xi\psi}_{0\infty\infty}$
$I^{\xi}_{\infty 0}$
$I^{\psi}_{\infty 0}$
$I^{\xi\psi}_{\infty 0\infty}$
$I^{\xi}_{\infty\infty}$
$I^{\psi}_{\infty\infty}$
$I^{\xi\psi}_{0\infty\infty}$
$I^{\psi}_{0\infty}$
$I^{\xi}_{0\infty}$
$I^{0}_{\infty 0}$
I^{0}_{30}
I^{1}_{03}
$L'_{\xi\psi}$
$L_{\xi+\delta}$
U_{ξ}
U_{ψ}
I^{0}_{20}
I^{1}_{02}
L'_{ψ}
L'_{ξ}
S_{ψ}
S_{ξ}
I^{0}_{10}
I^{1}_{01}
$L_{\xi\psi}$
$\xi,0$
$\psi,0$
$\bar{\xi}$
0
L_{ξ}
ξ,ψ
L_{ψ}
1
$\bar{\psi}$
ξ
ψ
E

Рисунок 8.

чения 0, 1, 2, либо лишь одно значение. Существует 9 таких функций /таблица 2/:

x	c_o	c_1	c_2	$\mathbb{I}_1$	$\mathbb{I}_2$	ξ_o	ξ_1	ξ_2	e_1^1
0	0	1	2	2	1	0	2	1	0
1	0	1	2	0	2	2	1	0	1
2	0	1	2	1	0	1	0	2	2

Таблица 2.

Результаты суперпозиции $f*g$, f, $g \in \{\mathbb{I}_1, \mathbb{I}_2, \xi_o, \xi_1, \xi_2\}$ показаны в таблице 3.

g \ f	$\mathbb{I}_1$	$\mathbb{I}_2$	ξ_o	ξ_1	ξ_2
$\mathbb{I}_1$	$\mathbb{I}_2$	e_1^1	ξ_2	ξ_o	ξ_1
$\mathbb{I}_2$	e_1^1	$\mathbb{I}_1$	ξ_1	ξ_2	ξ_o
ξ_o	ξ_o	ξ_1	e_1^1	$\mathbb{I}_2$	$\mathbb{I}_1$
ξ_1	ξ_2	ξ_2	$\mathbb{I}_1$	e_1^1	$\mathbb{I}_2$
ξ_2	ξ_2	ξ_o	$\mathbb{I}_2$	$\mathbb{I}_1$	e_1^1

Таблица 3.

Видим, что клон, содержащий одну из операций $\mathbb{I}_1$, $\mathbb{I}_2$, содержит и другую, и что операция $\mathbb{I}_1$ вместе с одной из операций ξ_o, ξ_1, ξ_2 порождает остальные две. Всего клон $L^{(1)}$ одноместных операций из L имеет 20 собственных подклонов, образующих решетку, изображенную на рис. 9.

Переходим к описанию подклонов клона L, содрежащих существен-

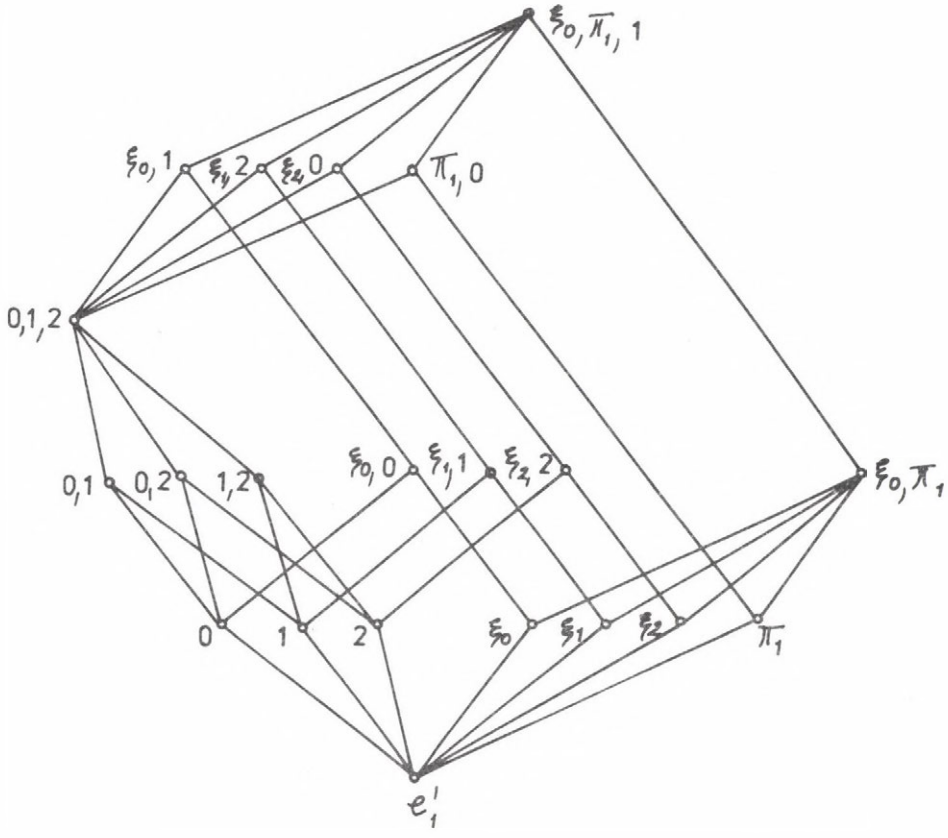


Рисунок 9.
Полугруппа $L^{(1)}$

но многоместные операции. Будем говорить, что операция $f(x_1,\ldots,x_n)$ из L сохраняет константу c_i, если $f(c_i,\ldots,c_i)=c_i$. Обозначим через L_i клон, образованный всеми операциями из L, сохраняющими константу c_i. Через L_s обозначим клон, образованный операциями f из L, обладающими следующим свойством

$$\forall_i\in\{1,2\}\ f(\Pi_i(x_1),\ldots,\Pi_i(x_n))=\Pi_i(f(x_1,\ldots,x_n)).$$

Лемма 1. Клон L порождается операциями $u(x_1,x_2)=x_1+x_2$, $c_1(x)$.

