ALGORYTMY

VOL. 1 • NO 2 • 1963

INSTYTUT MASZYN MATEMATYCZNYCH PAN

A L G O R Y T M Y

Vol. I N°2 1963

P R A C E

Instytutu Maszyn Matematycznych

P o l s k i e j A k a d e m i i N a u k

Adres redakcji: Warszawa, ul.Koszykowa 79, tel. 8-37-29

TREŚĆ

CONTENTS

NUMERICAL ANALYSIS

Instytut Maszyn Matematycznych PAN
Algorytmy N° 2

517.55:517.91

NOTES ON ASYMPTOTIC DISTRIBUTION OF EIGENVALUES IN THE STURM-LIOUVILLE PROBLEM AND RELATED RESULTS.

by Krzysztof MOSZYŃSKI

Received October 1962

The first part of this paper contains some remarks on the asymptotic distribution of eigenvalues in the classic Sturm-Liouville Problem for ordinary differential equations. In the second part, simple sufficient conditions for differentiability term by term of the generalized Fourier Series, associated with the above Problem, are given. The demonstration of the convergence for a simple recurrent algorithm solving the Sturm-Liouville Problem is presented as the application of obtained results.

PART I

Consider the equation

$$\ddot{u}(x) + (\lambda - q(x))\,u(x) = 0 \qquad /1/$$

for $x \in [a, b]$, with the boundary conditions

$$\begin{aligned} u(a)\cos\alpha + \dot{u}(a)\sin\alpha &= 0, \\ u(b)\cos\beta + \dot{u}(b)\sin\beta &= 0, \end{aligned} \qquad /2/$$

where a, b, α, and β are real numbers.

We shall discuss the case for real, continuous $q(x)$, bounded in $[a, b]$.

Let $\varphi(x)$ and $\vartheta(x)$ denote such a pair of solutions of /1/ that

$$\begin{aligned} \varphi(a) &= \sin\alpha, \\ \dot{\varphi}(a) &= -\cos\alpha, \\ \vartheta(b) &= \sin\beta, \\ \dot{\vartheta}(b) &= -\cos\beta, \end{aligned}$$

then the following formulae hold [1]

$$\varphi(x) = \cos s(x-a)\sin\alpha - \frac{\sin s(x-a)}{s}\cos\alpha + \frac{1}{s}\int_a^x \sin s(x-y)\, q(y)\,\varphi(y)\,dy$$

$$\vartheta(x) = \cos s(x-b)\sin\beta - \frac{\sin s(x-b)}{s}\cos\beta - \frac{1}{s}\int_x^b \sin s(x-y)\, q(y)\,\vartheta(y)\,dy \qquad /3/$$

where $s^2 = \lambda$.

By differentiation of /3/ we obtain

$$\dot\varphi(x) = -s\sin s(x-a)\sin\alpha - \cos s(x-a)\cos\alpha + \int_a^x \cos s(x-y)\, q(y)\varphi(y)\,dy,$$

$$\dot\vartheta(x) = -s\sin s(x-b)\sin\beta - \cos s(x-b)\cos\beta - \int_x^b \cos s(x-y)\, q(y)\vartheta(y)\,dy. \qquad /4/$$

The Wronskian $\omega(\lambda) = \begin{vmatrix} \varphi(x) & \vartheta(x) \\ \dot\varphi(x) & \dot\vartheta(x) \end{vmatrix}$ is an analytic function of the only variable λ. We shall denote the zeros of this Wronskian by λ_r these being the eigenvalues of the problem stated by /1/ and /2/.

It is well known that all λ_n - s are real and simple. Only a finite number of them can be negative. The corresponding functions $\varphi(x)$, when normalized, form an orthogonal normalized system $\{\psi_n(x)\}$ of eigenfunctions of the problem /1/, /2/.

Let us express the Wronskian $\omega(\lambda)$ in terms of $\varphi(x)$ and $\vartheta(x)$. Note that

$$\omega(\lambda) = s\sin\alpha\sin\beta\left[\sin s(b-a) + r(s)\right], \qquad /5/$$

where

$$r(s) = \frac{1}{s}\left[A\cos s(b-a) + B\int_a^b \cos s(y-a)\, q(y)\vartheta(y)\,dy\right] +$$

$$+ \frac{1}{s^2}\left[C\sin s(b-a) + D\int_a^b \sin s(y-a)\, q(y)\vartheta(y)\,dy\right]$$

and

$$A = \frac{\sin(\beta-\alpha)}{\sin\alpha\cdot\sin\beta}, \qquad B = \frac{-1}{\sin\beta},$$

$$C = \operatorname{ctg}\alpha\cdot\operatorname{ctg}\beta, \qquad D = \frac{\operatorname{ctg}\alpha}{\sin\beta}. \qquad /6/$$

Let s be a real number; this corresponds to positive λ. By simple differentiation:

$$\left.\begin{aligned}\frac{dr}{ds} = \frac{1}{s}\Big[-A(b-a)\sin s(b-a) + B\Big(\int_a^b \cos s(y-a)\, q(y)\frac{\partial \vartheta}{\partial s}(y)dy + \\ - \int_a^b (y-a)\sin s(y-a)\, q(y)\vartheta(y)\,dy\Big] + \frac{1}{s^2}\Big[(C(b-a) - A)\cos s(b-a) + \\ - \int_a^b (D(y-a) - B)\cos s(y-a)\, q(y)\vartheta(y)dy + D\int_a^b \sin s(y-a)\, q(y)\frac{\partial \vartheta}{\partial s}(y)dy\Big] + \\ - \frac{2}{s^3}\Big[C\sin s(b-a) + D\int_a^b \sin s(y-a)\, q(y)\vartheta(y)dy\Big].\end{aligned}\right\} \quad /7/$$

For real s we have /of. also [1]/

$$|\vartheta(x)| \leqslant |\sin\beta| + \frac{|\cos\beta|}{|s|} + \frac{1}{|s|}\int_x^b |q(y)|\,|\vartheta(y)|dy.$$

If $\mu(s) = \sup_{x\in[ab]} |\vartheta(x)|$ we have

$$\mu(s) \leqslant |\sin\beta| + \frac{|\cos\beta|}{|s|} + \frac{\mu(s)}{|s|}\int_a^b |q(y)|\,dy,$$

and for sufficiently large values of $|s|$

$$\mu(s) \leqslant \frac{|\sin\beta| + \frac{|\cos\beta|}{|s|}}{1 - \frac{1}{|s|}\int_a^b |q(y)|\,dy} = M(s). \qquad /8/$$

The same method applied to $\frac{\partial \vartheta}{\partial s}$ gives

$$\nu(s) \leqslant \frac{(b-a)\left[|\sin\beta| + \frac{|\cos\beta|}{|s|}\right]}{\left[1 - \frac{1}{|s|}\int_a^b |q(y)|\,dy\right]^2} = N(s), \qquad /9/$$

where

$$\nu(s) = \sup_{x\in[ab]}\left|\frac{\partial \vartheta(x)}{\partial s}\right|.$$

Hence the functions $\vartheta(x)$ and $\frac{\partial \vartheta(x)}{\partial s}$ are bounded for sufficiently large values of $\lambda > 0$.

Using above estimates we get from /6/ and /7/:

$$|r(s)| \leq \frac{1}{|s|}\left[|A| + |B| M(s) \int_a^b |q(y)|\,dy\right] + \frac{1}{|s|^2}\left[|C| + |D| \cdot M(s) \int_a^b |q(y)|\,dy\right], \qquad /10/$$

$$\left|\frac{dr(s)}{ds}\right| \leq \frac{1}{|s|}\left[|A| \cdot (b-a) + |B| \cdot \int_a^b |q(y)|\,dy\,(N(s) + (b-a)M(s))\right] +$$
$$+ \frac{1}{|s|^2}\left[|C(b-a) - A| + \left(|D(b-a) - B| M(s) + |D| N(s)\right) \int_a^b |q(y)|\,dy\right] + \qquad /11/$$
$$+ \frac{2}{|s|^2}\left[|C| + |D| M(s) \int_a^b |q(y)|\,dy\right].$$

Hence we have

$$r(s) = 0\left(\frac{1}{|s|}\right), \qquad /12/$$
$$\frac{dr(s)}{ds} = 0\left(\frac{1}{|s|}\right),$$

as $|s| \longrightarrow \infty$.

Now, consider the equation

$$F(s) \equiv \sin s(b-a) + r(s) = 0^{*)}. \qquad /13/$$

The zeros of this equation are the square roots of the eigenvalues of the problem /1/, /2/ /may be with the exception of zero/.

Put $s_k = \frac{\pi k}{b-a}$ for integer k.

If we denote the interval $\left[s_k - \frac{\pi}{2(b-a)}, s_k + \frac{\pi}{2(b-a)}\right]$ by I_k, we notice that

$$|\sin s(b-a)| \geq \frac{2}{\pi}(b-a)\,|s - s_k|,$$

for $s \in I_k$.

If s is an arbitrary root of /13/ contained in I_k, then

$$|\sin s(b-a)| = |r(s)|,$$

so that

$$|s - s_k| \leq |r(s)| \frac{\pi}{2(b-a)} = 0\left(\frac{1}{s}\right) = 0\left(\frac{1}{s_k}\right). \qquad /14/$$

---

*The author would like to express his gratitude to Dr K. Bochenek for his remarks, essential to this part of the paper.

On the other hand, in sufficiently close neighbourhood of s_k, and for sufficiently large k, the derivative

$$F'(s) = (b-a)\cos s(b-a) + r'(s) = (b-a)\cos s(b-a) + 0\left(\frac{1}{s_k}\right) \qquad /15/$$

does not change the sign.

Let ε_k be an interval of diameter ε, with the centre in s_k. From /14/ and /15/ we can deduce

/i/ For any ε, $\varepsilon \in \left(0, \frac{\pi}{b-a}\right)$ such integer N exists, that for $k > N$ all zeros of /13/ contained in I_k are contained in ε_k.

/ii/ Such integer N' exists that for $k > N'$ exactly one zero of /13/ is contained in ε_k.

/iii/ If $\bar{s}_k$ is the zero of /13/ contained in ε_k, then

$$\frac{1}{\bar{s}_k} = \frac{1}{s_k + 0\left(\frac{1}{s_k}\right)} = 0\left(\frac{1}{s_k}\right) = 0\left(\frac{1}{k}\right).$$

Thus the series

$$\sum_{n=0}^{\infty} \frac{1}{\lambda_n},$$

where λ_n are the eigenvalues of the problem /1/, /2/, converges absolutely.

By the way, let us notice that the overestimation /14/ for $|s-s_k|$ can be obtained effectively using the inequality /10/. This gives the upper bound of the eigenvalues worth computing with the given accuracy. For greater eigenvalues $\bar{s}_k = s_k$ may be taken. Such information can be of some practical interest but the nature of the problem suggests rather great values for the above mentioned upper bound.

PART II

We shall now investigate the Fourier series expansion of any real funotion $f(x)$ defined in $[a, b]$, related to the problem /1/, /2/

$$f \sim \sum_{n=0}^{\infty} c_n \psi_n(x), \qquad /16/$$

where $\psi_n(x)$ are the normalized eigenfunotions, and

$$c_n = \int_a^b f(x)\, \psi_n(x)\, dx.$$

It is well known that for sufficiently regular $f(x)$ the series /16/ converges in $[a, b]$ to the limit $f(x)$ [1]. If we use the results of PART I some sufficient conditions will derive for the absolute and uniform convergence of the series

$$\sum_{n=0}^{\infty} c_n \dot{\psi}_n(x), \qquad /17a/$$

$$\sum_{n=0}^{\infty} c_n \ddot{\psi}_n(x). \qquad /17b/$$

In this oase the equalities

$$f'(x) = \sum_{n=0}^{\infty} c_n \dot{\psi}_n(x), \qquad /18/$$

$$f''(x) = \sum_{n=0}^{\infty} c_n \ddot{\psi}_n(x),$$

hold.

Denote by $\varphi_n(x)$ the solution $\varphi(x)$ of /1/, defined in PART I, for $\lambda = \lambda_n$.

Then $$\psi_n(x) = \frac{\varphi_n(x)}{\sqrt{\int_a^b \varphi_n^2(x)\, dx}}$$

Next, according to /iii/ we have

$$\frac{1}{\sqrt{\int_a^b \varphi_n^2(x)\,dx}} = \frac{1}{|\sin\alpha|}\sqrt{\frac{2}{b-a}}\left[1 + 0\left(\frac{1}{s_n}\right)\right] \qquad /19/$$

Using the formulae /3/, /4/ and /19/ we derive the following asymptotic expressions for eigenfunctions $\psi_n(x)$ and their derivatives:

$$\left.\begin{aligned}
\psi_n(x) &= \sqrt{\frac{2}{b-a}}\cos\frac{n\pi}{b-a}(x-a) + 0\left(\frac{1}{s_n}\right),\\
\dot{\psi}_n(x) &= -\sqrt{\frac{2}{b-a}}\,\frac{n\pi}{b-a}\sin\frac{n\pi}{b-a}(x-a) + 0(1),\\
\ddot{\psi}_n(x) &= -\sqrt{\frac{2}{b-a}}\,\frac{n^2\pi^2}{(b-a)^2}\cos\frac{n\pi}{b-a}(x-a) + 0(s_n),
\end{aligned}\right\} \qquad /20/$$

as $n \longrightarrow \infty$.

Assuming the derivative $f'(x)$ being integrable in $[a, b]$ we get for $c_n = \int_a^b f(x)\,\psi_n(x)\,dx$:

$$c_n = \frac{1}{\lambda_n}\left\{\dot{\psi}_n(a)\,f(a) - f(b)\,\dot{\psi}_n(b) + \int_a^b f(t)\,q(t)\,\psi_n(t)\,dt + \int_a^b \dot{\psi}_n(x)\cdot f'(x)\,dx\right\}.$$

It can be obtained by integration 'by parts' using the equation /1/. Using the asymptotic formulae /20/ we can deduce:

For $q(x)$ and $f'(x)$ bounded and integrable in $[a, b]$ we have

$$c_n = 0\left(\frac{1}{s_n}\right), \qquad /21/$$

as $n \longrightarrow \infty$.

Integrating once more 'by parts' we get

$$c_n = \frac{1}{\lambda_n}\left\{\begin{vmatrix} f(a), \psi_n(a) \\ f'(a), \dot{\psi}_n(a) \end{vmatrix} - \begin{vmatrix} f(b), \psi_n(b) \\ f'(b), \dot{\psi}_n(b) \end{vmatrix} - \int_a^b \psi_n(t)\left[f''(t) - q(t) f(t)\right] dt\right\}. \qquad /22/$$

/20/ shows that

$$\begin{vmatrix} f(a), \psi_n(a) \\ f'(a), \dot{\psi}_n(a) \end{vmatrix} - \begin{vmatrix} f(b), \psi_n(b) \\ f'(b), \dot{\psi}_n(b) \end{vmatrix} = 0(1),$$

as $n \longrightarrow \infty$.

This gives the following condition:

For $L(f) = f''(x) - q(x) f(x)$ bounded and integrable in $[a, b]$ we have

$$c_n = 0\left(\frac{1}{\lambda_n}\right), \qquad /23/$$

as $n \longrightarrow \infty$.

According to the above results we can deduce:

1. If $f(x)$ satisfies the boundary conditions /2/, $q(x)$ is continuous and bounded in $[a, b]$, and $[L(f(x))]' = [f''(x) - q(x) f(x)]'$ is bounded and integrable in $[a, b]$, then $c_n = 0\left(\frac{1}{\lambda_n^2}\right)$ and the series /17a/ converges absolutely and uniformly to the limit $f'(x)$ in $[a, b]$.

2. If $f(x)$ satisfies the boundary conditions /2/, $q(x)$ is continuous and bounded in $[a, b]$, and $L[L(f(x))]$ is bounded and integrable in $[a, b]$, then $c_n = 0\left(\frac{1}{\lambda_n^2}\right)$ and the series /17b/ converges absolutely and uniformly to the limit $f''(x)$ in $[a, b]$.

APPLICATION OF THE OBTAINED RESULTS (c.f. [2])*)

Let us consider once more the problem /1/, /2/. We shall define the following sequence of functions:

$u_0(x)$ is an arbitrary, bouned function, satisfying the boundary conditions /2/, and such that $L(u_0(x))$ is bounded and integrable in $[a, b]$.

If the function $u_{k-1}(x)$ is already defined we choose $v_k(x)$ as the solution of the equation

$$\ddot{v}_k(x) - q(x)\, v_k(x) = -u_{k-1}(x) \qquad /24/$$

with the boundary conditions

$$v_k(a)\cos\alpha + \dot{v}_k(a)\sin\alpha = 0,$$
$$v_k(b)\cos\beta + \dot{v}_k(b)\sin\beta = 0,$$

and $u_k(x) = \alpha_k v_k(x)$, where α_k is some kind of a normalizing factor chosen to get all functions $u_k(x)$ bounded.

We can assume, without loss of generality, that all eigenvalues are positive and ordered

$$0 < \lambda_0 < \lambda_1 < \lambda_2 \ldots$$

Hence we have the following

Theorem

If the function $u(x)$ is chosen to be orthogonal to the eigenfunctions $\psi_0(x), \psi_1(x), \ldots, \psi_{p-1}(x)$ then

---

* A similar result, but for general selfadjoint system, has been obtained in a different way by P. Laasonen in [2]. The algorithm presented in [2] is of a slightly different form. Preparing this paper, P. Laasonen's results were unknown to the author.

$$u_k(x) = A_k\left[\psi_p(x) + r_{pk}(x)\right],$$

where A_k is a constant depending on normalizing factors $\alpha_1 \dots \alpha_k$ and

$$\left| r_{pk}(x) \right| \leq \left| \frac{\lambda_p}{\lambda_{p+1}} \right|^k \cdot M_p$$

where the constant M_p depends neither on $\alpha_1 \dots \alpha_k$ nor on k.

