ALGORYTMY, V, N° 10 /1969/, 5-19

MASZYNY PROGRAMOWANE

Zdzisław PAWLAK

Pracę złożono 18.4.1968 r.

Celem pracy jest sprecyzowanie podstawowych pojęć z zakresu maszyn matematycznych, takich jak pamięć maszyny, obliczenie, program itp. Punktem wyjścia pracy jest pojęcie procesu obliczania . Zdefiniowano maszynę jako funkcję częściową, której argumentami i wartościami są stany pamięci maszyny. Udowodniono pewne elementarne własności maszyn oraz wskazano na dalsze problemy związane z ich teorią.

Dla stworzenia teorii maszyn matematycznych przydatnej do formułowania i rozwiązywania rzeczywistych problemów prawdziwych wych maszyn matematycznych konieczne jest sprecyzowanie podstawowych pojęć takich jak, pamięć maszyny, maszyna cyfrowa, obliczenie, rozkaz, program etc.

Celem niniejszej notatki jest właśnie próba sprecyzowania tych pojęć. Punktem wyjścia jest pojęcie procesu obliczania. Pojęcie to pozwala na zdefiniowanie maszyny, jako funkcji częściowej, której argumentami i wartościami są stany pamięci maszyny. Taka definicja maszyny jest zgodna z intuicjami technicznymi, z drugiej zaś strony pozwala na łatwe formalne definiowanie i badanie różnych maszyn. Zdefiniowanie bowiem każdej maszyny sprowadza się do określenia pewnej funkcji, zaś badanie własności maszyn sprowadza się do badania odpowiednich własności funkcji. Przyjęta definicja jest na tyle ogólna, że

mieszczą się w niej wszystkie znane pojęcia maszyny jak np. maszyny Turinga, automaty skończone itp., jednakże tego rodzaju maszynami nie zajmujemy się tutaj: interesować nas będą takie maszyny, które są możliwie bliskie rzeczywistym maszynom matematycznym.

1. Procesy

Niech $\pi : T \to T$ będzie funkcją częściową. Ciąg $t_0, t_1, \ldots$ taki, że $t_i \in T$ dla każdego i, $0 \leqslant i$, $t_{i+1} = \pi(t_i)$ nazwiemy procesem[1] nieskończonym, t_i nazwiemy stanami procesu, zaś π funkcją przejścia procesu. Ciąg $t_0, t_1, \ldots, t_k$ nazwiemy procesem skończonym, jeżeli $t_i \in T$ oraz dla każdego i, $t_{i+1} = \pi(t_i)$, natomiast $t_k \notin D_\pi$, gdzie D_π oznacza dziedzinę funkcji π. t_0 nazwiemy stanem początkowym procesu, zaś t_k stanem końcowym. Proces ze stanem początkowym t_0 i funkcją przejścia π oznaczać będziemy przez $P_\pi(t_0)$.

Lemat 1. Jeżeli w procesie $P_\pi(t)$ istnieją dwa stany jednakowe tj. $(t_i = t_j,\ i \neq j)$, to dla każdego $k \geqslant 0$, $t_{i+k} = t_{j+k}$.

Dowód. Prawdziwość lematu 1 wykażemy przez indukcję względem k. Z założenia lemat 1 jest prawdziwy dla $k = 0$. Załóżmy, że lemat 1 jest prawdziwy dla $k = n$, tj. $t_{j+n} = t_{i+n}$. Wobec tego $\pi(t_{i+n}) = \pi(t_{j+n})$. Ponieważ $\pi(t_{i+n}) = t_{i+n+1}$, zaś $\pi(t_{j+n}) = t_{j+n+1}$, więc $t_{i+n+1} = t_{j+n+1}$, co kończy dowód.

Twierdzenie 1. Jeżeli proces $P_\pi(t_0) = t_0, t_1, \ldots, t_r$ jest skończony, to dla każdego i,j $(0 \leqslant i,j \leqslant r)$, $t_i \neq t_j$.

Dowód. Udowodnimy przez indukcję twierdzenie przeciwstawne, tj. jeżeli w procesie $P_\pi(t_0)$ istnieją takie i,j, że $i < j$ oraz $t_i = t_j$, to proces $P_\pi(t_0)$ nie jest skończony.

[1] Proces ten można uważać za szczególny przypadek procesu Markowa.