Доказательство. Докажем, что из указанных операций можно получить сумму /1/ . Очевидно, что суперпозициями из u можно получить сумму $x_1+\ldots+x_m$ любой длины. Пусть $m=\sum_{i=0}^{n} a_i$. Подставим c_1 вместо переменных $x_1,\ldots,x_{a_o}$ и отождествим в получившейся сумме переменные $x_1,\ldots,x_{a_o}$; $x_{a_o+1},\ldots,x_{a_1}$;...; $x_{a_{n-1}+1},\ldots,x_{a_n}$. В результате получим выражение /1/.

Лемма 2. Клоны $L^{(1)}$, L_o, L_1, L_2 и L_s и только они являются максимальными подклонами клона L.

Доказательство. Сначала докажем, что каждый из этих подклонов максимален в L. Пусть $f\in L\setminus L^{(1)}$. Это означает, что f - существенно многоместная операция, имеющая вид /1/. Ввиду леммы 1 нам достаточно из f получить операцию u.

Предположим для простоты, что в /1/ $a_1\neq 0$ и $a_2\neq 0$. Подставляя вместо переменных $x_3,\ldots,x_n$ константу c_o, получим операцию $a_o+a_1x_1+a_2x_2$. Подставив вместо x_i операцию $x_i+((3+1)-a_i)$, получим $a_o+x_1+x_2$. Наконец, подставив вместо x_1 операцию $x_1+(3-a_o)$, получим x_1+x_2.

Перейдем к клону L_i, $i\in\{0,1,2\}$. Множеству $L\setminus L_i$ принадлежат такие операции f, для которых $f(c_i,\ldots,c_i)=c_j$, $j\neq i$. Так как $c_i\in L_i$, то c_j принадлежит клону K, порождаемому операций f и операциями из L_i. Рис.9(10) показывает, что тогда клон K содер-

жит все константы. Операция x_1+x_2 принадлежит $L_i \subseteq K$.

Рассмотрим теперь клон L_s. Для любой операции $f \in L \setminus L_s$ найдутся такое $i \in \{1,2\}$ и такие $b_1,\ldots,b_n$ из A, что

$$f(\Pi_i(b_1),\ldots,\Pi_i(b_n)) \neq (\Pi_i * f)(b_1,\ldots,b_n).$$

Пусть $\Pi_i(b_1)=c_1,\ldots,\Pi_i(b_n)=c_n$, $\Pi_i(0)=p$. Так как Π_i - перестановка, то $b_j \neq c_j$, $j=1,\ldots,n$, $0 \neq p$. Обозначим через γ_i такую операцию из множества $\{\Pi_1,\Pi_2,e_1^1\}$, что $\gamma_i(p)=b_i$, $\gamma_i(0)=c_i$. Операция $g(x)=f(\gamma_1(x),\ldots,\gamma_n(x_n))$ принадлежит клону, порождаемому одноместными операциями из L_s /которыми являются Π_1, Π_2, e_1^1/ и операцией f. В то же время $g(\Pi_i(0)) \neq \Pi_i(g(0))$, т.е. $g \in L \setminus L_s$. Последнее означает, что g является константой или одной из операций ξ_j. Имея константу c_j, с помощью операции Π_i получаем остальные константы. Если же $g=\xi_j$, то g и Π_i порождают операцию ξ_o. Клону L_s принадлежит операция $t(x_1,x_2)=2x_1+2x_2$. Легко проверить, что $t(x,\xi_o(x))=c_o(x)$.

Клону L_s принадлежит операция $(t*t)(x_1,x_2,x_3)=x_1+x_2+2x_3$. Подставив вместо x_3 константу c_o и отождествив x_2 и x_3, получим x_1+x_2, что и требовалось.

Теперь докажем, что клон L других максимальных подклонов не имеет. Пусть C - система операций, не содержащаяся целиком ни в одном из клонов $L^{(1)}$, L_o, L_1, L_2, L_s. Системе C принадлежат

(i) существенно многоместная операция;

(ii) для каждого $j \in \{0,1,2\}$ такая операция $h_j(x_1,\ldots,x_n)$, что $h_j(C_j(x),\ldots,C_j(x)) \neq C_j(x)$;

(iii) для некоторого $j \in \{1,2\}$ такая операция $g(x_1,\ldots,x_n)$, что $g(\Pi_j(x_1),\ldots,\Pi_j(x_n)) \neq (\Pi_j * g)(x_1,\ldots,x_n)$.

Из (ii) видно, что операция $h_j(x,\ldots,x)$ принимает более одного значения и отлична от e_1^1. Остаются две возможности: либо

для некоторого i операция $h_j(x...x)$ совпадает с ξ_i, либо она совпадает с одной из операций Π_1, Π_2.

Если $C_1=\{h_i(x,...,x)|i\in\{0,1,2\}\}$ не содержит операций Π_1, Π_2, то эти операции порождаются системой C_1.

Действительно, $|C_1|\geq 2$, а любая пара различных элементов из $\{\xi_o,\xi_1,\xi_2\}$ порождает операции Π_1, Π_2.

Мы убедились, что операции Π_1, Π_2 порождаются системой C. Повторяя рассуждения, проведенные при доказательстве максимальности клона L_s в L, приходим к выводу, что порождаются также либо все константы, либо все операции ξ_j.