Proof

We have the Fourier expansion

$$u_k(x) = \sum_{n=0}^{\infty} c_n^{(k)} \psi_n(x); \qquad /25/$$

this series is converging absolutely and uniformly with its first and second derivatives, as for $k > 0$, $L\left[L(u_k)\right]$ is bounded and integrable in $\left[a, b\right]$. Using the equation /24/ we obtain

$$-\frac{1}{\alpha_k}\left[\sum_{n=0}^{\infty} c_n^{(k)} \ddot{\psi}_n(x) - q(x) \sum_{n=0}^{\infty} c_n^{(k)} \psi_n(x)\right] = \sum_{n=0}^{\infty} c_n^{(k-1)} \psi_n(x).$$

Applying /1/ we have the reourrence relation

$$c_n^{(k-1)} = \lambda_n \frac{c_n^{(k)}}{\alpha_k};$$

hence

$$c_n^{(k)} = \frac{\alpha_1 \cdots \alpha_k}{\lambda_n^k} c_n^{(0)}.$$

The orthogonality of $u_0(x)$ to $\psi_0(x), \dots, \psi_{p-1}(x)$ gives

$$c_0^{(o)} = c_1^{(o)} = \ldots\ldots = c_{p-1}^{(o)} = 0.$$

Thus the expansion /25/ takes the form

$$u_k(x) = \sum_{n=p}^{\infty} c_n^{(k)} \psi_n(x) = \sum_{n=p}^{\infty} \frac{\alpha_1 \cdots \alpha_k}{\lambda_n^k} c_n^{(o)} \psi_n(x) =$$

$$= \frac{\alpha_1 \cdots \alpha_k}{\lambda_p^k} c_p^{(o)} \left[\psi_p(x) + \sum_{n=p+1}^{\infty} \frac{c_n^{(o)}}{c_p^{(o)}} \left(\frac{\lambda_p}{\lambda_n} \right)^k \psi_n(x) \right]$$

If we denote $A_k = \frac{\alpha_1 \cdots \alpha_k}{\lambda_p^k}$ and $r_{pk}(x) = \sum_{n=p+1}^{\infty} \frac{c_n^{(o)}}{c_p^{(o)}} \left(\frac{\lambda_p}{\lambda_n} \right)^k \psi_n(x)$,

then, since the series is absolutely and uniformly converging, we have

$$|r_{pk}(x)| \leq \sum_{n=p+1}^{\infty} \left| \frac{c_n^{(o)}}{c_p^{(o)}} \right| \left(\frac{\lambda_p}{\lambda_n} \right)^k |\psi_n(x)| \leq \left| \frac{\lambda_p}{\lambda_{p+1}} \right|^k \sum_{n=p+1}^{\infty} \left| \frac{c_n^{(o)}}{c_p^{(o)}} \right| |\psi_n(x)| \leq \left| \frac{\lambda_p}{\lambda_{p+1}} \right|^k \cdot M_p$$

A slightly stronger result can be obtained in the following way.

Let us assume $c_n^{(o)} = O\left(\frac{1}{\lambda_n^m} \right)$. Then $c_n^{(o)} = \frac{\beta_n}{\lambda_n^m}$, where $\beta_n = O(1)$,

and we have

$$r_{pk}(x) = \sum_{n=p+1}^{\infty} \frac{\beta_n}{\beta_p} \left(\frac{\lambda_p}{\lambda_n} \right)^{k+m} \psi_n(x).$$

Hence

$$|r_{pk}(x)| \leq \left| \frac{\lambda_p}{\lambda_n} \right|^{k+m} \sum_{n=p+1}^{\infty} \left| \frac{\beta_n}{\beta_p} \right| |\psi_n(x)| \left| \frac{\lambda_{p+1}}{\lambda_n} \right|^{k+m}$$

Choosing the constant K_p so that

$$\left| \frac{\beta_n}{\beta_p} \right| |\psi_n(x)| \leq K_p$$

we have

$$\left| r_{pk}(x) \right| \leq \left| \frac{\lambda_p}{\lambda_{p+1}} \right|^{k+m} \cdot K_p \sum_{n=p+1}^{\infty} \left(\frac{\lambda_{p+1}}{\lambda_n} \right)^{k+m} = O\left(\left| \frac{\lambda_p}{\lambda_{p+1}} \right|^{k+m} \right).$$

The convergence of the series

$$\sum_{n=p+1}^{\infty} \left(\frac{\lambda_{p+1}}{\lambda_n} \right)^{k+m} \qquad \text{for} \quad k+m \geq 1$$

is easy to be verified.

Assuming that $u_0(x)$ satisfies the boundary conditions, and $q(x)$ is so regular that

$$L\left[L(u_0(x))\right]$$

is bounded and integrable in $[a, b]$, we get $m=2$.

Now it is impossible /in general/ to increase the value of m without assumptions concerning the boundary conditions for

$$L(u_0), \quad L\,L(u_0), \quad L\,L\,L(u_0), \quad \ldots$$

The eigenvalues satisfy the relation

$$\frac{u_k(x)}{v_{k+1}(x)} = \lambda_p \frac{\psi_p(x) + O\left(\left(\frac{\lambda_p}{\lambda_{p+1}} \right)^{k+m} \right)}{\psi_p(x) + O\left(\left(\frac{\lambda_p}{\lambda_{p+1}} \right)^{k+m+1} \right)}$$

References

1. TITCHMARSH E.C.: Eigenfunction Expansions Associated with Second-Order Differential Equations, London 1948:6-16.

2. LAASONEN P.: On the Simultaneous Determination of Several Eigensolutions of a Selfadjoint System of Differential Equations, M.T.A.C., 1959:13. 3-20.

Instytut Maszyn Matematycznych PAN
Algorytmy N° 2
© 1963.06

518.12

ON A CERTAIN CLASS OF ITERATION METHODS FOR NONLINEAR EQUATION

by Tomasz PIETRZYKOWSKI

Received November 1962

In [1] Hausholder and Bauer have presented a certain class of iterative methods for solving linear systems of equation. The purpose of this paper is to adopt these methods for nonlinear cases in real Hilbert spaces. As far as possible the same notation is used as in [1].

Let H be a real Hilbert space and G a positive selfadjoint linear operator on H. We shall define a G-norm $\| \ \|_G$ on H as follows

$$\| x \|_G = (Gx, x) \qquad \text{for } x \in H \qquad /1/$$

$\| x \|$ denotes the usual norm of $x \in H$ where $\| x \|^2 = (x, x)$.

Let $F : H \to H$ be a continuous nonlinear operator and x_0 the solution of the equation

$$F(x_0) = 0. \qquad /2/$$

Suppose that F has a continuous second derivate in a certain neighbourhood of x_0, and $F'(x_0)$ is invertible. Under these conditions we shall consider the following problem:

(N) find such a sequence $\{\bar{x}_\nu\}$ $(\nu = 1, 2, \ldots \ldots \bar{x}_\nu \in H)$ that

$$\lim_{\nu \to \infty} \bar{x}_\nu = x_0 \qquad /3/$$

and

$$\| \bar{x}_\nu - x_0 \|_G > \| \bar{x}_{\nu+1} - x_0 \|_G \qquad /4/$$

where G is a certain positive selfadjoint linear operator. Let

$$\bar{s}_\nu = x_o - \bar{x}_\nu \ , \quad \bar{r}_\nu = F\ (\bar{x}_\nu) \ , \quad A_\nu = F'\ (\bar{x}_\nu) \ . \qquad /5/$$

Suppose that the sequence $\{\bar{x}_\nu\}$ satisfying the conditions /3/ , /4/ of (N) is written in the form

$$\bar{x}_{\nu+1} = \bar{x}_\nu + \lambda_\nu \bar{y}_\nu \qquad (\nu = 1, 2, \ldots) \qquad /6/$$

where λ_ν is a real number and $\bar{y}_\nu$ a vector from H .

Moreover, let us assume the operator G , mentioned in /4/ , and the method of construction of vector $\bar{y}_\nu$ to be known. Hence we may describe how to find the parameter λ_ν .

From /1/ , /5/ and /6/ follows

$$\| s_{\nu+1} \|_G = \lambda_\nu^2 (G\, \bar{y}_\nu , \bar{y}_\nu) - 2\ \lambda_\nu (G\, \bar{s}_\nu , \bar{y}_\nu) + (G\, \bar{s}_\nu , \bar{s}_\nu).$$

Obviously the right hand side of the above formula is a quadratic function of λ_ν . Thus in view of /4/ and /5/ we choose the value of λ_ν to minimize this function.

$$\lambda_\nu = \frac{(G\ \bar{s}_\nu , \bar{y}_\nu)}{\| \bar{y}_\nu \|_G} \ . \qquad /7/$$

Let us now define the vector $\bar{y}_\nu$ and the operator G , so as to obtain the sequence $\{\bar{x}_\nu\}$ which satisfies the conditions /3/ and /4/ of the problem (N) .

The so-called linearization method will be used.

Let us consider the linear equation

$$Ax - h = 0 \qquad /8/$$

where $A = F'\ (x_o)$ and $h = Ax_o$.

Obviously x_o is the solution of this equation.

The problem of finding the sequence $\{x_\nu\}$ satisfying /3/ and /4/ will be called p r o b l e m (L).

Now let us define the sequences $\{r_\nu\}$ and $\{s_\nu\}$ similarly as the sequences $\{\bar{r}_\nu\}$ and $\{\bar{s}_\nu\}$.

Many methods of constructing the operator G and vector y_ν in the problem (L) are described in [1] .

For example

a. $G = I$ where I is the identity operator
$y_\nu = Ar_\nu$

b. $G = A$ /if A is a positive operator/
$y_\nu = r_\nu$

Further, only sequences $\{x_\nu\}$ satisfying the following conditions will be considered:

G is such a selfadjoint linear operator that $G = W(A)$, /9/ W being a continuous mapping of the space of linear operators into itself.

$y_\nu = B r_\nu$ where B is such a selfadjoint linear operator /10/ that $B = V(A)$, V being a continuous mapping of the space of linear operators into itself.

The sequence $\{x_\nu\}$ is said to be geometrically convergent if

$$\limsup_{\nu=\infty} \frac{\| s_{\nu+1} \|_G}{\| s_\nu \|_G} = \gamma < 1. \qquad /11/$$

The number γ will be called the convergence ratio.

Now, let us define the 'analogous' sequence which is the main concept of this paper.

Let $\{x_\nu\}$ be a sequence of the problem (L) , where λ_ν , y_ν and G are defined according to /7/ , /9/ and /10/. The sequence $\{\bar{x}_\nu\}$ is said to be analogous with $\{x_\nu\}$ if it is

obtained from formulae /6/, /7/, /9/, /10/ when $\bar{x}_\nu, \bar{r}_\nu, A_\nu$, $\bar{y}_\nu$, G_ν , B_ν replace x_ν , r_ν , A , y_ν , G, B , and $G_\nu = W(A_\nu)$, $B_\nu = V(A_\nu)$.

The properties of analogous sequences are described by the following theorem.

Theorem

Let $\{x_\nu\}$ be a sequence of the problem (L) , geometrically convergent to x_0 with the ratio δ.

Then the sequence $\{\bar{x}_\nu\}$ analogous with $\{x_\nu\}$ is the solution of the problem (N) geometrically convergent to x_0 with the same ratio δ.

Proof

Let us denote

$$\alpha_\nu = 1 - \frac{\|s_{\nu+1}\|_G^2}{\|s_\nu\|_G^2} \qquad /12/$$

and

$$\bar{\alpha}_\nu = 1 - \frac{\|s_{\nu+1}\|_G^2}{\|s_\nu\|_G^2} \qquad /13/$$

From /1/ , /6/ and /7/ we have

$$\alpha_\nu = \frac{(Gs_\nu , v_\nu)^2}{\|y_\nu\|_G^2 \quad \|s_\nu\|_G^2} \qquad /14/$$

To simplify the proof let us suppose that $x_\nu = \bar{x}_\nu$. Now, we shall estimate the difference between α_ν and $\bar{\alpha}_\nu$.

From formula /4/ and assuming the existence of the second

derivative of F it follows

$$\|A_\nu - A\| \underset{0}{\|s_\nu\|} 0 \quad \text{and} \quad \|\bar{r}_\nu - r_\nu\| \underset{1}{\|s_\nu\|} 0 \ *$$

Since the sequence $\{x_\nu\}$ satisfies the conditions /9/ and /10/ it can be shown that

$$\Delta G_\nu \underset{0}{\|s_\nu\|^2} 0 \quad \text{and} \quad \Delta y_\nu \underset{1}{\|s_\nu\|} 0 \qquad /15/$$

where $\Delta G_\nu = G_\nu - G$ and $\Delta y_\nu = \bar{y}_\nu - y_\nu$.

Since $\lim_{\nu\to\infty} s_\nu = 0$ and according to /15/ G_ν and $G_{\nu+1}$ are positive for sufficiently great ν hence they determine norms in the space H . Thus we can denote

$$\bar{\alpha}'_\nu = 1 - \frac{\|\bar{s}_{\nu+1}\|^2_{G_{\nu+1}}}{\|\bar{s}_\nu\|^2_G} \qquad /16/$$

and analogously to /14/ we obtain

$$\bar{\alpha}'_\nu = \frac{(G_{\nu+1}\bar{s}_\nu, \bar{y}_\nu)^2}{\|y_\nu\|^2 G_\nu \|\bar{s}_\nu\|^2_G} . \qquad /17/$$

Hence, in view of /14/ and /16/ , after certain transformations we have

$$\bar{\alpha}'_\nu - \alpha_\nu = \frac{1 + \frac{1}{(Gs_\nu, y_\nu)}\left((\Delta G_{\nu+1}\bar{s}_\nu, y_\nu) + (G\Delta s_\nu, y_\nu) + (G\bar{s}_\nu, \Delta y_\nu)\right)}{\left(1 + \frac{(\Delta G_\nu y_\nu, y_\nu) + 2(G\Delta y_\nu, \bar{y}_\nu)}{(Gy_\nu, y_\nu)}\right)\left(1 + \frac{(\Delta G_\nu \bar{s}_\nu, s_\nu) + 2(G\bar{s}_\nu, \Delta s_\nu)}{(Gs_\nu, s_\nu)}\right)} \qquad /18/$$

---

*Let f be a continuous function of real variable t and n is natural number 0 /zero/ then the formula $f \underset{n}{=} 0$ denotes that $\lim_{t\to 0} \frac{f(t)}{t^n} = 0$.

Since $\delta < 1$, using /11/ and /12/ , we can prove the existence of such $\theta > 0$ that for a sufficiently great ν we have $\alpha_\nu > \theta$. Thus from /15/ and /18/ we have

$$\bar{\alpha}'_\nu - \alpha_\nu \underset{0}{=\!=} \|s_\nu\| \; 0 \quad . \qquad /19/$$

According to /13/ , /15/ and /16/ it can be easily seen that

$$\bar{\alpha}'_\nu - \bar{\alpha}_\nu \underset{0}{=\!=} \|s_\nu\| \; 0 \quad . \qquad /20/$$

Finally we conclude that for a sufficiently great

$$\bar{\alpha}_\nu - \alpha_\nu \underset{0}{=\!=} \|s_\nu\| \; 0 \quad . \qquad /21/$$

Since $\lim_{\nu\to\infty} s_\nu = 0$

$$\lim_{\nu\to\infty} \inf \bar{\alpha}_\nu = \lim_{\nu\to\infty} \inf \alpha_\nu$$

and from /11/ , /12/ and /13/ we have

$$\lim_{\nu\to\infty} \sup \frac{\|\bar{s}_{\nu+1}\|_G}{\|\bar{s}_\nu\|_G} = \theta$$

that completes the proof.

Due to the above theorem the methods used in linear problems can also be used in nonlinear cases.

As an example we shall present formulae for nonlinear cases obtained by the method /a/ and /b/ .

/ā/ $$\bar{x}_{\nu+1} = \bar{x}_\nu - \frac{(\bar{r}_\nu , \bar{r}_\nu)}{(A_\nu \bar{r}_\nu , A_\nu \bar{r}_\nu)} A_\nu \bar{r}_\nu$$

$$/\bar{b}/\quad \bar{x}_{\nu+1} = \bar{x}_\nu - \frac{(\bar{r}_\nu\ ,\ \bar{r}_\nu)}{(A_\nu \bar{r}_\nu\ ,\ \bar{r}_\nu)}\ \bar{r}_\nu$$

/if A is a positive operator/.

The ratio of convergence of the above sequences is defined like in linear cases /a/ and /b/ .

Finally let us note that the method /$\bar{b}$/ can be applied to maximize such functional f defined on H , and that

- f" exists
- $f''(x_0)$ is invertible.

In this case x_0 is a 'vector of maximum' .