Jeżeli proces $P_\pi(t_0)$ jest skończony, to istnieje takie k, że $t_{j+k} \notin D_\pi$ i odwrotnie, jeżeli proces $P_\pi(t_0)$ jest nieskończony, to nie istnieje takie k, że $t_{j+k} \notin D_\pi$.

Pokażemy przez indukcję względem k, że dla każdego $k > 0$, $t_{j+k} \in D_\pi$. Jeżeli $k = 0$, to z założenia $t_i = t_j$. Ponieważ $t_i \in D_\pi$, więc i $t_j \in D_\pi$. Przyjmijmy, że $t_{j+k} \in D_\pi$ dla pewnego $k > 0$. Pokażemy, że $t_{j+k+1} \in D_\pi$.

Na podstawie lematu 1 jest $t_{j+k} = t_{i+k}$. Wobec tego $\pi(t_{j+k}) = \pi(t_{i+k})$. Ponieważ $\pi(t_{j+k}) = t_{j+k+1}$ oraz $\pi(t_{i+k}) = t_{i+k+1}$, więc $t_{j+k+1} = t_{i+k+1}$. Jednakże $t_{i+k+1} \in D_\pi$ na podstawie definicji procesu, więc $t_{j+k+1} \in D_\pi$ - co kończy dowód.

Proces $P_\pi(t_0)$ nazwiemy cyklicznym, jeżeli istnieją takie liczby p,r, że dla wszystkich $i \geqslant r$, $t_i = t_{i+p}$; najmniejszą liczbę p taką, że dla każdego $i \geqslant r$, $t_i = t_{i+p}$, nazwiemy cyklem procesu $P_\pi(t_0)$.

Z lematu 1 i twierdzenia 1 wynika

Wniosek 1. Jeżeli w procesie $P_\pi(t)$ istnieją takie i,j, że $i \neq j$ oraz $t_i = t_j$, to $f_\pi(t)$ jest cykliczny.

Jeżeli istnieją takie n,t, że

$$\underbrace{\pi \ldots \pi}_{n}(t) = t,$$

to π nazwiemy funkcją cykliczną, a liczbę n cyklem funkcji π i zapiszemy $\pi^n(t) = t$.

Jeżeli w procesie $P_\pi(t)$ dla każdego i,j, $t_i \neq t_j$, gdy $i \neq j$, to proces $P_\pi(t)$ nazwiemy różnowartościowym.

Twierdzenie 2. Dla wszystkich t proces $P_\pi(t)$ jest różnowartościowy wtedy i tylko wtedy, gdy jego funkcja przejścia π nie jest cykliczna.

Dowód. Pokażemy najpierw, że jeżeli dla każdego t proces $P_\pi(t)$ jest różnowartościowy, to funkcja przejścia π nie jest cykliczna. Jeżeli bowiem funkcja π jest cykliczna, to z definicji istnieją takie t i n, że $\pi^n(t) = t$, a więc istnieje proces $t, \pi(t), \pi^2(t), \ldots, \pi^n(t)$, w którym $t = \pi^n(t)$, co przeczy założeniu.

Pokażemy teraz, że jeżeli funkcja przejścia nie jest cykliczna, to dla każdego t proces $P_\pi(t)$ jest różnowartościowy. Załóżmy, że proces $P_\pi(t)$ nie jest różnowartościowy. Wtedy $P_\pi(t)$ zawiera dwa wyrazy jednakowe, a więc funkcja przejścia π jest cykliczna, co przeczy założeniu.

Twierdzenie 3. Jeżeli dla dowolnych $P_\pi(t_0)$ i $P_\pi(t'_0)$ istnieją takie i,j, że $t_i = t'_j$, to dla każdego $k > 0$ $t_{i+k} = t'_{j+k}$.

Dowód. Twierdzenie to udowodnimy przez indukcję.

Jeżeli $k = 0$, to twierdzenie jest prawdziwe na podstawie założenia.

Załóżmy, że twierdzenie 3 jest prawdziwe dla $k = n\ (n > 0)$, tj. $t_{i+n} = t'_{j+n}$. Pokażemy, że jest ono prawdziwe dla $k = n+1$. Ponieważ $t'_{j+n+1} = \pi(t_{j+n})$ oraz $t_{i+n+1} = \pi(t_{i+n})$, zaś na podstawie założenia indukcyjnego mamy $t_{i+n} = t'_{j+n}$, co należało pokazać.