а/ Пусть порождаются все операции ξ_j, и пусть операция f со свойством (i) имеет вид /1/, где каждое a_i, i=1,..., n, отлично от нуля. Из уравнения $a_o=ba_1$ находим $b\in\{0,1,2\}$. Подставив x_1+b вместо x_1 в f, получим $f_1(x_1,...,x_n)=$ $=a_1x_1+...+a_nx_n$. Отождествив переменные $x_3,...,x_n$, получим $f_2(x_1,x_2,x_3)=a_1x_1+a_2x_2+a'_3x_3$.

Одной из операций $a_1x_1+a_2x_2+a'_3x_2$, $a_1x_1+a_2x_2+2a'_3x_2$ обе переменные существенные. Обозначим ее через f_3, и пусть $f_3(x_1x_2)=a_1x_1+a'_2x_2$. Из уравнений $a_1c_1=1$, $a'_2c_2=1$ находим c_1 и c_2, подставляя в f_3 c_1x_1 и c_2x_2 получаем нужную операцию x_1+x_2.

Лемма 3. Клон L_o порождается операциями $c_o(x)$ и x_1+x_2 L_1 операциями $c_1(x)$ и $2x_1+2x_2+1$, L_2 операциями $c_2(x)$ и $2x_1+2x_2+2$, L_s операцией $2x_1+2x_2+1$.

Доказательство. Очевидно, сумма /1/ тогда и только тогда принадлежит L_o, когда $a_o=0$. Имея операцию x_1+x_2, суперпозициями получаем сумму $x_1+...+x_m$, где $m=\sum_{i=1}^{n} a_i$. Отождествляя переменные $x_1,...,x_{a_1}$; $x_{a_{i+1}},...,x_{a_2}$;...; $x_{a_{n-1+1}},...,x_{a_n}$ получаем

сумму /1/, в которой $a_o=0$.

Автоморфизмы клона L, порождаемые операциями $\xi_1(x)=2x+1$ и $\xi_2(x)=2x+2$, отображают L_o на L_1 и L_2 соответственно. Базис $c_o(x)$, $\xi_o(x_1x_2)=x_1+x_2$ при этом отображается на c_1, ξ_1 и $c_2\xi_2$ соответственно.

Сумма /1/ принадлежит клону L_s тогда и только тогда, когда $a_1+\ldots+a_n=1$.

Действительно $\Pi_1(x)=x+1$, $\Pi_2(x)=x+2$, и по условию

$$a_o+a_1(x_1+1)+\ldots+a_n(x_n+1)=a_o+a_1x_1+\ldots+a_nx_n+1,$$

$$a_o+a_1(x_1+2)+\ldots+a_n(x_n+2)=a_o+a_1x_1+\ldots+a_nx_n+2,$$

откуда получаем $a_o+1=a_o+a_1+\ldots a_n$, $a_o+2=a_o+2a_1+\ldots+2a_n$.

Достаточность указанного условия очевидна.

Применяя к $2x_1+2x_2+1$ операцию* последовательно $4n-2$ раза, получим выражение

$$2(\ldots2(2(2x_1+2x_2+1)+2x_3+1)+\ldots+2x_{4n}+1=2x_1^{4n-1}+2x_2^{4n-1}+\ldots+2x_{4n}^{4n-1}+2 . \qquad /2/$$

Из $2^k=(3-1)^k=3^k-\frac{3^{k-1}}{2}+\ldots\pm1$ видим, что по mod 3

$$2^1=2,\ 2^2=1,\ 2^3=2,\ 2^4=1,\ldots,$$

поэтому выражение /2/ можно переписать в виде

$$2x_1+2x_2+x_3+2x_4+\ldots+2x_{4n}+2 .$$

Переставляя переменные с помощью операций ξ, τ, преобразуем эту сумму к виду

$$a_1x_1+a_2x_2+\ldots+a_nx_n+b_1x_{n+1}+\ldots+b_{3n}x_{4n}+2 .$$

Так как эта операция принадлежит L_s, то $\sum_{i=1}^{n} a_i + \sum_{i=1}^{3n} b_i = 1$. В то же время $a_o+a_1x_1+\ldots+a_nx_n$ также принадлежит L_s, поэтому $\sum_{i=1}^{3n} b_i = 0$. Отождествив переменные $x_{n+1},\ldots,x_{4n}$, получим сумму

$$a_1x_1+\ldots+a_nx_n+2. \qquad /3/$$

Отождествляя x_1 и x_2, из $2x_1+2x_2+1$ получаем $x+1$, а из нее суперпозицией $x+2$. Подствляя в /3/ вместо x_1 операцию e_1^1, если $a_o=2$, $x+1$, если $a_o=0$, $x+2$, если $a_o=1$, получаем сумму /1/, в которой $\sum_{i=1}^{n} a_i = 1$, что и требовалось.

Обозначим через L' пересечение клонов L_o, L_1, L_2 и L_s, через Q_i-клон, порожденный операциями ξ_i и c_i, а через R-клон, порожденный операцией Π_1.

Лемма 4. Для $i\in\{0,1,2\}$ максимальными подклонами клона L_i являются клоны Q_i и L' и только они. Клон L_s имеет два максимальных подклона: L'и R.

Доказательство. Клон L' содержит операции, сохраняющие все константы и принадлежащие L_s. Этот клон не содержит одноместных операций, отличных от e_1^1, но содержит двуместную операцию $t(x_1,x_2)=2x_1+2x_2$.

Пусть $f\in L_o\setminus L'$. Это означает, что $f(0,\ldots,0)=0$, и либо $f(c_j(x),\ldots,c_j(x))\neq c_j(x)$ для некоторого $j\neq 0$, либо $f(\Pi_\ell(x_1),\ldots,\Pi_\ell(x_n))\neq(\Pi_\ell * f)(x_1,\ldots,x_n)$ для подходящего $\ell\in\{1,2\}$.