It is well known that the above problem is equivalent to the solution of the equation

$$f'(x) = 0 .$$

Applying formula /$\bar{b}$/ for $(\nu + 1)$ -th term of the sequence $\{\bar{x}_\nu\}$ we obtain

$$\bar{x}_{\nu+1} = \bar{x}_\nu - \frac{\left(f'(\bar{x}_\nu)\ ,\ f'(\bar{x}_\nu)\right)}{\left(f''(\bar{x}_\nu)\ f'(\bar{x}_\nu)\ ,\ f'(\bar{x}_\nu)\right)}\ f'(\bar{x}_\nu) .$$

References

1. HOUSHOLDER A.S., BAUER F.L.: On Certain Iterative Methods for Solving Linear Systems, Numerische Mathematik, 1960:2, 55-59.

2. LUSTERNIK L.A., SOBOLEV W.J.: Elementy funkcionalnogo analiza, Moskva 1951.

STATISTICAL AND OTHER APPLICATIONS

Instytut Maszyn Matematycznych PAN
Algorytmy N° 2

519.21
518.6
681.14:523.8

NIEKTÓRE METODY GENEROWANIA REALIZACJI PROCESU POISSONA

Elżbieta PLESZCZYŃSKA

Pracę złożono 13.11.1962 r.

W pierwszej części pracy zdefiniowano generator liczb losowych o zadanym rozkładzie i generator realizacji procesu stochastycznego. Podano sposób budowy generatora realizacji jednorodnego procesu Poissona przy pomocy generatora liczb losowych o rozkładzie wykładniczym z parametrem 1, nazwanego generatorem W.

W drugiej części pracy opisano i porównano metody konstrukcji generatora W.

1. Generator liczb losowych

Statystycy posługują się często w swojej pracy tzw. tablicami liczb losowych. Tablica liczb losowych o danym rozkładzie ma stanowić próbkę prostą z populacji generalnej o tym rozkładzie. Aby sprawdzić, czy podany w tablicy ciąg liczb można istotnie uważać za taką próbkę, autor tablicy opisuje zwykle na wstępie testy statystyczne, którym te liczby poddano, oraz zestawia otrzymane wyniki. Zespół testów jest wybierany przez autora tablicy według jego uznania i zgodnie z aktualnymi możliwościami ich przeprowadzenia. Możliwości te wzrosły dzięki wprowadzeniu maszyn cyfrowych. Obecnie wydawane tablice liczb losowych są przeważnie sporządzane i sprawdzane testami na maszynach cyfrowych za pomocą tzw. generatorów liczb losowych.

Generator liczb losowych o danym rozkładzie w maszynie cyfrowej jest to program, po odwołaniu się do którego maszyna oblicza i re-

jestruje w pamięci jako wynik liczbę, należącą do zbioru wartości zmiennej losowej o żądanym rozkładzie. Po n-krotnym odwołaniu się do tego programu otrzymuje się ciąg liczb, który ma stanowić n-elementową próbkę prostą z populacji generalnej o tym rozkładzie. Hipotezę tę, dla obranego dostatecznie dużego n, należy sprawdzić zespołem testów zaprojektowanych przez twórcę generatora.

Tak sprawdzony generator liczb losowych z jednej strony może służyć do wydrukowania tablicy liczb losowych, z drugiej strony zastępuje tablice liczb losowych we wszelkich obliczeniach na maszynie cyfrowej, w których używa się liczb losowych. Metody obliczeniowe posługujące się liczbami losowymi noszą nazwę metod Monte-Carlo.

2. Generator realizacji procesu Poissona

Określenie jednorodnego procesu Poissona*) : skokowy proces stochastyczny $(X_t,\ 0 \leqslant t < \alpha)$ o przyrostach niezależnych, o stanach $i = 0, 1, 2, \ldots$, jest jednorodnym procesem Poissona, jeśli dla dowolnego punktu $t\ (0 \leqslant t < \infty)$ zachodzi relacja

$$P(X_t = i) = \frac{(\lambda t)^i}{i!} e^{-\lambda t} ,$$

gdzie $\lambda > 0$.

Definicję niejednorodnego procesu można znaleźć w [4] .

Podobnie jak w przypadku generatora liczb losowych wynikiem jest jedna liczba ze zbioru wartości zmiennej losowej, w przypadku generatora realizacji procesu stochastycznego wynikiem powinna być jedna funkcja ze zbioru realizacji tego procesu dla podanego skończonego przedziału parametru procesu. Powstaje problem przybliżania tej funkcji w maszynie cyfrowej. Jeśli realizacja procesu

*) [1], str. 242.

jest linią schodkową o skończonej ilości i wielkości skoków w każdym skończonym przedziale, jak to ma miejsce dla procesu Poissona, to może być reprezentowana w maszynie przez pary liczb (t_i, X_{t_i}) dla $i = 1, 2, \ldots$, gdzie

t_i - wartości parametru, przy których następuje skok,

X_{t_i} - wartości procesu X_t dla $t = t_i$.

Dla procesu Poissona oznaczymy przez t_i wartość parametru, przy której następuje i-ty skok, i wówczas $X_{t_i} = i$. Należy jeszcze podać sposób obliczenia wartości t_i. Można je łatwo wyznaczyć w jednorodnym procesie Poissona.

3. Budowa generatora realizacji jednorodnego procesu Poissona

Wprowadźmy oznaczenia:

λ - intensywność procesu,

L - zmienna losowa, będąca odległością dwóch kolejnych sygnałów procesu; l - wartość L.

Wiadomo, że odległość dwu kolejnych sygnałów L w procesie jednorodnym o przyrostach niezależnych przy danej intensywności λ ma rozkład wykładniczy o gęstości

$$f(l) = \begin{cases} \lambda e^{-\lambda l} & l \geq 0 \\ 0 & \text{poza tym.} \end{cases}$$

Nazwijmy generator liczb losowych o rozkładzie wykładniczym przy danym λ generatorem L, a jego szczególny przypadek dla $\lambda = 1$ generatorem W /W - zmienna losowa o rozkładzie wykładniczym dla $\lambda = 1$/. Wobec związku $L = \frac{W}{\lambda}$, praca generatora L polega na odwołaniu się do generatora W i podzieleniu otrzymanej liczby losowej przez λ.

Budujemy realizację jednorodnego procesu Poissona na odcinku $(0, T)$. Niech $l_1, l_2, \ldots$ oznaczają kolejne liczby z generatora L.

Mamy wtedy

$$t_i = \sum_{k=1}^{i} l_k$$

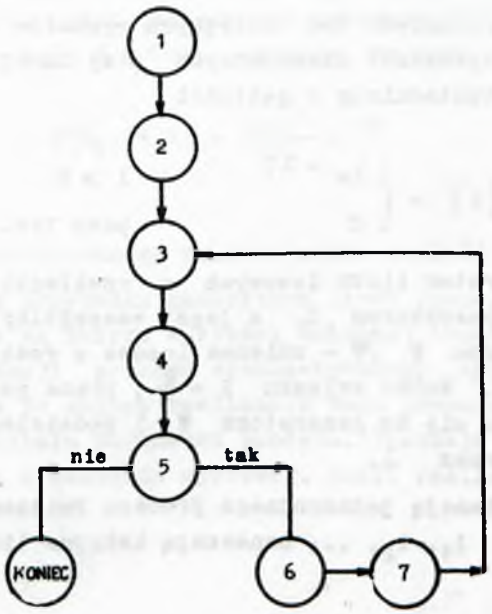
$$X_{t_i} = i$$

dla tych i, przy których $t_i < T$.

Bez trudności można także generować realizacje złożonego jednorodnego procesu Poissona /jest to proces jednorodny o przyrostach niezależnych z intensywnością sygnałów λ, przy czym wielkość sygnału jest zmienną losową o zadanym rozkładzie, a przez X_t oznaczamy sumę wielkości sygnału w czasie od 0 do t/. Należy wtedy zbudować generator X liczb losowych o rozkładzie takim, jak zadany rozkład wielkości sygnału. Niech x_1, x_2, ... będą kolejnymi liczbami z tego generatora. Wówczas

$$X_{t_i} = \sum_{k=1}^{i} x_k.$$

Ogólny schemat generatora realizacji jednorodnego procesu Poissona jest więc następujący:



(1) - Podajemy wartości λ i T, konieczne przy odwoływaniu się do generatora realizacji procesu X_t.

(2) - Nadajemy t i X_t wartość początkową równą zero.

(3) - Odwołujemy się do generatora W. Obliczamy odległość dwu kolejnych sygnałów $l = \frac{W}{\lambda}$, gdzie W jest liczbą otrzymaną z generatora W.

(4) - Zwiększamy wartość parametru t o liczbę l.

(5) - Badamy, czy $t < T$, a więc czy działanie generatora ma trwać dalej.

(6) - Odwołujemy się do generatora X. Zwiększamy wartość X_t o liczbę x otrzymaną z generatora X. Jeśli X_t oznacza ilość sygnałów, $x = 1$.

(7) - Rejestrujemy t i X_t zależnie od potrzeb, gdyż często interesuje nas tylko ostateczna wartość X_T, a wyników pośrednich nie trzeba przechowywać w pamięci maszyny cyfrowej.

4. Sprawdzanie generatora realizacji procesu X_t

Już dla generatora liczb losowych istnieje brak pewnych i jednolitych metod sprawdzania, czy generowane przezeń liczby są istotnie próbką prostą z populacji generalnej o zadanym rozkładzie. Dla generatora realizacji procesu stochastycznego trudności znacznie wzrastają. W przypadku jednorodnego procesu Poissona sprawa jest jednak stosunkowo prosta. Z ogólnego schematu budowy generatora realizacji tego procesu podanego w p.3 widać, że jeśli generator W pracuje dobrze, pozostałe operacje są tak proste, że zapewne całość również pracuje dobrze. W dalszym ciągu będzie mowa o budowie i sprawdzaniu generatora W. Warto jeszcze zauważyć, że liczby X_T są wartościami zmiennej losowej o rozkładzie Poissona z parametrem λT, a zatem schemat z p.3 może służyć jako generator liczb losowych o rozkładzie Poissona. Sprawdzenie tego generatora można by więc wykorzystać do weryfikacji generatora realizacji procesu Poissona.

5. Budowa generatora W

Opiszemy 3 różne sposoby generowania.

Sposób I. Dystrybuanta zmiennej losowej W ma postać

$$F(W) = \begin{cases} 0 & W \leq 0 \\ 1 - e^{-W} & W > 0. \end{cases}$$

Zatem, na podstawie twierdzenia, że jeśli F(x) jest dystrybuantą dowolnej zmiennej losowej X, to zmienna losowa R = F(x) ma rozkład jednostajny na odcinku $(0, 1)$*), liczba losowa z generatora W ma postać

$$w = -\ln r \ ,$$

gdzie r – liczba losowa z generatora R produkującego liczby losowe o rozkładzie jednostajnym na odcinku $(0, 1)$.

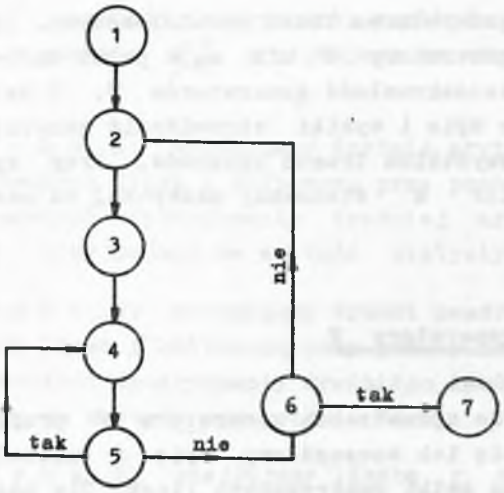
Sposób II. Zmienną losową W można przedstawić jako połowę sumy kwadratów dwóch zmiennych losowych o rozkładzie normalnym ze średnią zero i odchyleniem średnim 1**).

Za wartość zmiennej losowej normalnej o rozkładzie N $(0, 1)$, opierając się na centralnym twierdzeniu granicznym, bierzemy połowę sumy 12 wartości zmiennej losowej o rozkładzie równomiernym na odcinku $(-1, 1)$.

Sposób III. Sposób ten i jego uzasadnienie opisano dokładnie w [3]***). W sposobie tym generowanie liczby w odbywa się według następującego schematu:

---

*[2], str. 43.
**[1], str. 286.
***[3], str. 2-3.



(1) - Nadajemy parametrowi j wartość początkową równą - 1.

(2) - Zwiększamy wartość parametru j o 1.

(3) - Odwołujemy się do generatora R /wspomnianego w opisie sposobu I/; otrzymaną liczbę r oznaczamy $R_o^{(j)}$. Nadajemy parametrowi i wartość początkową równą zero.

(4) - Zwiększamy wartość parametru i o 1. Odwołujemy się do generatora R; otrzymaną liczbę r oznaczamy $R_i^{(j)}$.

(5) - Badamy, czy $R_{i-1}^{(j)} > R_i^{(j)}$.

(6) - Badamy, czy i nieparzyste.

(7) - Obliczamy $w = j + R_o^{(j)}$.

Badane są więc serie wyrazów malejących w ciągu kolejnych liczb z generatora R. Jeśli ilość wyrazów tworzących serię jest nieparzysta, przyjmuje się numer kolejno badanej serii za część całkowitą liczby w, a pierwszą liczbę serii - za część ułamkową liczby w; w przeciwnym razie bada się następną serię. Serie numerowane są od zera.

We wszystkich trzech sposobach korzysta się z generatora R,

zwanego zwykle generatorem liczb pseudolosowych. Jak wiadomo, stosowane obecnie generatory R nie są w pełni zadawalające, co pociąga za sobą niedoskonałość generatorów W. W dalszym ciągu pracy, p.6, podano opis i wyniki sprawdzania generatorów W, zbudowanych według wszystkich trzech sposobów. Przy sprawdzaniu tym używano generator R stosowany zazwyczaj na maszynie cyfrowej ZAM-2.

6. Sprawdzanie generatora W

Układ testów do sprawdzania generatora W przyjęto wg [3], gdzie znajduje się ich szczegółowy opis i uzasadnienie wyboru. Badane są kolejne setki generowanych liczb. Dla każdej setki oblicza się wartości pięciu statystyk, zdefiniowanych następująco:

a. s t a t y s t y k a F: znajdujemy, ile spośród stu generowanych liczb należy do każdego z dziewięciu równie prawdopodobnych przedziałów, na które dzieli się zbiór wartości zmiennej losowej W; obliczamy wartość χ^2, oznaczamy ją przez χ^2_F i odczytaną z tablic rozkładu χ^2 wartość dystrybuanty χ^2 z 8 stopniami swobody w punkcie χ^2_F przyjmujemy jako wartość statystyki F.

b. s t a t y s t y k a C: dzielimy zbiór wartości zmiennej losowej W na trzy równie prawdopodobne przedziały I_1, I_2, I_3 i znajdujemy w 99 parach kolejnych liczb danej setki ilość par a_{jk}, w których pierwsza liczba należy do I_j, druga do I_k $(j, k = 1, 2, 3)$.

Znajdujemy $\chi^2_C = \sum_j \sum_k \frac{(a_{jk} - 99\hat{p}_j\hat{p}_k)^2}{99\hat{p}_j\hat{p}_k}$,

gdzie $p_k = \frac{\sum_j a_{jk}}{99}$ dla $k = 1, 2, 3$ i odczytaną z tablic

rozkładu χ^2 wartość dystrybuanty rozkładu χ^2 z 4 stopniami swobody w punkcie χ^2_C przyjmujemy za wartość statystyki C.

c. s t a t y s t y k a M: wyznaczamy średnią arytmetyczną $\bar{x}$ w setce generowanych liczb i obliczoną przy pomocy rozwinięcia Edgeworth'a wartość dystrybuanty średniej arytmetycznej w punkcie $\bar{x}$ przyjmujemy za wartość statystyki M.

d. s t a t y s t y k a V: wyznaczamy średni kwadrat S^2 w setce generowanych liczb i obliczoną przy pomocy rozwinięcia Edgeworth'a wartość dystrybuanty średniego kwadratu w punkcie S^2 przyjmujemy za wartość statystyki V.

e. s t a t y s t y k a R: znajdujemy liczbę r serii wyrazów mniejszych i większych od mediany w setce generowanych liczb i odczytaną z tablicy 1 w [3] wartość dystrybuanty liczby serii w punkcie r przyjmujemy za wartość statystyki R.

W tabelce 1 zestawiono wartości tych statystyk dla początkowych sześciu setek otrzymanych z trzech generatorów W /według I, II i III sposobu generacji/.

Z sześciu wartości trudno odczytać, czy istotnie zmienne losowe F, C, M, V i R mają rozkład jednostajny na odcinku $(0,1)$*). Jeszcze trudniej powiedzieć, który z trzech rozkładów jest 'bardziej jednostajny' - innymi słowy, nie ma metody wyboru pomiędzy tymi trzema generatorami.

Sprawdzamy po prostu oddzielnie hipotezy o rozkładzie, niezależności kolejnych liczb, średniej, średnim kwadracie i seriach liczb większych i mniejszych od mediany w sześciu setkach liczb dla każdego generatora z osobna. W tym celu w tabelce 1 podano także wartości średnich arytmetycznych dla zmiennych losowych F, C, M, V i R. Teoretyczny rozkład tych średnich można dobrze przybliżyć przez rozkład normalny z parametrami 0,5 i 0,12.

*patrz twierdzenie sformułowane w p. 5.