а/ Пусть $f(c_j,\ldots,c_j)\neq c_j$. Отсуда следует, что $f(x,\ldots,x)\neq e_1^1(x)$. Если $f(x,\ldots,x)$ принимает только одно значение, то она совпадает с c_o. Если же $f(x,\ldots,x)\neq c_o(x)$, то $f(x,\ldots,x)=\xi_o(x)$. Однако и в этом случае $t(x,\xi_o(x))=c_o(x)$.

Из $2x_1+2x_2$ суперпозицией получаем $4x_1+4x_2+2x_3=x_1+x_2+2x_3$; под-

ставляя c_o вместо x_3 и отождествляя x_2 и x_3, получим x_1+x_2, что и требовалось.

б/ Пусть $f(\Pi_\ell(x_1),\ldots,\Pi_\ell(x_n))\neq(\Pi_\ell * f)(x_1,\ldots,x_n)$, тогда $f(x,\ldots,x)\neq x$. Так как $f\in L_o$, то $f(0,\ldots,0)=0$. Опять имеем только две возможности: либо $f(x,\ldots,x)=c_o(x)$, либо $f(x,\ldots,x)=\xi_o(x)$. Далее рассуждаем, как и в пункте а/.

Пусть теперь $f\in L_o\setminus Q_o$. Это означает, что операция f существенно многоместная, сохраняющая c_o, то есть $f(x_1,\ldots,x_n)=b_1x_1+\ldots+b_mx_m$. Нужно используя эту сумму и операции c_o, ξ_1, получить сумму $a_1x_1+\ldots+a_nx_n$. Сначала подставляя f в себя, получим сумму, длина которой не меньше n. Затем, используя c_o и отождествляя переменные, получим сумму $d_1x_1+\ldots+d_nx_n$, длина которой равна n. Наконец, подставляя вместо x_i операцию $e_1^1(x_i)$, если $a_i=d_i$ и $\xi_1(x_i)=2x_i$, если $a_i\neq d_i$, получим сумму $a_1x_1+\ldots+a_nx_n$, что и требовалось.

Докажем, что других максимальных подклонов в клоне L_o нет. Пусть система операций $C\subseteq L_o$ не содержится целиком ни в L' ни в Q_o. В C содержится

(i) операция f, либо не сохраняющая c_j для некоторого $j\neq 0$, либо такая, что $f(\Pi_\ell(x_1),\ldots,\Pi_\ell(x_n))\neq(\Pi_\ell * f)(x_1,\ldots,x_n)$ для подходящего $\ell\in\{1,2\}$;

(ii) существенно многоместная операция $g(x_1,\ldots,x_m)=b_1x_1+\ldots+b_mx_m$. Условие (i) означает, что $f(x,\ldots,x)\neq e_1^1(x)$, поэтому $f(x,\ldots,x)$ совпадет либо с $c_o(x)$, либо с $\xi_o(x)$. Пусть $f(x,\ldots,x)=c_o(x)$. Если у операции (ii) $b_j=2$ для некоторого $j\in\{1,\ldots,m\}$, то, подставляя вместо x_i, $i\neq j$, операцию c_o и отождествляя переменные, получим из g операцию ξ_o. Если же все b_j равны 1, то в сумме $g_1=\Delta g$ первый коэффициент равен 2. Ввиду максимальности клона Q_o в L_o операции ξ_o, c_o, g порождают L_o.

Пусть $f(x,\ldots,x)=\xi_o(x)$.

а/ В сумме $b_1x_1+\ldots+b_mx_m$ число слагаемых равно $3k$ или $3k+2$. Подставляя вместо x_i операцию $e_1^1(x)$, если $b_i=2$, ξ_o, если $b_i=1$, получим $c_o(x)$.

б/ $m=3k+1$. С переменными $x_2,\ldots,x_m$ поступаем так же, как и в пункте а/, а вместо x_1 подставляем $e_1^1(x)$, если $b_1=1$, $\xi_o(x)$, если $b_1=2$. Получим $c_o(x)$.

Автоморфизмы клона L_1 порождаемые операциями ξ_1 и ξ_2 оставляют клон L' неподвижным, а клон Q_o отображают на Q_1 и Q_2 соответственно, поэтому клоны Q_i, L' и только они являются максимальными подклонами клона L_i, $i=1,2$.

Докажем теперь, что L' и R являются максимальными подклонами клона L_s. Пусть $f\in L_s\setminus L'$. Это означает, что $f(c_i(x),\ldots,c_i(x))\neq c_i(x)$ для некоторого $i\in\{0,1,2\}$. Отсюда следует, что $f(x,\ldots,x)\neq e_1^1(x)$. Остаются две возможности: $f(x,\ldots,x)=\Pi_1(x)$ или $f(x,\ldots,x)=\Pi_2(x)$. Далее, $\Pi_1*\Pi_1=\Pi_2$, $2x_1+2x_2$ принадлежит L', $2x_1+2\Pi_2(x_2)=2x_1+2x_2+4=2x_1+2x_2+1$.

Пусть $f\in L_s\setminus R$. В этом случае f_n - существенно многоместная операция, имеющая вид /1/, где $\sum_{i=1} a_i=1$. Предположим, что $a_1\neq 1$. Отождествив переменные $x_2,\ldots,x_n$, получим операцию $a_o+a_1x_1+a_2'x_2$. Так как $a_1+a_2'=1$ и $a_1=2$, то $a_2=2$. Если же все коэффициенты $a_1,\ldots,a_n$ равны единице, то $n>2$. Отождествив x_1 и x_2 получим существенно многоместную операцию, у которой первый коэффициент в сумме равен 2.

Итак, мы получим операцию $a_o+2x_1+2x_2$. Подставив вместо x_1 операцию $\xi_1(x_1)$, если $a_o=0$, $\xi_2(x_1)$, если $a_o=2$, $e_1^1(x_1)$, если $a_o=1$, получим операцию $2x_1+2x_2+1$.

Докажем наконец, что клон L_s имеет только два максимальных подклона. Пусть C-система операций из клона L_s, не содержащая целиком в клонах L' и R. Система C содержит существенно много-

местную операцию f и операцию g, не сохраняющую константу c_i для подходящего i /эти операции могут совпадать/. Как было показано ранее, из операции g отождествлением получаем Π_1 или Π_2, которые порождают клон R. Так как R максимален в L_s, то вместе с f входящие в него операции порождают L_s.

Лемма 5. Клон L′ порождается операцией $2x_1+2x_2$. Единственным максимальным подклоном клон L′ является клон E.

Доказательство. Каждая принадлежащая клону L′ операция принадлежит также клонам L_o, L_1, L_2 и L_s, поэтому она может быть представлена суммой /1/, в которой $a_o=0$ и $\sum_{i=1}^{n} a_i=1$. Применяя к $2x_1+2x_2$ операцию * последовательно 4n-2 раза, получим сумму

$$2x_1+2x_2+x_3+2x_4+\ldots+2x_{4n}$$

/см. доказательство леммы 3/. Далее переставляя переменные, приводим сумму к виду

$$a_1x_1+a_2x_2+\ldots+a_nx_n+b_1x_{n+1}+\ldots+b_{3n}x_{4n}.$$

Отождествив переменные $x_{n+1},\ldots,x_{4n}$, получим сумму /1/, в которой $a_o=0$.

При n=2 и $a_o=0$ возможны лишь 4 варианта суммы /1/ /при условии $a_1\neq0$ и $a_2\neq0$/, из них лишь 2x+2y принадлежит L′. Отсюда следует, что любой подклон клона L′, содержащий существенно многоместные операции, содержит 2x+2y и потому совпадает с L′.

Из лемм 1-5 следует истинность следующего утверждения.

Теорема 2. Подклоны клона L образуют решетку, изображенную на рисунке 10.

Минимальные и существенно минимальные ТС-клоны.

Построенные решетки позволяют легко найти все минимальные и существенно минимальные ТС-клоны на трехэлементном множестве.

Автоморфизмы клона Бурле, порождаемые операциями $\xi_o(x)=2x$ и $\xi_1(x)=2x+1$, отображают клон Z на клоны Z_o и Z_1 соответственно. Эти клоны состоят из всех операций, принадлежащих клону B и принимающих значения из множеств $\{0,2\}$ /клон Z_o/ и $\{1,2\}$ /клон Z_1/. В попарных пересечениях клонов Z, Z_o и Z_1, содержатся лишь константы.

Пусть $C\subseteq P_A$. Через $[C]$ обозначим клон, порождаемый операциями из C. Введем следующие обозначения:

$$M_\zeta=[\zeta,e_1^1],\ M_\psi=[\psi,e_1^1],\ M_i=[c_i,e_1^1]\ (i=0,1,2),$$

$$M_i^\xi=[\xi_i]\ (i=0,1,2),\quad M^\Pi=[\Pi_1].$$

Теорема 3. При $|A|=3$ минимальными TC-клонами на A являются клоны M_φ, M_ψ, клоны, двойственные к ним относительно ξ_o и ξ_1, а клоны M_o, M_1, M_2, M_o^ξ, M_1^ξ, M_ξ^2 и клон M^Π. Существенно минимальными являются клоны L_ζ, L_ψ, $L_{\zeta+\gamma}$, J_{20}^o, L_{02}^1, клоны, двойственные к ним относительно ξ_o и ξ_1, и клоны L'.