Tabelka 1

Wartości statystyk P, C, M, V i R w trzech różnych generatorach W

Nazwa statystyki	Generator I							Generator II							Generator III						
	numer setki						średnia arytm.	numer setki						średnia arytm.	numer setki						średnia arytm.
	1	2	3	4	5	6		1	2	3	4	5	6		1	2	3	4	5	6	
P	.56	.73	.64	.41	.22	.17	.46	.10	.08	.51	.47	.98	.76	.48	.75	.32	.96	.73	.32	.32	.56
C	.62	.41	.08	.42	.22	.20	.33	.52	.53	.87	.60	.14	.41	.51	.43	.06	.70	.59	.61	.80	.51
M	.87	.90	.67	.92	.71	.83	.82	.19	.50	.87	.28	.99	.05	.48	.93	.50	.49	.13	.41	.17	.44
V	.85	.74	.86	1.0	.72	.76	.82	.23	.52	.83	.28	.94	.08	.48	.84	.65	.18	.27	.48	.29	.45
R	.62	.13	.18	.07	.38	.54	.32	.90	.81	.54	.18	.54	.76	.62	.10	.07	.76	.76	.18	.18	.34

Jeśli zatem hipoteza o jednostajnym rozkładzie jest prawdziwa, to prawdopodobieństwo otrzymania wartości spoza przedziału (0,26; 0,74) wynosi około 0,05. Ponieważ dla generatora I średnie zmiennych losowych M i V wynoszą 0,82, uznajemy te wyniki za mało prawdopodobne i odrzucamy hipotezę o jednostajnym rozkładzie M i V, a co za tym idzie, dyskwalifikujemy generator I /przynajmniej do czasu przeprowadzenia liczniejszych badań/. Nie ma natomiast podstaw do dyskwalifikacji generatorów II i III. Za używaniem generatora III przemawia znacznie krótszy czas generowania.

Ze względu na wielkie obciążenie maszyny ZAM-2 nie dało się sprawdzić generatorów W w szerszym zakresie. W przyszłości powinno się jednak obliczyć wartości statystyk F, C, M, V i R dla znacznie większej liczby kolejnych setek liczb z wybranego generatora W, aby ułatwić planowanie doświadczeń Monte-Carlo przyszłym użytkownikom generatora. W tablicach losowych liczb wykładniczych [3] wartości tych statystyk są wydrukowane na początku każdej setki liczb, co ma służyć do losowania warstwowego /'stratified sampling'/, pozwalającego na większą efektywność analizy Monte-Carlo.

W doświadczeniach przeprowadzanych na maszynie cyfrowej eksperymentator powinien mieć możność decyzji, czy będzie korzystał z kolejnych setek, czy też wybierze niektóre z nich, zgodnie z ideą losowania warstwowego.

7. Zastosowanie generatora realizacji procesu Poissona

W badaniach operacyjnych częste są modele, w których 'na wejściu' znajduje się proces Poissona; w szczególności są to różne modele teorii kolejek. Klasycznym przykładem są zgłoszenia do centrali telefonicznej. Niejednokrotnie proces Poissona dobrze przybliża różne procesy w naukach przyrodniczych. Uruchomienie generatora realizacji procesu Poissona na maszynie cyfrowej pozwala

przeprowadzać na niej całe doświadczenia. Generator ten zastosowano z dobrym skutkiem przy badaniu modelu kolejek ze sprzężeniem zwrotnym /opis w [5] /.

Literatura

1. FISZ M.: Rachunek prawdopodobieństwa i statystyka matematyczna, PWN 1958.
2. BUSLENKO N.P., ŠREJDER J.A.: Metod statističeskich ispytanij, Moskwa 1961.
3. CLARK Ch.E., HOLZ B.W.: Exponentially Distributed Random Numbers, Published for Operations Research Office, 1960.
4. RENYI A.: On Some Problems Concerning Poisson Processes, Publications Mathematicae /Debrecen/ 1951:2, 66-73.
5. PLESZCZYŃSKA E.: O modelowaniu pewnego procesu stochastycznego na maszynie cyfrowej. Wysłane do Zeszytów Problemowych Nauki Polskiej.

SOME METHODS OF THE POISSON PROCESS GENERATION

Summary

The paper describes the way of generating the Poisson's process used on the ZAM-2 computer. The so-called generation process W of pseudo-random numbers from the eksponential distribution /the probability density being e^{-x}, $0 \leq x < \infty$/ is the integral part of the described process. Three various ways of designing the process W are given, their randomness being checked.

Some examples are given of the Poisson process generation used for Operation Research.

THEORY OF PROGRAMMING

Instytut Maszyn Matematycznych
Algorytmy N° 2

518.6

O PEWNEJ METODZIE ADRESACJI ZASTOSOWANEJ PRZY BUDOWIE SYSTEMU SAKO-SAS

Jacek MOSZCZYŃSKI
Andrzej WIŚNIEWSKI

Pracę złożono 13.12.1962 r.

Praca podaje opis metody przejścia od wewnętrznej postaci języka symbolicznego /tzw. języka WJS/ do wewnętrznego kodu rozkazowego maszyny. Omawiana metoda, zwana adresacją, zastosowana jest na maszynach cyfrowych XYZ i ZAM-2 przy budowie kompilatora SAKO i translatora SAS, może jednak z łatwością być adaptowana do innych języków i dla innych typów maszyn.

1. U w a g i w s t ę p n e

Artykuł jest przeznaczony dla osób znających dobrze programowanie w językach SAKO i SAS, pragnących zapoznać się z kolei z niektórymi problemami związanymi z przejściem od ww. języków do wewnętrznego kodu maszyny.

Będzie tu omówiona metoda zwana adresacją, zastosowana na maszynach XYZ i ZAM-2 przy budowie kompilatora SAKO i translatora SAS . Adresacja polega na przejściu od Wewnętrznego Języka Symbolicznego, będącego wynikiem pracy SAKO i SAS, do Wewnętrznego Kodu Maszyny. Od SAKO do Wewnętrznego Języka Symbolicznego dochodzimy w ten sposób, że każdemu zdaniu SAKO odpowiada sekwencja instrukcji Wewnętrznego Języka Symbolicznego.

W SAS każdemu rozkazowi lub dyrektywie odpowiada jedna instrukcja w Wewnętrznym Języku Symbolicznym.

2. Terminologia i oznaczenia

Poniżej zamieszczamy tablicę ze znakami kodu Ferrantiego i z odpowiadającymi im wartościami liczbowymi w maszynach XYZ i ZAM-2.

Tablica 1

ZNAKI DALEKOPISOWE DLA WEJŚCIA I WYJŚCIA NA TAŚMIE DZIURKOWANEJ

Cyfry	Litery	Wartość dziesiętna
C y f r y		0
1	A	1
2	B	2
*	C	3
4	D	4
(	E	5
)	F	6
7	G	7
8	H	8
≡	I	9
=	J	10
-	K	11
ν	L	12
LINIA	M	13
SP (spacja)	N	14
,	O	15
0	P	16
>	Q	17
:	R	18
3	S	19
→	T	20
5	U	21
6	V	22
/	W	23
x	X	24
9	Y	25
+	Z	26
L i t e r y		27
.	.	28
≠	?	29
P.K.	£	30
≠/BŁĄD/	≠/BŁĄD/	31

Uwaga: Wewnątrz m.c. wartość dziesiętna litery jest zwiększoną o 32 wartością dziesiętną dla cyfry.

Dla wygody czytelnika zamiast wartości liczbowej danego znaku, będziemy pisali ten znak zamknięty w znaki dosłowności $\ulcorner i \urcorner$. Tak więc zamiast 19 piszemy $\ulcorner 3 \urcorner$, zamiast 33 piszemy $\ulcorner A \urcorner$. Wszystkie programy jak i formuły występujące w artykule są zapisane według języka SAKO i jedynymi odstępstwami od tych reguł będzie używanie znaków dosłowności, małych liter, a także zmiennych międzyparagrafowych.

Poniższy zapis

$$A \underset{\text{def}}{=} 0,\ 1,\ \ulcorner C \urcorner$$

oznacza, że symbol A z definicji przyjmuje wartości 0, 1 lub $\ulcorner C \urcorner$.

Natomiast

$$A \underset{\text{def}}{=} 1,\ 2,\ \ldots\ldots,\ 10$$

oznacza, że A przyjmuje kolejne wartości liczb naturalnych począwszy od pierwszej, skończywszy na ostatniej włącznie. W naszym przypadku pierwszą wartością jest 1, ostatnią 10. Niech $Z_o, Z_1, \ldots Z_n$ oznacza dowolny ciąg znaków kodu Ferrantiego. Wówczas przez zapis $\ulcorner Z_o\ Z_1 \ldots\ldots Z_n \urcorner$ rozumie się

$$\left(\ldots\ldots, \left(\left(\ulcorner Z_o \urcorner * 6\right) + \ulcorner Z_1 \urcorner\right) * 6 + \ldots\right) * 6 + \ulcorner Zn \urcorner .$$

Przykład:

$$\ulcorner A * B \urcorner \equiv \left(\left(\ulcorner A \urcorner * 6\right) + \ulcorner * \urcorner\right) * 6 + \ulcorner B \urcorner .$$

W dalszym ciągu przez L_i będziemy oznaczali dowolną literę kodu Ferrantiego, a przez C_i dowolną cyfrę tego kodu; symbol ϕ oznacza binarne zero.

3. Wewnętrzny Język Symboliczny

Zamiast pisać Wewnętrzny Język Symboliczny używać będziemy skrótu WJS.

3.1. Krótka charakterystyka WJS .

Program w WJS , otrzymany w wyniku kompilacji SAKO lub translacji SAS , składa się z

- instrukcji i dyrektyw,
- liczników: LR /Licznik Rozkazów/
  LMR /Licznik Miejsc Roboczych/,
- list.

3.1.1. Instrukcje i dyrektywy

Każda instrukcja lub dyrektywa WJS jest zapisana w jednym lub kilku 36-bitowych słowach maszyny.

Słowo WJS definiujemy:

$$\text{SŁOWO} \equiv ((((Zn * 5 + OP) * 1 + Mod) * 1 + ZA) * 1 + Dod) * 3 + Puste) * 24 + AS,$$

gdzie $Zn \stackrel{def}{=} 0, 1$

$OP \stackrel{def}{=} 0, 1, \ldots, 31$

$Mod \stackrel{def}{=} 0, 1$

$ZA \stackrel{def}{=} 0, 1$

$Dod \stackrel{def}{=} 0, 1$

$Puste \stackrel{def}{=} \lceil\phi\rceil$

$AS \stackrel{def}{=} Z_0\ Z_1\ Z_2\ Z_3$.

W ostatniej definicji

Z_o jest znakiem $\neq \phi$

Z_i - dowolne.

Objaśnienia oznaczeń:

Zn Znak instrukcji
OP Numer operacji w wewnętrznym kodzie maszyny
Mod Numer modyfikacji
ZA Znak adresu
Dod rozróżnienie: 0 - instrukcja
1 - dyrektywa
AS Adres symboliczny instrukcji lub dyrektywy.

W procesie Adresacji instrukcje WJS są zamieniane na rozkazy w wewnętrznym kodzie maszyny. Dyrektywy zaś dostarczają dodatkowych informacji programowi adresującemu i nie mają odpowiedników w wewnętrznym kodzie maszyny.

3.1.2. Liczniki

Licznik rozkazów /LR/ służy do rozmieszczania rozkazów adresowanego rozdziału w pamięci operacyjnej.

Licznik miejsc roboczych /LMR/ służy do rezerwowania miejsc dla Zmiennych, na których działa dany rozdział adresowanego programu.

3.1.3. Listy

3.1.3.1. Lista Rozdziałów

Lista rozdziałów jest takim układem elementów, że w każdym elemencie zawarta jest informacja o elemencie następnym, zaś w elemencie końcowym informacja o elemencie początkowym /jest to łańcuch zamknięty w pętlę/. Służy ona do rozmieszczenia zaadresowanych rozdziałów w pamięci pomocniczej maszyny.

3.1.3.2. Lista Adresów

Elementy listy adresów zawierają informacje o danym adresie symbolicznym i odpowiadającym mu adresie rzeczywistym, dotyczącym pamięci operacyjnej maszyny. Listę adresów nazywamy inaczej /słownikiem symboli /Label Address Book/.

Dzięki opisanej w punkcie 3.1. budowie programu WJS , każda jego instrukcja może być zaadresowana niezależnie i kolejno bez względu na kontekst, co pozwala w dużym stopniu uprościć proces adresacji.

3.2. Przebieg kompilacji SAKO i translacji SAS .

Proces kompilacji programu napisanego w SAKO lub SAS przebiega w dwu etapach. Etapem pierwszym zajmujemy się w punkcie 3.2. Drugim etapem jest adresacja omawiana w punkcie 4.

3.2.1. WJS jako rezultat kompilacji SAKO .

Deklaracje języka SAKO są zastępowane dyrektywami WJS , zaś symbole na liście deklaracji są listowane. Na listach tych działa tylko kompilator SAKO w czasie wypełniania szkieletów sekwencji /patrz niżej/. Istnieją w SAKO deklaracje nie tłumaczone na dyrektywy, lecz na ciąg instrukcji WJS /STRUKTURA ( ) :/, oraz takie, których symbole są tylko listowane /CALKOWITE :/. Pozostałe zdania SAKO są zastępowane sekwencjami instrukcji WJS . Proces ten przebiega następująco:

- Każdemu zwrotowi lub działaniu odpowiada szkielet sekwencji, w który są wpisane tylko części operacyjne.
- Symbole na liście zdania lub argumenty działań są wpisywane do szkieletów jako adresy symboliczne.

Zasady tworzenia adresów symbolicznych są następujące:

- Jeżeli symbolem jest zmienna, to

$$AS = \ulcorner L_0\ Z_1\ Z_2\ Z_3 \urcorner .$$

- Jeżeli symbolem jest nazwa podprogramu, to

$$AS = \ulcorner * L_1\ Z_2\ Z_3 \urcorner .$$

- Jeżeli symbolem jest etykieta, to

$$AS = \ulcorner C_0\ Z_1\ Z_2\ Z_3 \urcorner .$$

3.2.2. WJS jako rezultat translacji SAS .

Postacią zewnętrzną WJS jest SAS . Wszystkie informacje dotyczące budowy instrukcji WJS są podane w rozkazie SAS explicite.

Aby język SAS mógł być użyteczny w programowaniu, dokonano w WJS szeregu rozszerzeń.

Czynności pierwszego etapu translacji programu napisanego w SAS są następujące:

- Przyporządkowanie danej dwuliterowej nazwie operacji odpowiadającego jej numeru rozkazu w wewnętrznym kodzie maszyny i wpisanie go w pozycję OP .
- Zasady tworzenia AS są analogiczne jak w p. 3.2.1, z tym, że nazwie podprogramu odpowiada w SAS nazwa paragrafu.

Jeżeli adresem rozkazu SAS jest liczba całkowita bez znaku, niech:

$$AC_{d\bar{e}f}\left(\left(\left(\ulcorner C_0' \urcorner \times 10 + \ulcorner C_1' \urcorner\right) \times 10 + \ulcorner C_2' \urcorner\right) \times 10 + \ulcorner C_3' \urcorner\right) \times 10 + \ulcorner C_4' \urcorner ,$$

gdzie $C_i' = C_i \times 000.017$ i C_0 jest zawsze cyfrą, a C_i dla $0 < i \leqslant 4$ jest cyfrą lub φ ;
wówczas

$$AS \equiv AC .$$

Jeżeli adresem rozkazu SAS jest liczba całkowita ze znakiem, niech

$$\text{Znak}_{def} \ \ulcorner + \urcorner \ , \ \ulcorner - \urcorner \ ,$$

wówczas

$$AS \equiv \text{Znak} \ * \ 18 \ + AC \ .$$

Jeżeli adresem rozkazu SAS jest etykieta międzyparagrafowa, to

$$AS = \ulcorner x \ Z_1 \ Z_2 \ Z_3 \urcorner \ .$$

3.2.3. Określanie wartości pozostałych pozycji instrukcji WJS .

Zn - jeżeli przed nazwą części operacyjnej w SAS znajduje się $\ulcorner . \urcorner$, wtedy Zn = 1 , inaczej Zn = 0 .

ZA - jeżeli dana zmienna w programie SAKO nie wystąpiła na liście zdania CAŁKOWITE, lub bezpośrednio po nazwie części operacyjnej w rozkazie SAS wystąpiła $\ulcorner . \urcorner$, wtedy ZA = 1 , inaczej ZA = 0 .

Mod - jeżeli jednym z indeksów zmiennej indeksowanej w SAKO jest zmienna, lub jeżeli ostatnim znakiem rozkazu w SAS jest +, wtedy Mod = 1 inaczej Mod = 0 .
Jeżeli indeksami zmiennej indeksowanej w SAKO są liczby całkowite, lub po ostatnim znaku adresu w SAS występuje liczba całkowita ze znakiem zamknięta w parę nawiasów/$\ulcorner$(i$\urcorner$)/, wtedy do ciągu instrukcji WJS zostaje dopisana dyrektywa IND /patrz p. 3.3/.

3.3. Dyrektywy w WJS

Zarówno kompilator SAKO jak i translator SAS muszą w WJS pozostawiać pewne dodatkowe dyspozycje dla adresacji. Instrukcje WJS zawierające w. w. informacje nazywamy dyrektywami.

Podstawową część wszystkich dyrektyw /PCD/ definiujemy następująco:

$$PCD = ((((Zn * 5 + OP) * 1 + Mod) * 1 + ZA) * 1 + Dod) * 3 + Puste,$$

gdzie:

$$Zn \underset{def}{=} 0$$

$$Op \underset{def}{=} 1, 2, \ldots, 13$$

$$Mod \underset{def}{=} 0$$

$$ZA \underset{def}{=} 0, 1$$

$$Dod \underset{def}{=} 1 .$$

Poniżej podajemy budowę poszczególnych dyrektyw WJS .