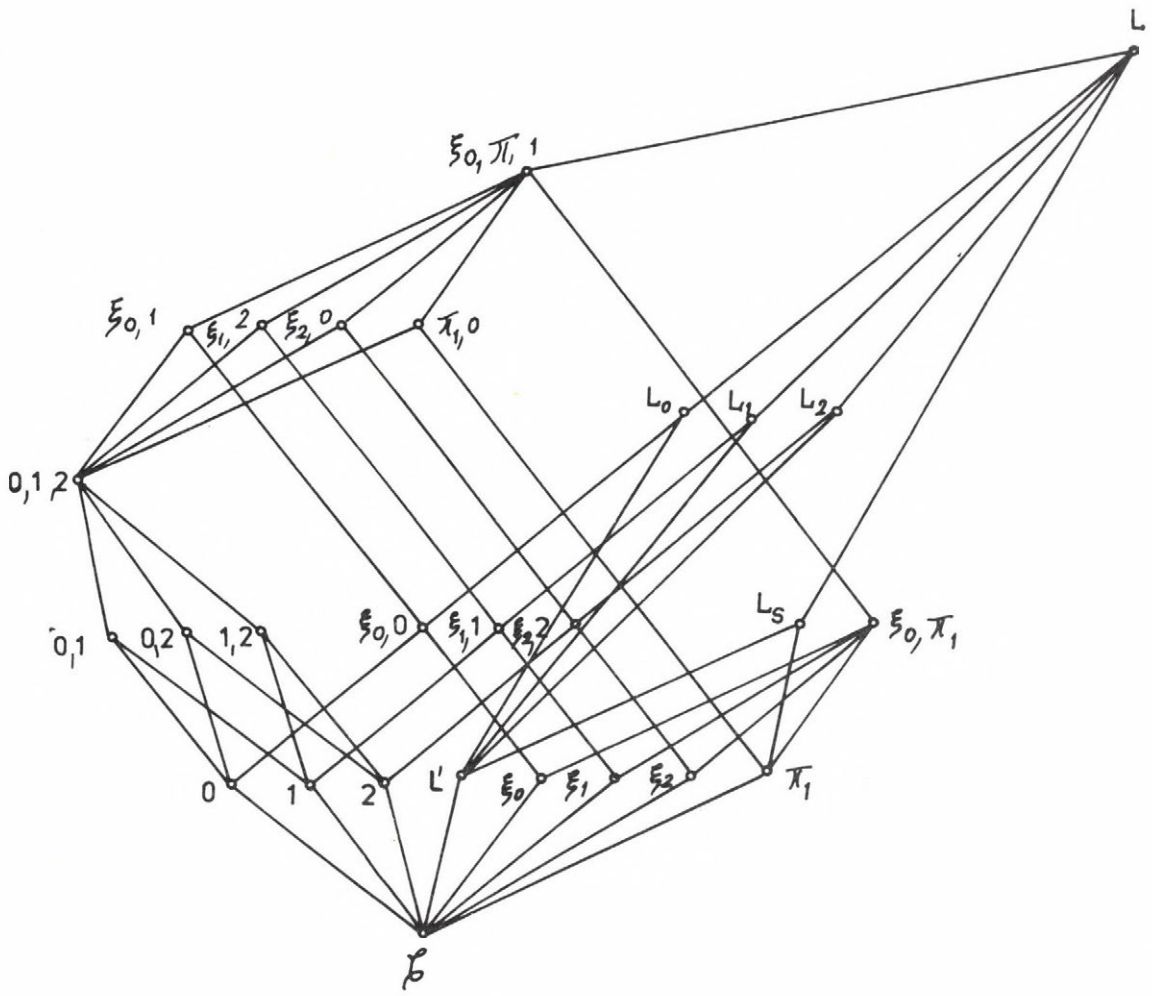


Рисунок 10.
Решётка L(L)

Л И Т Е Р А Т У Р А

1. Bagyinszki J., Demetrovics J.: The structure of linear classes in prime-valued logics./Hungarian/ MTA SZTAKI, Közlemények, 16/1976/, 25-52.

2. Bagyinszki J., Demetrovics J.: The lattice of linear classes in prime-valued logics. Discrete mathematics. Banach center publication, volume 7, Warsaw , /1982/, 105-123.

3. Berman J., Mc Kenzie R.: Clones satisfying the term condition. Preprint, 1983.

4. Бурле Г.А.: Классы k-значных логик, содержащие все функции одной переменной. Дискретный анализ, 10/1967/, 3-7.

5. Мальцев А.И.: Итеративные алгебры Поста. Новосибирск, 1976.

6. Мальцев И.А.: Некоторые свойства клеточных подалгебр алгебры Поста и их основных клеток. Алгебра и логика, 11, № 5 /1972/, 571-587.

7. Мальцев И.А.: Конгруэнции и автоморфизмы на клетках алгебр Поста. Алгебра и логика, 11, № 6 /1972/, 666-672.

8. Мальцев И.А.: Некоторые свойства клеток алгебр Поста. Дискретный анализ, 23 /1973/, 24-31.

9. Мальцев И.А.: О конгруэнциях на подалгебрах итеративных алгебр Поста. Методы дискретного анализа в теории кодов и схем. 29 /1976/, 40-52.

10. Machida H.: Toward a classification of minimal closed sets in 3-valued logic. Proceedings of the 12-th international symposium on multiple-valued logic. Paris, /1982/,313-317.

11. Post E.: The two-valued iterative systems of mathematical logic. Annals Math. Studies 5, Princeton, 1941.

12. Taylor W.: Some appelications of the term conditions. Algebra Universalis /to appear/.

13. Csákány B.: All minimal clones on theree-element set. Prepint. CRMA-1136, Montreal, 1982.

14. Яблонский С.В., Гаврилов Г.П., Кудрявцев В.Б.: Функции алгебры логики и классы Поста. М., "Наука", 1966.

S U M M A R Y

ESSENTIAL MINIMAL TC CLONES IN THE 3-VALUED LOGICS

J. Demetrovics - I.A. Malcev

In this paper the authors describe the essential minimal TC clones. Moreover, structure of all TC clones having no more than 2 values in analysed. Finally, structure of all the linear classes is given.

Ö S S Z E F O G L A L Á S

LÉNYEGESEN MINIMÁLIS TC KLONOK A 3-ÉRTÉKÜ LOGIKÁBAN

Demetrovics J. - Malcev I.A.

Dolgozatunkban sikerül meghatározni és pontosan leirni a lényegesen minimális TC klónokat. Ezenkivül meghatározzuk az összes olyan TC klónnak a strukturáját, amely legfeljebb 2 értéket vesz fel. Végül megadjuk az összes lineáris osztály strukturáját is.

 Közlemények 31/1984 153-158

МОНОТОННЫЕ СИСТЕМЫ НА МАТРИЦАХ ДАННЫХ

Е.Н. Кузнецов, И.Б. Мучник, Г. Хенчей, Н.Ф. Чкуасели

1. Монотонные системы /МС/ для агрегирования данных /в частности, для решения задач типа задачи классификации множества объектов/ предложены И.Э. Муллатом в 1971 г. [1, 2]. Метод МС характеризуется большой общностью подхода к задачам такого типа, но в отличии от других методов требует задания числовой функции связи между отдельным элементом и любым подмножеством исходного множества - функции связи "элемент-подмножество" /ФСЭП/. Сам метод МС средств порождения таких функций, а тем самым, различных монотонных систем не содержит. В практических исследованиях [3, 4] использовалось лишь несколько конкретных МС. В данной работе ставится задача показать регулярные способы построения монотонных систем для реализации различных макроописаний матрицы данных и создания единого программного обеспечения для такого анализа.

2. МС и задача структуризации. Монотонной системой $\langle W,\Pi\rangle$ называется множество W, $|W|=N$, с заданной на нем функцией /ФСЭП/ $\Pi(i,H)$, $i\in H$, $H\subseteq W$, удовлетворяющей неравенству $\Pi(i,H\setminus j)\leq\Pi(i,H)$, $\forall i\in H\setminus j$, $\forall j\in H$, $\forall H\subseteq W$ /свойство монотонности/ [2]. Ядром МС называется множество $G\subseteq W$, для которого $F(G)=\max\limits_{H\subseteq W} F(H)$, где $F(H)=\min\limits_{i\in H}\Pi(i,H)$ [1].