3.3.1. WYM

Zajmuje dwa słowa w WJS .

$$WYM\ 1 \equiv PCD * 24 + AS$$

$$Ilość\ słów \underset{def}{=} AC$$

$$WYM\ 2 = WYM + AC * 18 ,$$

gdzie w PCD OP = 1.

Dyrektywa WYM w SAKO powstaje ze zdania BLOK . Każdej zmiennej z listy odpowiada jedna dyrektywa WYM . Przez 'ilość słów' rozumiemy ilość miejsc w pamięci operacyjnej, obliczoną na podstawie zakresów indeksów.

W SAS powstaje przez translację dyrektywy WYM . Dyrektywa WYM powoduje zarezerwowanie dla danej zmiennej ilości miejsc pamięci równej liczbie zapisanej w WYM 2 .

3.3.2. TAB

Postać słów w WJS - taka, jak w WYM , z tym że

w PCD OP = 4 .

W SAKO powstaje ze zdania TABLICA w SAS z Translacji dyrektywy TAB . Poprzedza zawsze tablicę z liczbami.

Różnica pomiedzy dyrektywami WYM i TAB:

WYM rezerwuje miejsca dla zmiennych danego rozdziału w polu miejsc roboczych rozdziału. Działa na LMR.

TAB rezerwuje miejsca na liczby zawarte explicite w tablicy, we wnętrzu programu. Działa na LR.

3.3.3. LOC

$$\text{LOC} \equiv \text{PCD} * 24 + \text{AS} ,$$

gdzie OP = 3 .

Dyrektywa LOC powoduje zarezerwowanie jednego miejsca w ciągu rozkazów w wewnętrznym kodzie maszyny dla wielkości nazwanej daną zmienną.

Podczas kompilacji programu SAKO każdemu argumentowi lub wynikowi na liście zdania PODPROGRAM odpowiada LOC w tym podprogramie. W SAS powstaje z translacji dyrektywy LOC .

3.3.4. PAR

$$\text{PAR} \equiv \text{PCD} * 24 + \text{AS} ,$$

gdzie OP = 2 .

W SAKO powstaje ze zdania PODPROGRAM lub po wystąpieniu w formule arytmetycznej nazwy funkcji języka, patrz 3.3.6.

W SAS powstaje, gdy z lewej strony rozkazu wystąpi nazwa paragrafu. Dyrektywa PAR oznacza, że każdy występujący po niej symbol /etykieta lub zmienna/ posiada nowe znaczenie, niezależnie od analogicznego symbolu występującego przed tą dyrektywą. Nie dotyczy to etykiet międzyparagrafowych.

3.3.5. POC

$$\text{POC} \equiv \text{PCD} * 24 + \text{AC} ,$$

gdzie OP = 8 .

Nie odpowiada żadnemu z symboli czy zwrotowi SAKO .

W SAS powstaje z translacji numeru bezwzględnego.

Powoduje umieszczenie programu od miejsca AC pamięci operacyjnej maszyny.

3.3.6. FUN

$$FUN \equiv PCD * 24 + AC ,$$

gdzie OP = 5 . AC - adres pamięci pomocniczej maszyny.

Dyrektywa FUN jest dołączona do ciągu instrukcji WJS , gdy w programie SAKO użyto nazwy funkcji języka. Każdej nazwie funkcji języka odpowiada jedna lub kilka dyrektyw.

W SAS powstaje z translacji dyrektywy FUN , z tym że numer funkcji jest zastępowany przez jej adres w pamięci pomocniczej.

Znaczenie: Do rozdziału programu wynikowego dołącz podprogram, o którym informacje znajdują się w pamięci pomocniczej maszyny poczynając od wskazanego adresu.

3.3.7. IND

$$IND \equiv (PCD * 6 + Z) * 18 + AC ,$$

gdzie $Z \underset{def}{=} 0, 1$

OP = 12 .

W SAKO wartość liczbową AC obliczamy według

$$AC = (\ldots((i_1 \times d_2 + i_2) \times d_3 + i_3) \times d_4 + \ldots + i_{n-1}) \times d_4 + i_n ,$$

gdzie:

$i_1 , i_2 , \ldots , i_n$ są wartościami liczbowymi indeksów rozpatrywanej zmiennej,

$d_1 , d_2 , \ldots , d_n$ są odpowiednio wartościami liczbowymi zakresów indeksów,

oraz Z = 0 .

W SAS AC jest równe bezwzględnej wartości liczby całkowitej, zamkniętej w parę nawiasów, stojących za ostatnim znakiem adresu rozkazu.

Jeżeli znakiem liczby jest ⌜+⌝ lub ϕ, to Z = 0 ; jeżeli znakiem liczby jest ⌜−⌝ , to Z = 1 .

Dyrektywa IND powoduje podczas adresacji dodanie AC do przyporządkowanej adresowi symbolicznemu wartości liczbowej.

3.3.8. LIK

$$\text{LIK} = (\text{PCD} * 6 + Z) * 18 + \text{AC} ,$$

gdzie OP = 6 .

Poprzedza każdą liczbę całkowitą, która w programie SAKO lub SAS wystąpiła explicite.

Jeżeli znakiem liczby całkowitej jest ⌜+⌝ lub ϕ , to Z = 0; jeżeli tym znakiem jest ⌜−⌝ , to Z = 1 .

3.3.9. LID

Zajmuje w programie WJS dwa słowa.

$$\text{LID1} = \text{PCD} * 24 ,$$

gdzie OP = 7 .

LID2 zawiera przeliczoną na kod binarny i ustawioną w podanej skali liczbę ułamkową, która w programie SAKO lub SAS wystąpiła explicite.

3.3.10. ROZ

$$\text{ROZ} = \text{PCD} * 24 + \text{AC} ,$$

gdzie: OP = 10 AC – numer rozdziału.

Odpowiada: w SAKO - zdaniom ROZDZIAŁ i KONIEC ROZDZIAŁU ,
w SAS - dyrektywie ROZ .

Jeżeli przed programem SAKO lub SAS nie było odpowiednika dyrektywy ROZ , to do ciągu instrukcji WJS zostaje dopisana jako pierwsza dyrektywa ROZ , gdzie AC = 0 .
Znaczenie : Przejdź do Adresacji.

3.3.11. STA

$$STA \equiv (PCD * 8 + AC1) * 16 + AC2 ,$$

gdzie: OP = 11 AC 2 - numer pierwszego wykonywanego rozdziału programu.

Rozdział ten ma się zaczynać od miejsca AC 1 pamięci operacyjnej maszyny. W SAKO powstaje ze zdania KONIEC, gdzie AC 2 jest równe numerowi pierwszego z kompilowanych rozdziałów programu. AC 1 $_{def}$ 0. W SAS powstaje z translacji dyrektywy STA .

3.3.12. OMI

$$OMI \equiv PCD * 24 ,$$

gdzie OP = 9 .

W SAKO służy do wymazywania zbędnych instrukcji z ciągu WJS przez program optymalizujący.

W SAS nie istnieje.

Oznacza 'Nic nie rób' dla programu adresacji.

4. A d r e s a c j a

Program adresacji, w skrócie 'Adresator' , zajmuje się przetwarzaniem instrukcji wewnętrznego języka symbolicznego na wewnętrzny kod maszyny, wykorzystując do tego dane zawarte w listach. Instrukcje WJS są po I etapie zgrupowane w tzw. magazynie WJS . Maga-

zyn mieści się w całości w pamięci pomocniczej maszyny.

4.1. Listy

Adresator pracuje na dwóch rodzajach list.

Lista Adresów prowadzona przez kompilator SAKO lub translator SAS jest wykorzystywana i budowana w dalszym ciągu w czasie pracy Adresatora.

Lista Rozdziałów prowadzona przez Adresator jest wykorzystywana w czasie działania programu dla komunikacji pomiędzy rozdziałami.

4.1.1. Lista adresów

Dla danego słowa WJS element listy adresów definiujemy następująco

$$\text{Element} \equiv ((Zn * 1 + ZA) * 10 + \text{LICZNIK}) * 24 + AS ,$$

gdzie: $\text{LICZNIK} \underset{\text{def}}{=} LR, LMR$.

Lista adresów jest rozdzielona w zależności od AS na dwie podlisty:

- Podlista etykiet budowana jest przez kompilator SAKO lub translator SAS ,
- Podlista zmiennych budowana jest przez program adresacji.

4.1.1.1. Podlista etykiet - zasady budowy

1. Zn = 1 jest traktowane w ciągu elementów listy jako nawias, rozgraniczający dwa poziomy etykiet.
2. Element o Zn = 1 zostaje dopisany do listy etykiet wtedy, gdy do ciągu instrukcji WJS zostaje wpisana dyrektywa PAR, przy czym $AS_{PAR} = AS_{ELEM}$.
3. Elementom o tych samych wartościach AS zostają przydzielone wartości pozycji LICZNIK.

Dowolna ilość etykiet o różnych adresach symbolicznych może posiadać tę samą wartość pozycji LICZNIK .

Elementy o AS postaci : $\ulcorner x\ Z_0\ Z_1\ Z_2 \urcorner$ mogą wystąpić w dowolnym miejscu listy.

4.1.1.2. Podlista zmiennych

Początkowy stan listy definiujemy przez

Element = $\ulcorner 1 \urcorner * 35$.

Adresator dopisuje nowy element do podlisty zmiennych w momencie spotkania w ciągu instrukcji WJS jednej z dyrektyw : WYM , TAB , LOC , lub gdy spotka instrukcję WJS , w której AS = $L_0\ Z_1\ Z_2\ Z_3$. W tym ostatnim wypadku dopisywanie odbywa się tylko wówczas, gdy w ciągu elementów podlisty brak jest elementu o AS równym AS rozpatrywanej instrukcji WJS .

Dopisywanie na podlistę zmiennych dokonuje się przez odwołanie do rozpisanego poniżej podprogramu SAKO :

```
PODPROGRAM :  (LIC ;  PN)  = DOPISANIE  (WJS, LICZNIK, P, L2, AC)
BLOK  (215) :  S
CALKOWITE  :  LICZNIK ,     P, L2, AC, I, Z, LIC, PN
I  =  LICZNIK  x  000.001
Z  =  WJS  x  002.000
GDY  Z = 0 : 1, INACZEJ NASTEPNY
SKOCZ WEDŁUG  I :  3,2
3) LIC  = LICZNIK + 1 + 2 x AC
SKOCZ DO 4
2) LIC  = LICZNIK + 2 x AC
SKOCZ DO 4
1) LIC  = LICZNIK + AC
4) WJS  = LIC * 6 + Z * 6 + WJS x 000.077.777.777
S (P)  = WJS
```

```
PN  = P + 2
S (PN) = -0
GDY ABS (ABS (LIC) - ABS (L2)) < 2 : NASTEPNY, INACZEJ 5
IDŹ DO ROZDZIAŁU : SYGNALIZACJA BŁĘDU
5) WRÓĆ
```

gdzie: S - blok elementów listy adresów,
WJS - rozpatrywana instrukcja WJS ,
P - wskaźnik pierwszego wolnego miejsca na podliście zmiennych,
L2 - gdy LICZNIK = LR to L2 = LMR
gdy LICZNIK = LMR to L2 = LR
I, Z - zmienne robocze podprogramu.

Dyrektywa PAR napotykana w ciągu instrukcji WJS ustawia zawsze podlistę zmiennych w jej stan początkowy. Jest to równoważne przejściu na nowy poziom zmiennych.

4.1.2. Lista rozdziałów

Lista rozdziałów jest zbudowana w pamięci pomocniczej maszyny i aktualny jej stan jest reprezentowany przez jedno słowo w pamięci operacyjnej maszyny. Definiujemy je następująco:

$$\text{Słowo} = AC1 * 18 + AC2 ,$$

gdzie: AC1 - numer aktualnie adresowanego rozdziału, lub na etapie wykonywania programu numer aktualnie liczącego rozdziału programu.

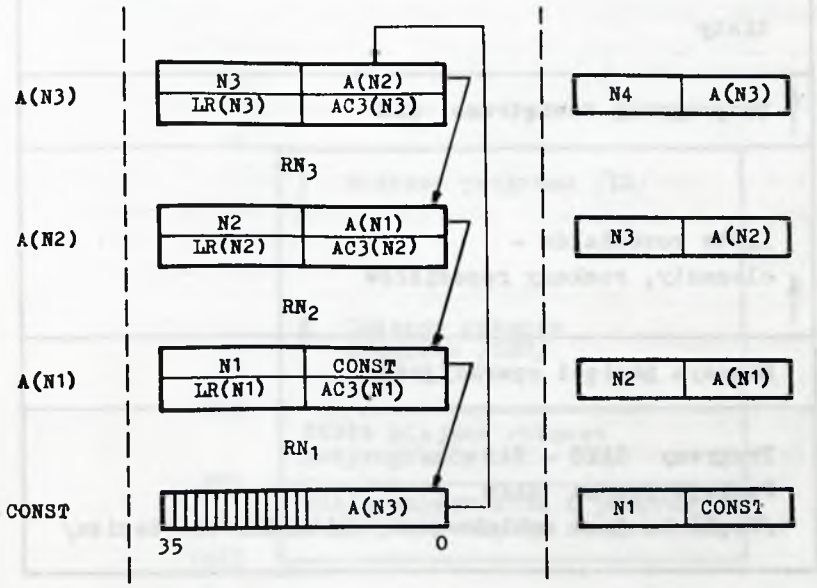
AC2 - na obu ww. etapach adres pamięci pomocniczej, od którego zaczyna się informacja o aktualnym rozdziale.

Dopisywanie nowego elementu do listy rozdziałów odbywa się w momencie napotkania dyrektywy ROZ w ciągu instrukcji WJS . Wykonywany jest wtedy program następujący:

```
CAŁKOWITE  :  AC2, LR, A,
AC = WJS x 000.000.177.777
A  = AC2 - LR - 4
PP(A) = Słowo
B  = AC3 + LR
PP(A+2)= B
B = 377.777 + A * 18
PP(CONST) = B
Słowo  = (AC * 18 + A) * 18
```

A, B - miejsca robocze w.w. programu
LR - maksymalny zakres licznika rozkazów w danym rozdziale
AC3 - adres początku rozdziału w pamięci operacyjnej maszyny
PP(K)- K jest adresem pamięci pomocniczej PP .
Pozostałe oznaczenia - jak w punktach poprzednich.

Poczynając od adresu A + 4 , umieszczane są kolejno rozkazy przeadresowanego aktualnie rozdziału. Dla objaśnienia dodatkowego podajemy poniżej szkic 3-elementowej listy rozdziałów.



Objaśnienia:

A (I) - Adres pamięci pomocniczej - początek informacji o rozdziale numer I

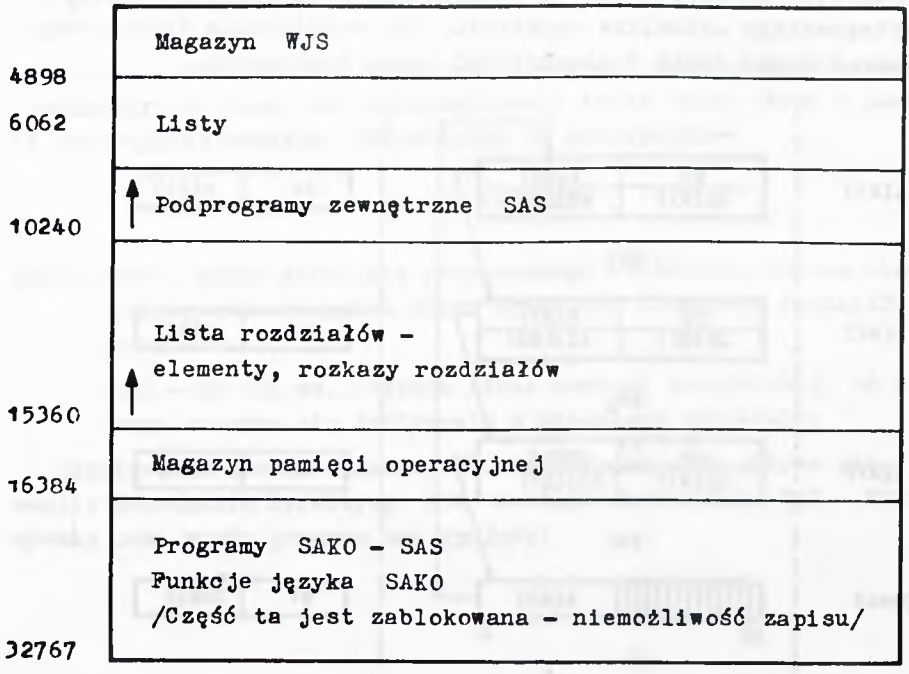
RN (I) - Rozkazy rozdziału o numerze I .

W prawej kolumnie pokazano kolejne /licząc od dołu/ aktualne stany słowa w pamięci operacyjnej maszyny, reprezentującego listę rozdziałów.

4.1.3. Plany pamięci maszyny

Programy systemu SAKO - SAS - ADRESATOR , działając w pamięci operacyjnej, wykorzystują pamięć pomocniczą do zapisywania wyników działania.

4.1.3.1. Pamięć pomocnicza



Magazyn WJS – służy do przechowywania dla programu adresacji programu w WJS.

Listy – służą do przechowywania zawartości list deklaracji SAKO.

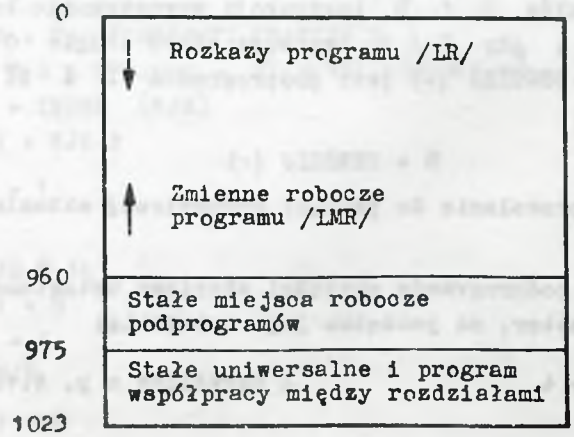
Podprogramy – służą do przechowywania podprogramów bibliotecznych, które mają wejść w skład adresowanego programu.

Lista rozdziałów – służy do przechowywania elementów listy rozdziałów, jak i wszystkich rozkazów programu.

Magazyn pamięci operacyjnej – służy do przechowywania zawartości pamięci operacyjnej podczas wykonywania czynności awaryjnych.

4.1.3.2. Pamięć operacyjna

Szkic przedstawia bieg licznika rozkazów LR i licznika miejsc roboczych LMR . Końcowy stan tych liczników odpowiada rzeczywistemu rozmieszczeniu rozkazów i zmiennych roboczych w zaadresowanym rozdziale programu.



4.2. Przebieg adresacji programu WJS .

Adresator, po sprowadzeniu do pamięci operacyjnej maszyny podlisty etykiet i ustawieniu wartości początkowych /liczników: LR = 0, LMR = 960/, pobiera partiami z magazynu WJS materiał do adresacji. W danej partii bada, czy słowo WJS jest instrukcją czy dyrektywą. Algorytmy adresacji instrukcji podajemy w programach p. 4.2.1, algorytmy adresacji dyrektyw podajemy w p.4.2.2.

Proces adresacji trwa do momentu zidentyfikowania w ciągu WJS dyrektywy STA lub ROZ .

Standartowe czynności adresatora wykonywane są w czasie opracowywania instrukcji i dyrektyw; wyodrębnione są w podprogramy.

Algorytm realizowany przez pierwszy z nich rozpisano w p.4.1.1.2.

Pobieranie partiami materiału do adresacji wykonuje podprogram POBIERZ (·). Odwołanie

B = POBIERZ (·)

ustawia w miejscu B kolejne słowo WJS . Zapisywanie partiami przeadresowanego materiału w pamięci pomocniczej wykonuje podprogram ZAPISZ (WJS, L, N) . Odwołanie

B = ZAPISZ (C, D, N)

powoduje wpisanie do zarejestrowanego w pamięci bloku OP zestawionej ze słów C i D instrukcji wewnętrznego kodu maszyny, gdy N = 0 ; gdy N ≠ 0, wpisuje słowo długie C do bloku OP . Podprogram ODEŚLIJ (·) jest podprogramem I i II rzędu. Odwołanie

B = ODESLIJ (·)

realizuje przesłanie do pamięci pomocniczej aktualnego stanu bloku OP .

Dla ww. podprogramów wartości startowe ustawione są zewnętrznie przez adresator, na początku jego działania:

A0 = A + 4 A określone w p. 4.1.2

```
J = 0                    J bieżący indeks bloku OP
IO = CONST 1             IO adres pamięci pomocniczej,
                            skąd pobierane są kolejne
                            grupy instrukcji WJS .
I1 = 32                  I1 bieżący indeks bloku PP .
```

Rozpisane w SAKO algorytmy realizowane przez ww. podprogramy podajemy poniżej.

```
        PODPROGRAM :  WJS = POBIERZ (·)
        BLOK (31)  :  PP
        CALKOWITE  :  IO, I1
        GDY I1 - 32 = 0:  NASTEPNY, INACZEJ 1
        CZYTAJ z BEBNA OD IO:*PP
        I1 = 0
        IO = IO + 32
1)      WJS = PP (I1)
        I1 = I1 + 2
        WROC
        PODPROGRAM: ZAPISZ  (WJS, L, N)
        CALKOWITE:  OP, J, N, B
        BLOK (32) :  OP
        GDY  J - 32 = 0 :  NASTEPNY, INACZEJ 1
        B = ODESLIJ (·)
1)      GDY  N = 0:  NASTEPNY, INACZEJ 3
        WJS = WJS x 776.000 +  (L x 177.700)*(-6)
        WJS 2 = INDEX  (WJS)
        OP (J) = WJS 2
2)      J = J + 1
        WROC
3)      B = WJS * 18
        OP (J) = B
        J = J + 1
        B = WJS
        OP (J) = B
        SKOCZ  DO 2
```

k) Parzystość J dla N ≠ 0, jest zagwarantowana przez programy I etapu /Kompilator SAKO lub translator SAS/.

```
PODPROGRAM:  ODEŚLIJ (·)
PISZ NA BEBEN OD AO: * OP
K/ ZAWSZE  J  SŁÓW KRÓTKICH
AO = AO + J
J = 0
WRÓĆ
```

Podany poniżej program realizuje następny takt pracy adresatora.

```
CAŁKOWITE:  V, U, I
P)  WJS = POBIERZ (·)
    V ≡ WJS x 001.000
    GDY V = 0: NASTĘPNY INACZEJ 1 DYR
    V ≡ WJS x 000.077
    Z ≡ WJS x 000.077.777.777
    GDY V = 0: 1 A RZECZ, INACZEJ NASTĘPNY
    U ≡ V x 000.040
    GDY U = 0: NASTĘPNY, INACZEJ  1 AZM
    GDY V - ⌜-⌝ = 0:  1 AW, INACZEJ NASTĘPNY
    GDY V - ⌜+⌝ = 0:  2 AW, INACZEJ NASTĘPNY
    GDY V - ⌜*⌝ = 0:  1 AP, INACZEJ NASTĘPNY
    GDY V - ⌜x⌝ = 0: 1 AMP, INACZEJ NASTĘPNY
    SKOCZ DO 1 AWP
1 DYR) I ≡(WJS x 370.000) * (-12)
    SKOCZ WEDŁUG I: 1 WYM, 1 TAB, 1 LOC, 1 PAR, 1 POC -
    1 PUN, 1 IND, 1 LIK, 1 LID, 1 ROZ, 1 STA, 1 OMI
```

4.2.1 Adresacja instrukcji

4.2.1.1 Instrukcje z adresem rzeczywistym.

```
1 A RZECZ)   Z ≡ (WJS x 000.000.777.777) * 24
2P)  B = ZAPISZ  (WJS, Z, 0)
     SKOCZ DO  1P
```

Uwaga:

Przyjmujemy dla ułatwienia, że Deklaracja CALKOWITE występuje tylko wtedy, gdy zachodzi potrzeba dodeklarowania nowej zmiennej. Wartości deklarowane są tylko raz na cały program i podprogramy adresatora.

4.2.1.2 Instrukcja z adresem względnym.

```
1 AW)  CALKOWITE: U, LR
       U ≡ (WJS x 000.000.777.777) * 18
       U =  LR - U
       Z ≡  U * 6
       SKOCZ DO 2P
2 AW)  U ≡ (WJS x 000.000.777.777) * 18
       U =  LR + U
       Z ≡  U * 6
       SKOCZ DO  2P
```

4.2.1.3 Instrukcja o etykiecie międzyparagrafowej.

```
1 AMP) CALKOWITE: MAXS, I
   K)  MAXS JEST MAKSYMALNYM ZAKRESEM PODLISTY ETYKIET  S
   K)  S jest blokiem zadeklarowanym w podprogramie DOPISANIE
```

```
*3)  U = S (I)
     V ≡ 4 x 000.077.777.777
     GDY Z - V = 0: 3P, INACZEJ NASTEPNY
     POWTORZ OD 3: I = 0  (1)  MAXS
     SKOCZ DO  1P
3P)  U = ZAPISZ  (WJS, U, 0)
     SKOCZ DO  1P
```

4.2.1.4. Instrukcja z symbolem podprogramu.

```
1 AP)*4)  U = S (I)
          V ≡ U x 400.077.777.777
          GDY 0 > V:5, INACZEJ NASTEPNY
       6) POWTORZ  OD 4: I = 0 (1) MAXS
          IDZ DO ROZDZIAŁU :  SYGNALIZACJA BŁEDU
       5) GDY ABS  (U - V)  = 0: 3P, INACZEJ NASTEPNY
          SKOCZ DO  6
```

4.2.1.5. Instrukcja o etykiecie wewnątrzparagrafowej.

M jest wskaźnikiem **listy** adresów S takim, że w miejscu S (M - 2) znajduje się element rozgraniczający dwa różne poziomy etykiet /p. 4.1.1.1 i 3.3.4/.

```
1 AWP)  CALKOWITE: M
        I = M
7)      U = S (I)
        V ≡ U x 400.077.777.777
        GDY 0 > V: NASTEPNY, INACZEJ 8
        IDZ DO ROZDZIALU:  SYGNALIZACJA BLEDU
```

```
8)   GDY Z - V = 0: 3P, INACZEJ NASTEPNY
     I = I + 2
     SKOCZ DO 7
```

4.2.1.6 Instrukcja zmiennej w adresie symbolicznym.

P1 jest wskaźnikiem początku podlisty zmiennych, listy adresów S. P2 - wskaźnikiem pierwszego wolnego miejsca tej podlisty. Na początku adresacji jest równe P1 = P2 = MAXS + 2

```
1 AZM)  CALKOWITE:   LMR, I, P1, B, C, P2, D
        I = P1
9)      U = S (I)
        GDY U = 0: NASTEPNY INACZEJ 10
        B =-LMR
        C = P2
        (D, P2) = DOPISANIE  (WJS, B, C, LR, 1)
        LMR = - D
        U = S (P2 - 2)
        SKOCZ DO 3P
10)     V ≡ U x 000.077.777.777
        GDY Z - V = 0: 3P, INACZEJ NASTEPNY
        I = I + 2
        SKOCZ DO 9
```

4.2.2. Przetwarzanie dyrektyw.

Dyrektywy dostarczają adresatorowi informacji, na podstawie których może on pracować rozpatrując każdorazowo tylko jedną instrukcję WJS.

```
1 WYM)  CALKOWITE:  AC
        WJS 1 ≡ POBIERZ (·)
        AC  ≡ WJS 1 * 18
        B = - LMR
        C = P2
       (D, P2) = DOPISANIE  (WJS, B, C, LR, AC)
        LMR = - D
        SKOCZ DO 1P
1 TAB)  WJS 1 = POBIERZ (·)
        AC ≡ WJS 1 * 18
        B = LR
        C = P2
       (LR, P2) = DOPISANIE  (WJS, B, C, LMR, AC)
        SKOCZ DO 1P
1 LOC)  D ≡ (WJS x 002.000) * 10
        B = LR
        C = P2
       (LR, P2) = DOPISANIE  (WJS, B, C, LMR, 1)
        B = ZAPISZ  (0, 0, D)
        SKOCZ DO 1P
1 PAR)  I = M
   12)  GDY 0 > S (I) :  11, INACZEJ NASTEPNY
        I = I + 2
        SKOCZ DO 12
   11)  P1 = P2
        M = I + 2
        SKOCZ DO 1P
1 POC)  D ≡ (WJS x 000.000.001.777) * 18
        GDY D = LR:  1P, INACZEJ NASTEPNY
        B = ODESLIJ (·)
```

```
     AO = A + 4 + D
K)   A  określone w p. 4.1.2
     LR = D
     SKOCZ DO 1P
K)   KOMENTARZ DO 1 FUN: Jest to oddzielna część adresatora, dzia-
K)   łająca na bloku miejsc krótkich równym ilości miejsc pamięci
K)   operacyjnej maszyny, zajmowanej przez przeadresowywany podpro-
K)   gram. C 1  jest ilością rozkazów podprogramu do przeadresowa-
K)   nia; C2 jest ilością miejsc pamięci operacyjnej zajmowanych
K)   przez podprogram.
1 FUN) CALKOWITE: C1, C2, H
       B = LR
       LR = LR + C2
       STRUKTURA (C2) : H
       B = ODESLIJ (·)
 * 13) GDY 0 > H (I) : NASTEPNY, INACZEJ 14
       H (I) = H (I) - B
   14) POWTORZ OD 12: I = 0 (1) C1
       PISZ NA BEBEN OD AO: * H
       AO = AO + C2
       SKOCZ DO 1P
K)     Dyrektywa IND może stać za każdą z dowolnych instrukcji WJS
K)     - realizuje ją wtedy podprogram II rzędu  INDEX.
       PODPROGRAM: (A) = INDEX (B)
       CALKOWITE: A,B,C,D,E
   16) W = POBIERZ (·)
       D ≡  777.777 x W
       C ≡  151.000
       GDY D = C: NASTEPNY, INACZEJ 15
       E ≡ (W x 000.000.777.777) * 18
       A = B + E
       SKOCZ DO 16
```

```
15)   A = B
      I1 = I1 - 2
K)    I1 z podprogramu POBIERZ
      WROC
K)    Jeżeli dyrektywa IND jest pobrana do analizy przez część
K)    wstępną adresatora /od etykiety 1P/, to jest traktowana:
1 IND) SKOCZ DO 1P
K)    Jest to pusty takt pracy adresatora.
1 LIK) (B) = PL (WJS, 0)
       SKOCZ DO 1P
1 LID) WJS = POBIERZ (·)
       (B) = PL (WJS, 1)
       SKOCZ DO 1P
       PODPROGRAM: (B) = PL (A,N)
       CALKOWITE: N,C,B
       GDY I - 32 = 0: NASTEPNY, INACZEJ 17
K)     I z ZAPISZ
       B = ODESLIJ (·)
17)    GDY N = 0: NASTEPNY, INACZEJ 19
       B ≡ (WJS x 000.000.777.777)*18
       OP (J) = B
18)    J = J + 1
       WROC
19)    B ≡ WJS * 18
       OP (J) = B
       J = J + 1
       B ≡ WJS
       OP (J) = B
       SKOCZ DO 18
K)     Uwaga ta sama jak w podprogramie ZAPISZ.
1 ROZ) X = 0
20)    AC ≡ WJS x 000.000.177.777
K)     p.4.1.2
       B = ODESLIJ (·)
       IDZ DO ROZDZIALU: INTERPRETACJA ZĄDAŃ PROGRAMISTY.
```

```
1 STA) CAŁKOWITE: ASTA
       ASTA ≡ (WJS x 000.377.600.000) * 2
       X = 1
       SKOCZ DO 20
1 OMI) SKOCZ DO 1A.
```

Jest to pusty takt pracy adresatora.
Dyrektywą tą program optymalizacji wymazuje zbędne instrukcje w WJS.

5. W s m i a n k a h i s t o r y c z n a.

Wewnętrzny język symboliczny powstał w wyniku seminariów w Zakładzie Programowania Instytutu Maszyn Matematycznych PAN. W seminarium tym uczestniczyli prof.dr L.Łukaszewicz, mgr A.Mazurkiewicz i mgr J.Swianiewicz. Pierwszy translator SAS dla maszyny XYZ zrealizowano w 1960 r.

Pierwszy Kompilator SAKO dla XYZ zrealizowano w 1961 r. Drugi, znacznie ulepszony i rozszerzony translator SAS dla maszyny ZAM-2, zrealizowano pod koniec roku 1961. SAKO w ulepszonej wersji dla maszyny ZAM-2 zrealizowano w początkach 1962 roku.

Autorzy pragną szczególnie gorąco podziękować mgr A.Mazurkiewiczowi za wydatną pomoc i cenne uwagi udzielone podczas pisania artykułu.

Literatura

1. System automatycznego kodowania SAKO, Cz.I. Opis języka. Prace ZAM PAN Ser.C Nr.2, Warszawa 1961.
2. Maszyna ZAM-2. Kompendium programowania w języku SAS. Prace ZAM PAN Ser.C Nr.3, Warszawa 1962.
3. ŁUKASZEWICZ L.: SAKO - An Automatic Coding System, Annual Rev. in Autom. Program, 1961:2
4. MAZURKIEWICZ A.: Arithmetic Formulae and the Use of Subroutines in SAKO, Annual Rev. in Autom. Program, 1961:2
5. SWIANIEWICZ J., SAWICKI S.: SAKO Translation, preprint. Presented at the Conf. on Automatic Programming Warsaw, Sept. 1961.
6. SZORC P.: Subroutines in SAKO, preprint. Presented at the Conf. on Automatic Programming Warsaw, Sept. 1961.
7. BOROWIEC J.: Translation of Arithmetic Formulae in SAKO, Prace IMM PAN, Ser. Algorytmy, 1962:1, 1, 37-56.
8. SCHURMANN A.: O translacji formuł arytmetycznych SAKO, Prace IMM PAN, Ser. Algorytmy, 1962:1,1,57-66.

A METHOD OF ADDRESSING USED IN SAKO-SAS TRANSLATIONS

Summary

A method of a translation of the symbolic language, so-called WJS, into the machine language is described. WJS is the result of the action of a SAKO compiler and a SAS translator.

A given WJS statement may be an instruction or a directive. Every instruction corresponds to a machine language instruction. Every directive corresponds to a determined action of the WJS program the so-called Addressator.