Для выделения наибольшего по мощности ядра МС предложены эффективные алгоритмы [1, 2, 5]. В ходе работы некоторых из них строится последовательность вложенных множеств $\bar{\Gamma}=\langle\Gamma_1,\Gamma_2,\ldots,\Gamma_p\rangle$, $\Gamma_1=W$, $\Gamma_p=G$, являющихся локальными максимумами функции $F(H)$ [2].

Для структуризации исходного множества W используется как разбиение его на две части: G и $W\setminus G$, так и на подмножества $\Gamma_1\setminus\Gamma_2$, $\Gamma_2\setminus\Gamma_3$,...,$\Gamma_{p-1}\setminus\Gamma_p$, Γ_p . Кроме того, если функция $\Pi(i,H)$

имеет смысл расстояния, то G можно рассматривать как множество эталонов-классообразующих элементов искомой классификации. Разбиение W на "однородные" части получают также путем последовательного построения МС<W\G,Π> на множестве W\G элементов, не вошедших в ядро предыдущей системы, затем на оставшихся вне нового ядра и т.д. /до исчерпания исходного множества/.

3. Выбор множества W элементов МС на матрице данных $\Phi = \| \varphi_{ij} \|$ (NxM). Есть четыре возможности [6]: а/ множество X объектов - строк Φ; б/ множество Y признаков - столбцов Φ; в/ X U Y; г/ множество пар (i,j), $i = \overline{1,N}$, $j = \overline{1,M}$. При этом произвольному подмножеству H множества W /в т.ч. ядру G/ соответствует часть Φ(H) матрицы Φ определенной конфигурации: а/ горизонтальная полоса, б/ вертикальная полоса, в/ "крест" или его прямоугольный блок - пересечение, г/ "пятно" произвольной формы либо минимальный охватывающий его прямоугольный блок.

Эти четыре способа можно использовать и комбинированно, например, сначала представив матрицу разделенной на "однородные" вертикальные полосы, а затем - внутри каждой из них независимо - на горизонтальные блоки, или наоборот. /Можно менять характер элементов МС и после каждого выделения ядра, получая макроописание матрицы типа "паркет./

Вопрос о связи МС на множестве X, Y и X U Y частично решают следующие теоремы. Введем обозначения: $\forall H \subseteq X \cup Y$, $H_X = H \cap X$, $H_Y = H \cap Y$; $\Phi(H_Y)$ - подматрица матрицы Φ, определяемая столбцами H_Y; $\Pi(i,H,\Phi)$ - ФСЭП $\Pi(i,H)$, вычисляемое по матрице Φ.

Теорема 1. Если G - ядро МС $<W,\Pi^1>$, где

$$\Pi^1(i,H,\Phi) = \begin{cases} \Pi(i,H_X,\Phi), & \text{если } i \in H_X \subseteq X, \\ \Pi(i,H_Y,\Phi^T), & \text{если } i \in H_Y \subseteq Y, \end{cases}$$

а G^X и G^Y - ядра, соответственно, МС $\langle X,\Pi,\Phi\rangle$ и $\langle Y,\Pi,\Phi^T\rangle$, то верно одно из трех

а/ $G = G^X$, $G^Y \subseteq W\backslash G$, если $F(G^X) > F(G^Y)$,

б/ $G = G^Y$, $G^X \subseteq W\backslash G$, если $F(G^X) < F(G^Y)$,

в/ $G = G^X \cup G^Y$, если $F(G^X) = F(G^Y)$.

Теорема 2. Если для ядра G^X МС $\langle X,\Pi,\Phi\rangle$ на множестве строк X матрицы Φ выполняется $F(G^X,\Phi) \le F(Y,\Phi^T(G^X))$, где

$F(G^X,\Phi) = \min\limits_{i\in G^X} \Pi(i,G^X,\Phi)$, $F(Y,\Phi^T(G^X)) = \min\limits_{j\in Y} \Pi(j,Y,\Phi^T(G^X))$, то

$G_X = G \cap X = G^X$, где G - ядро МС $\langle W,\Pi^1\rangle$, удовлетворяющей

$$\Pi^1(i,H,\Phi) = \begin{cases} \Pi(i,H_X,\Phi(H_Y)), & \text{если } i\in H_X \subseteq X, \\ \Pi(i,H_Y,\Phi^T(H_X\backslash j)), & \text{если } i\in H_Y \subseteq Y. \end{cases}$$

Таким образом, МС на $W=X \cup Y$ при некоторых условиях можно конструировать из "частных" МС, определенных на X и Y отдельно, либо как МС только на множестве X′ строк или Y′ столбцов другой матрицы, составленной из исходной матрицы Φ и матрицы Φ^T, полученной транспорированием исходной. В случае г/ исходную матрицу Φ можно интерпретировать как двудольный граф с весами и строить МС на дугах графа 1, 7.

4. Три типа функций связи "элемент-подмножество".

1/ суммарно-попарные ФСЭП $\Pi(i,H) = \sum\limits_{k\in H} a_{ik}$, где a_{ik} - мера парной связи i-го и k-го элементов, в частности один из известных коэффициентов связи двух векторов /коэффициент корреляции признаков, число общих признаков объектов, расстояние Хэмминга и др./ [1, 5]; 2/ представительские ФСЭП $\Pi(i,H) = a_{ih}$, где a_{ih} - мера парной связи, а h - элемент, быть может, специально сконструированный, - "представитель" множества H /например, для булевой матрицы Φ признаки элемента - представителя h могут определяться как $\varphi_{hj} = \bigcap\limits_{k\in H} \varphi_{kj}$ или $\varphi_{hj} = \bigcup\limits_{k\in H} \varphi_{kj}$ [3];

3/ суммарно-признаковые ФСЭП $\Pi(i,H) = \sum_{j \in Y_i} \omega_j(H)$, где $\omega_j(H)$ - вес признака j на множестве элементов H, Y_i - множество признако i-го объекта /например, $\omega_j(H) = \sum_{k \in H} \varphi_{kj}$ - число объектов из H, имеющих этот признак [6]. На компоненты ФСЭП каждого типа накладываются ограничения, необходимые для выполнения свойства монотонности /например, в 1/ - $a_{ik} \geq 0$, в 3/ - $\omega_j(H \backslash k) \leq \omega_j(H)$. В работе рассматривается целая "коллекция" конкретных примеров ФСЭП каждого типа.

Границы между ФСЭП разных типов всегда жесткие.

Пример. Пусть $a_{ik} = \sum_{j \in Y} \varphi_{ij} \cdot \varphi_{kj}$ - число общих признаков пары объектов /Ф - булева/, а $\omega_j(H) = \sum_{k \in H} \varphi_{kj}$ - число объектов в H с j-ым признаком. Тогда $\Pi^{1)}(i,H) = \sum_{k \in H} a_{ik}$ и $\Pi^{2)}(i,H) = \sum_{j \in Y_i} \omega_j(H)$ совпадают.

5. Порождение параметрических семейств монотонных систем.

Теорема 3. Если $\langle W, \Pi_1 \rangle$ и $\langle W, \Pi_2 \rangle$ - МС, то $\langle W, \Pi \rangle$ - тоже МС, если

$$\Pi(i,H) = \alpha_1 \Pi_1(i,H) + \alpha_2 \Pi_2(i,H), \quad \alpha_1, \alpha_2 \geq 0.$$

Теорема 4. Если $\langle W_1 \Pi_1 \rangle$ и $\langle W_1 \Pi_2 \rangle$ - МС, то $\langle W, \Pi \rangle$ - тоже МС, если

$$\Pi(i,H) = [\Pi(i,H)]^{\beta_1} \cdot [\Pi_2(i,H)]^{\beta_2}, \quad \beta_1, \beta_2 \geq 0.$$

Очевидны обобщения на произвольное число базовых МС. Возможна и внутренняя параметризация, например,

$$\Pi(i,H) = \sum_{k \in H} a_{ik}^{\gamma} \quad \text{или} \quad \Pi(i,H) = \sum_{j \in Y} [\omega_j(H)]^{\delta}, \quad \gamma, \delta \geq 0$$

Легко видеть, таким образом, что в ФСЭП I-го и III-го типов можно использовать не сумму, а любую симметричную по аргумен-

там, т.е. a_{ik} или $\omega_j(H)$, функцию, например,

$$\Pi(i,H) = \sum_{s\in H} \alpha_s \cdot \prod_{k\in H} \alpha_k \cdot (a_{ik})^{\gamma_s}$$

Таким образом, имея разные базовые ФСЭП трех типов и варьируя значения параметров, можно строить разнообразные МС с наперед заданными содержательными свойствами и, тем самым, получать разные макроописания исходных данных.

Л И Т Е Р А Т У Р А

1. Муллат И.Э.: Экстремальные подсистемы монотонных систем I-III.- Автоматика и телемеханика, 1976, №5, с. 130-139; №8, с. 169-178; 1977, №1, с. 109-119.

2. Кузнецов Е.Н., Мучник И.Б., Шварцев Л.В.: Монотонные системы и их свойства. - В кн.: Анализ нечисловой информации в социологических исследованиях. М.: Наука, 1984, с. 123-149.

3. Кузнецов Е.Н., Мучник И.Б.: Анализ распределения функций в организационной системе. - Автоматика и телемеханика, 1982, №10, с. 119-127.

4. Выханду Л.К., Выханду П.Л.: Быстрый поиск на битматрицах. - В кн.: Труды Таллинского политехн. ин-та, 1983, №554, с. 49-60.

5. Кузнецов Е.Н.: Анализ структуры матрицы связей с помощью построения на ней монотонной системы. - Автоматика и телемеханика, 1980, №7, с. 128-136.

6. Выханду Л.К.: Монотонные системы в анализе данных. - В кн.: Труды Таллинского политехн. ин-та, 1981, №511, с. 91-100.

7. Мучник И.Б.: Обработка экспертных суждений о структурных объектах. - В кн.: Экспертные оценки в задачах управления. Ин-т проблем управления, 1982, с. 27-32.

S U M M A R Y

E.N. Kuznetzov, I.B. Muchnik, G. Hencsey, N.F. Tchkuasely

MONOTONIC SYSTEMS ON DATA MATRICES

In the paper a technique for the structuralization of large-scale data matrices is considered, namely the use of the theory of monotonic systems. This is expected to be highly suitable for different applications.

For the method of monotonic systems a numerical function coupling any element with any subject of the basic set is needed.

We discuss three main classes of such functions and four possibilities of determining them for data matrices.

An idea of a generator for parameter families of such functions defining monotonic systems is suggested.

Ö S S Z E F O G L A L Á S

E.N. Kuznetzov, I.B. Muchnik, G. Hencsey, N.F. Chkuasely

MONOTON RENDSZEREK ADAT-MÁTRIXOKON

A cikkben a szerzők a nagyméretü adat-mátrixok strukturalizálására adnak meg egy technikát, amely a monoton rendszerek módszerén alapszik. Ez várhatóan igen hasznos lesz különböző alkalmazásokra.

A monoton rendszerek módszere egy pont-halmaz függvény megadásán alapszik.

A szerzők ilyen függvényeknek három osztályát, illetve adat-mátrixokon való négy fajta megadását tárgyalják. Ezen függvények paraméteres családjainak generálását is megemlitik.