A WJS statement may include:

- the sign
- the operation number in the internal machine code
- the modification sign
- the address sign
- the sign to distinguish directive from other statements
- the symbolic address, which may be a variable, a label, an integer with a sign /relative address/ or an integer unsigned /real address/.

Because of its generality the described method can be used not only for SAKO and SAS translators.

THEORY OF COMPUTERS

Instytut Maszyn Matematycznych PAN
Algorytmy N° 2

517.11

CALCULATION OF PRIME IMPLICANTS OF TRUTH FUNCTIONS

by Stanisław WALIGÓRSKI

Received October 8th, 1962

A method of calculating prime implicants of the truth function on the digital computer is presented. Using this method, computations can be performed as recursive procedures. Three main operations are used for calculating prime implicants, decomposition of zero-one sequence, alternative of elements of one set with elements of another set, taking minimal elements of a set of zero-one sequences. An additional reduction of intermediate results depending on the cost function of expressions may also be performed.

Introduction

The method of determining disjunctive normal equivalents of truth functions described in [14] requires the knowledge of an algorithm determining the set of all prime implicants or the set A_{max} of prime implicants, and of all conjunctions greater than the prime implicants for an arbitrary isotone function.

Quite a number of algorithms for calculating prime implicants have been developed /refer, among others to [2-7], [10-12] /. However, the effectiveness of the majority of existing methods decreases rapidly with the increase of the number of variables and with the complexity of the function. Digital computers considerably enlarge the possibility of calculation, however there still exist rather strict limitations due either to an exceedingly great quantity of stored data /e.g. too great a quantity of intermediate results/ or to an exceedingly great number of computer operations. Therefore the problem of finding an algorithm for

calculating prime implicants of a function of a great number of variables is still an open question. The development of those methods and their suitability for digital computers is a matter of great importance.

This paper presents the method for calculating the set of all prime implicants adapted for digital computers. In the discussed method all values of variables for which the function is equivalent to 0 are taken as input data. These values may be entered successively during computation. Intermediate results can be reduced in each phase of computation; this may effectively decrease the storage capacity needed.

The terminology and notation used in this paper are in principles like that in [14]. The set of all N-element zero-one sequences is denoted by B^N. The sequences belonging to B^N are denoted by small Latin letters. The elements of these sequences are denoted by letters, identical with those denoting the whole sequence, with appropriate indices, and thus, for $x \in B^N$ we have $x = x_1, x_2, \ldots, x_N$. In order to distinguish explicitly the meet and joint operations on B^N from the union and intersection of the subsets of B^N, we introduce, in contradistinction to [14], the following denotations:

meet of elements x, y of the set B^N :	$x \cap y$
joint of elements x, y of the set B^N :	$x \cup y$
intersection of sets K and L :	$K \cdot L$
union of sets K and I :	$K + L$
intersection of all sets K_i of a certain family:	$\prod_{i \in I} K_i$
union of those sets	$= \sum_{i \in I} K_i$
difference of sets K and L	$= K-L$
the empty set	$= 0$
the least element of B^N (sequence of zeros)	$= 0$
the set of atoms of B^N (the set of all sequences $e \in B^N$ containing a single 1)	$= E$

We also introduce the following operation on the subsets of B^N : if K, $L \in B^N$, then

$$K \cup L = B^N \left\{ x : (\exists\, a \in K)(\exists\, b \in L)\ x = a \cup b \right\}$$

The set being the result of operation performed on all sets of a certain family is denoted $\bigvee_{i \in I} K_i$.
The set of all minimal elements of set K is denoted by Min K .
The set of all maximal elements of set K is denoted by Max K .

For an established isotone truth function of N variables, let Z denote the subset of B^N, on which this function equals 0. Hence /refer to [14]/

$$A_{max} = B^N \left\{ x : (\forall\, y \in Z)\ x \not\leqslant y \right\} = B^N \prod_{y \in Z} \left\{ x : x \not\leqslant y \right\} \qquad /1/$$

But, $x \not\leqslant y$ if, and only if, $x \cap \bar{y} \neq 0$, i.e. if there exists a zero-one sequence $e \in E$ containing a single 1 such that $e \leqslant x \cap \bar{y}$.
Hence

$$\left\{ x : x \not\leqslant y \right\} = \left\{ x : (\exists e \in E) e \leqslant x \cap \bar{y} \right\} = \sum_{\substack{e \leqslant \bar{y} \\ e \in E}} \left\{ x : e \leqslant x \right\} \qquad /2/$$

$$A_{max} = \prod_{y \in Z} \sum_{\substack{e \leqslant \bar{y} \\ e \in E}} \left\{ x : e \leqslant x \right\} \qquad /3/$$

In the method determining Min A_{max} the following properties of operations $\cdot$, $+$, $\cup$, Min will be used:

If $K \subset B^N$, $L \subset B^N$, $M \subset B^N$, then

$K \cup L = L \cup K$

$(K \cup L) \cup M = K \cup (L \cup M)$

$K \subset K \cup K$

If $K \subset L$, then $K \cup M \subset L \cup M$

Min $K \subset K$

Min Min K = Min K

For every $a \in K$ such $b \in \mathrm{Min}K$ exists that $b \leqslant a$.

Those properties follow immediately from definitions of $\cup$ and Min. Moreover, we have the following lemmas:

Lemma 1

If $K \subset B^N$, $L \subset B^N$, and both these sets have the property: if $a \in K$ and $b \geqslant a$, then $b \in K$, and the same holds for L, then

$$K \cdot L = K \cup I \qquad /4/$$

where $K \cup L = B^N \left\{x: (\exists a \in K)\ (\exists b \in L)\ x = a \cup b\right\}$

Proof

If $a \in K$ and $b \in L$, then $a \cup b \geqslant a$ and $a \cup b \geqslant b$ implies $a \cup b \in K \cdot L$ i.e. $K \cup L \subset K \cdot L$. Conversely, if $c \in K \cdot L$ then $c \cup c = c \in K \cup L$, hence $K \cdot L \subset K \cup L$.

Lemma 2

If $K_1, K_2, \ldots, K_m$ are subsets of B^N, then

$$\mathrm{Min} \bigvee_{i=1}^{m} K_i = \mathrm{Min}\left(K_m \cup \mathrm{Min} \bigvee_{i=1}^{m-1} K_i\right) \qquad /5/$$

Proof

If $m = 2$, then the equality /5/ reduces to

$$\mathrm{Min}\,(K_2 \cup K_1) = \mathrm{Min}\,(K_2 \cup \mathrm{Min}\, K_1) \qquad /6/$$

If /6/ is true for arbitrary $K_1, K_2 \subset B^N$, then

$$\mathrm{Min} \bigvee_{i=1}^{m} K_i = \mathrm{Min}\left(K_m \cup \bigvee_{i=1}^{m-1} K_i\right) = \mathrm{Min}\left(K_m \cup \mathrm{Min} \bigvee_{i=1}^{m-1} K_i\right). \qquad /7/$$

Hence it is sufficient to prove /6/ for every $K_1, K_2 \subset B^N$.

$K_2 \cup \operatorname{Min} K_1 \subset K_2 \cup K_1$ and there is no element of $K_2 \cup K_1$ less than an arbitrary minimal element of $K_2 \cup \operatorname{Min} K_1$; hence every element minimal in $K_2 \cup \operatorname{Min} K_1$ is minimal in $K_2 \cup K_1$ and $\operatorname{Min}(K_2 \cup \operatorname{Min} K_1) \subset \operatorname{Min}(K_2 \cup K_1)$.

Suppose that there exists $c \in \operatorname{Min}(K_2 \cup K_1) - (K_2 \cup \operatorname{Min} K_1)$. Then, $c \in \operatorname{Min}(K_2 \cup K_1)$ means that there exist such $a \in K_2$ and $b \in K_1$, that $c = a \cup b$. $a \cup b \notin K_2 \cup \operatorname{Min} K_1$ implies that $b \notin \operatorname{Min} K_1$, and consequently there exists such $b' \in \operatorname{Min} K_1$ that $b' < b$ and $c > a \cup b' \in K_2 \cup \operatorname{Min} K_1$. It follows that $c \notin \operatorname{Min}(K_2 \cup K_1)$, and we finally obtain a contradiction of the assumption. Therefore $\operatorname{Min}(K_2 \cup K_1) \subset K_2 \cup \operatorname{Min} K_1$.

Since there is no element of $K_2 \cup \operatorname{Min} K_1$ less than an arbitrary element of $\operatorname{Min}(K_2 \cup K_1)$, we have $\operatorname{Min}(K_2 \cup K_1) \subset \operatorname{Min}(K_2 \cup \operatorname{Min} K_1)$.

Corollary

$$\operatorname{Min} \bigvee_{i=1}^{m} K_i = \operatorname{Min} \bigvee_{i=1}^{m} \operatorname{Min} K_i \qquad /8/$$

Proof

$$\operatorname{Min} \bigvee_{i=1}^{m} K_i = \operatorname{Min}\Big(\bigvee_{i=1}^{m-1} K_i \cup \operatorname{Min} K_m\Big) = \operatorname{Min}\Big(\operatorname{Min} K_m \cup \bigvee_{i=1}^{m-2} K_i \cup K_{m-1}\Big) =$$

$$= \operatorname{Min}\Big(\operatorname{Min} K_m \cup \operatorname{Min} K_{m-1} \cup \bigvee_{i=1}^{m-2} K_i\Big)$$

Using this transformation m-1 times we obtain /8/ .

Lemma 3

If $K_1, K_2, \ldots, K_m$ are subsets of B^N, then

$$\operatorname{Min} \sum_{i=1}^{m} K_i = \operatorname{Min} \left(K_m + \operatorname{Min} \sum_{i=1}^{m-1} K_i\right) \qquad /9/$$

Proof

Similarly as in the proof of Lemma 2 it is sufficient to prove /9/ only for $m = 2$:

$$\operatorname{Min} \left(K_2 + K_1\right) = \operatorname{Min} \left(K_2 + \operatorname{Min} K_1\right) \qquad /10/$$

If $a \in \operatorname{Min} \left(K_2 + K_1\right)$ then $a \in K_2 + K_1$, and for every $b \in K_2 + K_1$ we have $b \not< a$. This implies $a \in \operatorname{Min} K_2 + \operatorname{Min} K_1 \subset K_2 + \operatorname{Min} K_1$, and furthermore $a \in \operatorname{Min} \left(K_2 + \operatorname{Min} K_1\right)$.

Hence $\operatorname{Min} \left(K_2 + K_1\right) \subset \operatorname{Min} \left(K_2 + \operatorname{Min} K_1\right)$.

The reversed inclusion $\operatorname{Min} \left(K_2 + \operatorname{Min} K_1\right) \subset \operatorname{Min} \left(K_2 + K_1\right)$ is also true, because $K_2 + \operatorname{Min} K_1 \subset K_2 + K_1$ and there is no element of $K_2 + K_1$ less than an arbitrary minimal element of $K_2 + \operatorname{Min} K_1$.

Corollary

$$\operatorname{Min} \sum_{i=1}^{m} K_i = \operatorname{Min} \sum_{i=1}^{m} \operatorname{Min} K_i \qquad /11/$$

The proof is analogous to that of the equality /8/.

Using the above lemmas we have

$$A_{max} = \prod_{y \in Z} \sum_{\substack{e \in \bar{y} \\ e \in E}} \left\{x: x \geq e\right\} = \bigvee_{y \in Z} \sum_{\substack{e \in \bar{y} \\ e \in E}} \left\{x: x \geq e\right\} \qquad /12/$$

$$\operatorname{Min} A_{max} = \operatorname{Min} \bigvee_{y \in Z} \sum_{\substack{e \in \bar{y} \\ e \in E}} \left\{x: x \geq e\right\} = \operatorname{Min} \bigvee_{y \in Z} \operatorname{Min} \sum_{\substack{e \in \bar{y} \\ e \in E}} \operatorname{Min} \left\{x: x \geq e\right\} = \operatorname{Min} \bigvee_{y \in Z} R\left(\bar{y}\right) \qquad /13/$$

where

$$R\left(\bar{y}\right) = E\left\{e: e \leq \bar{y}\right\} \qquad /14/$$

If we number the elements of Z forming the sequence

$$y_{(1)}, y_{(2)}, \ldots, y_{(l)} \qquad /15/$$

then for calculating A_{max} the following recurrent formulae are obtained:

$$U_1 = R(\bar{y}_{(1)}) \qquad /16/$$

$$U_{i+1} = \text{Min}\,(U_i \cup R(\bar{y}_{(i+1)})) \qquad i < k \qquad /17/$$

$$\text{Min}\, A_{max} = U_k \qquad /18/$$

Operation Min may also be applied to partial results obtained during the computation of $U_i \cup R(\bar{y}_{i+1})$. This operation can be omitted in some steps according to /17/, /even in all of them except the last one/, particularly in the case when the number of intermediate results obtained while computing is rather small.

In the case of computing γ - minimal normal expressions, when function γ satisfies assumptions of the Theorem 3 [14], /the value of γ decreases when conjunctions in the expression are cancelled/, a smaller set can be taken instead of $\text{Min} \bigvee_{y \in Z} R(\bar{y})$, and all such $c \in \text{Min} \bigvee_{y \in Z} R(\bar{y})$ are rejected for which there does not exist such $a \in \text{Min}\{x: f(x) = 1\}$ that $c \leqslant a$. This results directly from the properties of the family of sets corresponding to normal equivalents of the given function [14]. Thus, in each step of computation /13/ we can reject elements with given properties from U_{i+1}. This is possible, as each step causes at most an increase of elements of U_i, and if $b \not\leqslant a$ and $b \leqslant c$, then $c \not\leqslant a$. The described reduction and operation Min are commutative. It should be noted, that such a reduction cannot be used to calculate the Quine's table by the method presented in [13] as the function there is isotone and equals 1 only for such a sequence that is greater than any other sequence.

If the function satisfies the assumptions of the Theorem 2 [14], i.e. $X \subset Y$ and $X \neq Y$ imply $\gamma(X) < \gamma(Y)$, and if besides $\gamma(X) \gamma(Y)$ implies $\gamma(X + T) < \gamma(Y + T)$ for arbitrary X, Y, T,

then calculating the Quine's table, we can reject all elements of U_i for which the value of γ will not be minimal. Indeed, let us suppose that $a, b \in U_i$ and $\gamma(a) < \gamma(b)$. For arbitrary $e \in R(\overline{y}_{(i+1)})$ we have $a \cup e \in U_i \cup R(\overline{y}_{(i+1)})$ and $b \cup e \in U_i \cup R(\overline{y}_{(i+1)})$. However, $\gamma(a \cup e) < \gamma(b \cup e)$, i.e. the element, for which the value of γ is not minimal in U_i, generates the elements for which the value of γ is not minimal in U_{i+1}.

For calculating $\mathrm{Min}(U \cup R(\overline{b}))$ there is no need to calculate joints of all possible combinations of U and $R(\overline{b})$ elements. If $a \cap \overline{b} \neq 0$ then $\mathrm{Min}(\{a\} \cup R(\overline{b})) = \{a\}$. Owing to this, for arbitrary set U we have

$$\mathrm{Min}(U \cup R(\overline{b})) = \mathrm{Min}\Big(\mathrm{Min}\sum_{\substack{a \in U \\ a \cap \overline{b} = 0}} (\{a\} \cup R(\overline{b})) + \mathrm{Min}\sum_{\substack{a \in U \\ a \cap \overline{b} \neq 0}} \mathrm{Min}(\{a\} \cup R(\overline{b}))\Big) = \tag{19}$$

$$= \mathrm{Min}\Big(\mathrm{Min}\sum_{\substack{a \in U \\ a \cap \overline{b} = 0}} (\{a\} \cup R(\overline{b})) + \sum_{\substack{a \in U \\ a \cap \overline{b} \neq 0}} \{a\}\Big) = \mathrm{Min}\Big(\sum_{\substack{a \in U \\ a \leq b}} (\{a\} \cup R(\overline{b})) + \sum_{\substack{a \in U \\ a \cap \overline{b} \neq 0}} \{a\}\Big)$$

In particular, when calculating U_{i+1}, all such $a \in U_i$ can be chosen that $a \not\leq y_{i+1}$, these a will also be the elements of U_{i+1}. However, certain elements $U_i \cup R(\overline{y}_{(i+1)})$ can be greater than the above elements a.

If the simplified function is not isotone, then before using the described method of calculating prime implicants, to each sequence of values of variables a sequence of their negations should be added [14]. All elements of the set Z will then be incomparable*).

* If $\underline{f}$ denotes a set on which the function f is equivalent to 0, and if the function is not isotone, and transformation $\psi: x \to x,x$ is used, then $Z = \underline{f}_{\psi}$ /denotation as in [14]/. If f is isotone and only equivalents without negated variables are to be found, then this transformation is not needed, and $Z = \underline{f}$ can be taken.

If the function is isotone then, instead of Z, the set Max Z can be taken as input data according to the following equality

$$\operatorname{Min}\bigvee_{y\in Z} R(\bar{y}) = \operatorname{Min}\bigvee_{y\in \operatorname{Max} Z} R(\bar{y}) \qquad /20/$$

Indeed, for $b \leq c$ is $\operatorname{Min}(R(b) \cup R(c)) = R(b)$; hence following the equality /8/ we obtain /20/.

Example 1

Find $\operatorname{Min}\bigvee_{y\in Z} R(\bar{y})$ for

$Z = \{00011,\ 00110,\ 01100,\ 01001,\ 10010,\ 11000,\ 10101,\ 11011\}$ /isotone function/.

Max Z = {00110, 01100, 10101, 11011}

$\bar{y}_{(1)} =$ 11001

$U_1 = R(\bar{y}_{(1)})$: 10000
01000
00001

$\bar{y}_{(2)} =$ 10011

$U_2 :=$ 10000
00001
~~11000~~
01010
~~01001~~

$U_1 \cdot \{a: a \cap \bar{y}_{(2)} \neq 0\}$ /cancelled elements are not minimal/

$\bar{y}_{(3)} =$ 01010

$U_3 =$ 01010 $\quad U_2 \cdot \{a: a \cap \bar{y}_3 \neq 0\}$
11000
01001
10010
00011

$\bar{y}_{(4)} =$ 00100

$\operatorname{Min}\bigvee_{y\in Z} R(\bar{y}) = U_4 :$ 01110
11100
01101
10110
00111

Example 2

Find normal equivalents with the smallest number of letters for the function defined by the following table:

x_1	x_2	x_3	x_4	$f/x/$
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	1	1	1

For the remaining values of variables the function is not determined.

Let us add the negations of variables

$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_3$	$\bar{x}_4$	x_1	x_2	x_3	x_4	$f/x/$
1	1	0	0	0	0	1	1	0
1	0	0	1	0	1	1	0	0
0	1	1	0	1	0	0	1	0
0	0	1	1	1	1	0	0	0
1	0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0	1
0	0	0	0	1	1	1	1	1

In order to distinguish from the operation Min, the sequences, of variables $a \in U_i$ for which do not exist such x that $f(x) = 1$ and $a < x$ will be twice crossed. The sets U_i reduced in this way will be denoted by V_i.

The sequences passing unchanged from V_{i-1} to V_i are braced like in Example 1.

$\bar{y}_{(1)}$ = 00111100 $\bar{y}_{(2)}$ = 01101001 $\bar{y}_{(3)}$ = 10010110 $\bar{y}_{(4)}$ = 11000011

V_1 :	V_2 :	V_3 :	V_4 :
00100000	00100000	01010000	01010000
00010000	00001000	00000101	00000101
00001000	01010000	10100000	10100000
00000100	~~01000100~~	~~10001000~~	00001010
	~~00110000~~	~~00110000~~	~~10011000~~
	~~00100100~~	00011000	~~10100100~~
	~~00011000~~	00100100	~~10001100~~
	~~00001100~~	00001100	~~01011000~~
	~~00010001~~	~~00100010~~	~~01100100~~
	00000101	00001010	~~01001100~~
			~~00011010~~
			~~00100110~~
			~~00001110~~
			~~00011001~~
			~~00100101~~
			~~00001101~~

The Quine's Table ([13]) :

	01010000	00000101	10100000	00001010	F
10100101	1	0	0	1	0
01011010	0	1	1	0	0
00001111	1	0	1	0	0
	1	1	1	1	1

$\bar{y}'_{(1)}$ = 0110 $\bar{y}'_{(2)}$ = 1001 $\bar{y}'_{(3)}$ = 0101

U'_1 :	U'_2 :	U'_3 :
0100	1100	1100
0010	1010	0101
	0101	0011
	0011	~~1110~~

Elements U_3' determine the solution. Thus

$$f\,/x/ = \bar{x}_2\,\bar{x}_4 + x_2\,x_4$$

or $$f\,/x/ = x_2\,x_4 + x_1\,x_3$$

or $$f\,/x/ = \bar{x}_1\,\bar{x}_3 + x_1\,x_3$$

Conclusion

The described method of determination of prime implicants has been elaborated for computations performed in a binary digital computer. These computations may be reduced to the performance of appropriately ordered simple subroutines executing:

1. decomposition of sequences into sequences with a single 1 /R operation/,
2. computing of alternatives of words belonging to various sets /$K \cup L$ operation/,
3. data reduction by means of Min operation.

If necessary, subroutines, executing an additional reduction of intermediate results according to principles given in this paper, may be added to the above subroutines. As pointed out, this method permits to perform computations in various ways, for instance, more or less often using the Min operation for sets of intermediate results, with or without an additional reduction, and so on.

Precise algorithms of computation and data preparing depend upon the properties of the digital computer used, as well as of its language. Apart from this, the algorithm may be changed according to the complexity of the simplified function.

Acknowledgements

The author wishes to thank Dr. A.W. Mostowski and Mr. Z. Juszczyk for their valuable remarks and suggestions.

References

1. BIRKHOFF G.: Lattice Theory, New York 1948.

2. BUTLER K.J.Jr., WARFIELD J.N.: A Digital Computer Program for Reducing Logical Statements to a Minimal Form, Proc. Natl. Electronics Conf. 1959:15, 456-466.

3. GAVRILOV M.A.: Minimizacija bulevych funkcij charakterizujuščich relejnyje cepi, Avtomatika i Telemechanika, 1959:20, 1217-1238.

4. HARRIS B.: An Algorithm for Determining Minimal Representations of a Logic Function, IRE Trans., 1957:EC-6, 103-108.

5. KARNAUGH M.: The Map Method for Synthesis of Combinational Logic Circuits Commun. and Electronics, Trans. AIEE I, 72.

6. Mc CLUSKEY E.J.: Minimization of Boolean Functions, Bell System Technical Journal, 1953:35, 1417-1444.

7. MOTT T.H.Jr.: Determination of the Irredundant Normal Forms of a Truth Function by Iterated Consensus of the Prime Implicants, IRE Trans., 1960:EC-9, 245-252.

8. NELSON R.J.: Weak Simplest Normal Truth Functions, Journal of Symbolic Logic, 1955:20, 232-234.

9. QUINE W.V.: The Problem of Simplifying Truth Functions, Amer. Math. Monthly, 1952:59, 521-531.

10. QUINE W.V.: A Way to Simplify Truth Functions, Amer. Math. Monthly, 1955:62, 627-631.

11. QUINE W.V.: On Cores and Prime Implicants of Truth Functions, Amer. Math. Monthly, 1959:66, 755-760.

12. VOJSVILLO E.K.: Metod uproščenija form vyraženija funkcij istinnosti, Naučnye Doklady Vysšej Školy Filosofskie Nauki, 1958:2, 120-135.

13. WALIGÓRSKI S.: Calculation of the Quine's Table for Truth Functions, Prace ZAM PAN, 1961:2, A15.

14. WALIGÓRSKI S.: On Normal Equivalents of Truth Functions, Algorytmy, 1962:1, 1, 73-96.

Instytut Maszyn Matematycznych
Algorytmy Nº 2

517.11

ON SUPERPOSITIONS OF ZERO-ONE FUNCTIONS

by Stanisław WALIGÓRSKI

Received October 1962

The paper deals with the problem what are the zero-one functions f and g respectively determined on partly ordered sets P and Q for which exists such isotone mapping $h:P \to Q$ that for $x \in P$ we obtain $f(x) = g(h(x))$. It is shown that the existence of function h depends on variations of functions f and g defined in this paper. The solution of this problem is useful for the synthesis of multi-level switching circuits.

Introduction

When designing switching circuits one may face the following problem: is it possible to present the given zero-one function of N variables $f(x_1, x_2, \ldots, x_N)$ /where variables x_i can take the value 0 or 1/ in the form of a superposition of a certain fixed zero-one function $g(y_1, y_2, \ldots, y_M)$/ , and M isotone zero-one functions $y_j = h_j(x_1, x_2 \ldots, x_N)$.

This problem arises in practice when, for instance, a circuit is required that realizes a certain zero-one function in the form of a network made of 'and' and 'or' gates, the outputs of which are associated with the inputs of an element /or, more generally, of a network/ realizing a certain given function. Such an element may be, for instance, a flip-flop. If the structure of the element in question /or network/ is more complex than elements 'and' and 'or' , then from the viewpoint of the system simplicity it is

advantageous to design its parts in the form of a network of the described type.

The discussed problem, concerning the possibility to present a function in the form of the mentioned superposition, is partly solved in [1] when g is a symmetrical difference or an equivalence. In this paper the method of Karnaugh's maps is used. Another particular case of solving this problem is given in [4], where g is a function realized by flip-flop

$$g(y_1, y_2, y_3) = y_1 \cdot \bar{y}_2 + y_3 .$$

The presented method is a generalization of the results given in [4].

Assume that the set of values of the sequence of function arguments $x_1, x_2, \ldots, x_N$ is partly ordered in the following way /see [4], [5]/ : $a_1, a_2, \ldots, a_N \leqslant b_1, b_2, \ldots, b_N$ if and only if $a_i \leqslant b_i$ for $i = 1, 2, \ldots, N$. The function that is realized by the switching circuit with N inputs, can be determined for all or only for some values of such a sequence; more generally, it is determined on a partly ordered set. Thus, the problem formulated at the very beginning may be generalized as follows: what are the conditions for the existence of isotone function h such that $f(x) = g(h(x))$ where $x \in P$ and the functions f and g are respectively determined on partly ordered sets P and Q.

The values 0 and 1 of functions f and g will be treated below as numbers - therefore, arithmetic operations can be performed on them.

Definition

Let P be a partly ordered set, and let f be a real function determined on P and taking only the values 0 or 1 .

We define a variation of function f on P :

$$W(f;P) = \sup_{C} \sum_{i=1}^{d[C]} |f(x_i) - f(x_{i-1})| \qquad /1/$$

where the upper bound is taken on the set of all finite chains

$$C : x_0 < x_1 < x_2 < \ldots < x_{d[C]}$$

contained in P.

When every non empty chain $C \subset P$ contains only one element then for every function f determined on P we take $W(f;P) = 0$.

If $W(f;P)$ is a finite number, then the function f is called a function with finite variation.

Particularly, if P is a finite chain

$$x_0 < x_1 < \ldots < x_k$$

then every zero-one function determined on P is a function with finite variation and

$$W(f;P) = \sum_{i=1}^{K} |f(x_i) - f(x_{i-1})| \; . \qquad /2/$$

Theorem 1

If zero-one function g determined on partly ordered set Q is a finite variation function, and if function h is isotone $h \in Q^P$ /according to denotations [2]/, and

$$f(x) = g(h(x)) \qquad \text{for } x \in P$$

then

$$W(f;P) \leq W(g;Q) \; .$$

Proof

Let us take the finite chain $C \subset P : x_1 < x_2 < \ldots < x_{d[C]}$.

The magnitude of the function variation on C is determined only by such pairs of elements of the chain which cover one another and for which values of the function are different. $g(x_{i+1}) \neq g(x_i)$ if and only if $f(x_{i+1}) \neq f(x_i)$. Since h is isotone, then $h(x_{i+1})$ covers $h(x_i)$ in $h(C)$. Therefore

$$W(f; C) \leq W(g; h(C)) \qquad /3/$$

$$W(f;P) = \sup_{C \subset P} W(f;C) \leq \sup_{C \subset P} W(g; h(C)) \leq W(g;Q) \qquad /4/$$

Theorem 2

If zero–one functions f and g are respectively determined on partly ordered sets P and Q, f is a finite variation function, and $W(f;P) < W(g;Q)$; then such isotone function $h \in Q^P$ exists that

$$f(x) = g(h(x)) \qquad \text{for } x \in P \qquad /5/$$

Proof

f is a finite variation function; this implies the existence of the chain $D \in Q : y_0 < y_1 < \ldots < y_{d[D]}$ such that $d[D] = W(g;D) = W(f;P) + 1$.

$g(y_i) \neq g(y_{i+1})$, thus $g(y)$ has equal values on the elements of chain D with equal parity indices.

Let

$$m(x) = \{z: \ z \leq x\} \qquad /6/$$

$$a(x) = (f(x) - g(y_0) + W(f; m(x))) \bmod 2 \qquad /7/$$

$$h(x) = y_{W(f; m(x))} + a(x) \ . \qquad /8/$$

Therefore

$$a(x) + W(f;m(x)) \equiv (f(x) - g(y_o)) \mod 2 \qquad /9/$$

and

$$g(h(x)) = g(y_{|f(x) - g(y_o)|}) . \qquad /10/$$

However, it is readily verified that we also have *)

$$f(x) = g(y_{|f(x) - g(y_o)|}) . \qquad /11/$$

Hence equality /5/ is satisfied.

To show that h is isotone it is sufficient to prove that $W(f; m(x)) + a(x)$ is isotone in the sense of a simple ordering of natural numbers. Assume that it is not true and that there exist $x,z \in P$ such that $x < z$ and

$$W(f; m(z)) + a(z) < W(f; m(x)) + a(x) . \qquad /12/$$

From the definition of the variation it follows that

$$W(f; m(z)) \geq W(f; m(x)) . \qquad /13/$$

$a \leq 1$ and /13/ imply that inequality /12/ can be satisfied only if

$$W(f; m(z)) = W(f; m(x)) . \qquad /14/$$

Since $x < z$, it is possible only if $f(x) = f(z)$. Hence, $a(x) = a(z)$ and inequality /12/ is false, i.e. for $x < z$ we have

$$W(f; m(x)) + a(x) \leq W(f; m(z)) + a(z) .$$

Theorem 3

If zero-one functions f and g are respectively determined on partly ordered sets P and Q, and if there exists such b that

---

* Formula /11/ can be checked by substitution: if $f(x) = g(y_o)$ then $g(y_{|f(x) - g(y_o)|}) = g(y_o)$ and similarly for $f(x) \neq g(y_o)$.

1. for every finite chain $C \subset P$ such that $W(f; C) = W(f; P)$ we have $f(\min_{x \in C} x) = b$;
2. there exists the finite chain $D \subset Q$ such that $W(g;D) = W(f;P)$ and $g(\min_{y \in D} y) = b$,

then there exists the isotone function $h \in Q^P$ such that

$$f(x) = g(h(x)) \qquad \text{for } x \in P .$$

Proof

Chain $D: y_0 < y_1 < \ldots < y_{d[D]}$ exists such that $g(y_0) = b$, and $d[D] = W(g;D) = W(f;P)$.

Function h is to be determined as previously. It should only be proved that

$$W(f; m(x)) + a(x) \leq d[D]$$

i.e. if $W(f; m(x)) = W(f;P)$ then $a(x) = 0$. However, x is then the maximal element of the chain C such that $W(f;C) = W(f;P)$.

$f(\min_{z \in C} z) = b$ and, therefore, $f(x) \equiv (b + W(f;P)) \bmod 2$. Hence $a(x) \equiv b + W(f;P) - b + W(f;P) \bmod 2 = 0$.

Corollary

Theorem 3 remains true if it is formulated so that 'min' is replaced by 'max' . If, however, the assumptions of Theorem 3 are satisfied, then the assumptions modified in the above way are also satisfied, and conversely.

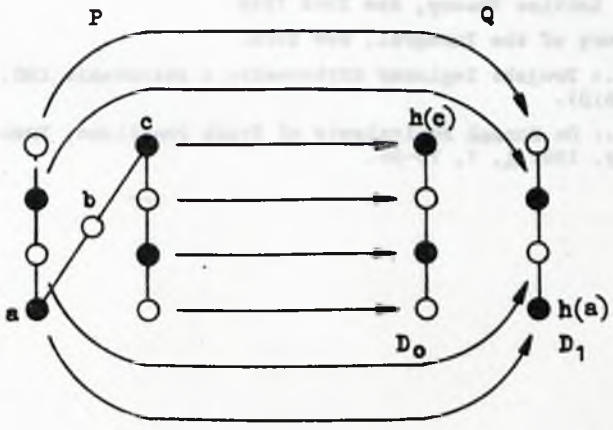
If assumption 2 of Theorem 3 /or dual theorem/ is not satisfied, then the sought function h does not exist.

The proof follows from formula /3/ .

If assumption 1 of Theorem 3 is not satisfied, i.e. there

exist chains C_0, $C_1 \subset P$, $f(\min_{x \in C_j} x) = j$, $W(f,Cj) = W(f;P)$ and for every chain $D \subset Q$ the equality $W(g;D) = W(f;P)$ implies $g(\min_{x \in D} x) = b$. then the function h also does not exist. The existenoe of the chains D_0, $D_1 \subset Q$ such that $f(\min_{y \in D_j} y) = j$, $W(g;D) = W(f;P)$ does not imply, however, the existence of function h. In the example given below the function h does not exist. The contrary example may be easily found by the reader.

On Hass's diagrams of partly ordered sets P and Q we blacken the points in which the functions f and g take the value 1 respectively. Arrows connect such points x, y that $y = h(x)$, for which the function h is determined uniquely.



It must be $h(a) < h(b) < h(c)$. However, $h(a) \in D_1$, $h(c) \in D_0$, and the elements of D_0 and of D_1 are incomparable. h(b) can be neither the element of D_0 nor of D_1, hence the sought isotone function h does not exist.

Conclusion

The problems handled in the present paper were elaborated in 1958-60 while networks with flip-flops have been designed. The described method for comparing the variations of zero-one functions permitted to evaluate the minimum of flip-flops required in the designed systems, and consequently to design the network as economically as possible.

References

1. MUKHOPADHYAY A.: Detection of Disjuncts of Switching Functions and Multilevel Circuit Design, J. of Electronics and Control, 1961:10,1, 45-56.
2. BIRKHOFF G.: Lattice Theory, New York 1948.
3. SAKS S.: Theory of the Integral, New York.
4. WALIGÓRSKI S.: Projekt logiczny arytmometru i sterowania ABC, Biuletyn ZAM, 1958:D1.
5. WALIGÓRSKI S.: On Normal Equivalents of Truth Functions, Prace IMM, Algorytmy, 1962:1, 1, 73-96.