Z twierdzenia 3 wynikają następujące wnioski:

Wniosek 2. Jeżeli procesy $P_\pi(t_0) = t_0, t_1, \ldots, t_k$ oraz $P_\pi(t'_0) = t'_0, t'_1, \ldots, t'_p$ są skończone oraz istnieją takie i,j, że $i \neq j$ oraz $t_i = t'_j$, to $t_k = t'_p$ oraz $k-i = p-j$. Procesy $P_\pi(t_0)$ i $P_\pi(t'_0)$ są równe $(P_\pi(t_0) = P_\pi(t'_0))$, jeżeli dla każdego i, $t_i = t'_i$.

Wniosek 3. Jeżeli $t_0 = t'_0$, to $P_\pi(t_0) = P_\pi(t'_0)$.

2. Ogólne pojęcie maszyny

Powiemy, że stany t i t' T są w relacji M_π wtedy i tylko wtedy, gdy istnieje skończony proces $P_\pi = t_0, t_1, \ldots, t_k$ taki, że $t_0 = t$ i $t_k = t'$.

Twierdzenie 4. Relacja M_π jest funkcją.

Dowód. Prawdziwość tego twierdzenia wynika bezpośrednio z wniosku 2.

Każdą funkcję M_π będziemy nazywać maszyną.

Zbiór T nazwiemy zbiorem stanów albo pamięcią maszyny M_π; zbiór $T' = T - D_\pi$ - zbiorem stanów końcowych maszyny M_π, zaś funkcję π - funkcją przejścia albo sterowania maszyny M_π.

Proces $P_\pi(t)$ nazwiemy obliczeniem w maszynie M_π. Jeżeli ciąg $t_0, t_1, \ldots$ jest obliczeniem w maszynie M, to zapiszemy $(t_0, t_1, \ldots) \in M$.

Z definicji maszyny wynika, że każdą maszynę M możemy przedstawić w postaci:

$$M_\pi(t) = h(t, k_\mu),$$

gdzie

$$h(t,0) = t$$

$$h(t,k+1) = \pi(h(t,k)),$$

zaś k_μ - oznacza najmniejsze takie k, że

$$h(t,k) \notin D_\pi .$$

Niech M_1 i M_2 oznaczają odpowiednie maszyny o funkcjach przejścia π_1 i π_2 oraz pamięciach T_1 i T_2. Powiemy, że maszyna M_1 jest φ - naśladowana /symulowana/ przez maszynę M_2 (symbolicznie: $M_1 \overset{\subset}{\underset{\varphi}{\leq}} M_2$), wtedy i tylko wtedy, gdy istnieje takie wzajemnie jednoznaczne odwzorowanie $\varphi : T_1 \longrightarrow T_2$, że

$$1^\circ \quad \bigwedge_{t \in D_{M_1}} \varphi M_1(t) = M_2(\varphi(t)) \quad ,$$

$$2^{o} \quad \bigwedge_{t \in D_{\pi_1}} \bigvee_{k \geq 1} \varphi_{\pi_1}(t) = \pi \frac{k}{2}\left(\varphi(t)\right)$$

Jeżeli $M_1 \overset{\subset}{\underset{\varphi}{\leq}} M_2$ oraz $M_2 \overset{\subset}{\underset{\varphi^{-1}}{\leq}} M_1$, to powiemy, że maszyny M_1 i M_2 są izomorficzne i zapiszemy $M_1 \underset{\varphi}{\sim} M_2$.

Powiemy, że maszyna M_π oblicza funkcję $f : X \longrightarrow X$ wtedy i tylko wtedy, gdy istnieją takie odwzorowania $\varkappa : X \to T$ oraz $\delta : T \longrightarrow X$, że dla każdego $x \in X$, $f(x) = \delta\left(M_\pi(\varkappa(x))\right)$.

Jeżeli M_π oblicza funkcję f przy ustalonym δ i $\varkappa$ to powiemy, że f jest obliczalna przez maszynę M_π i zapiszemy f_{M_π} lub krócej f_M (zakładając, że $\delta, \varkappa, \pi$ są ustalone i znane).

Niech M_1 oznacza maszynę M_{π_1}, zaś M_2 - maszynę M_{π_2}. Powiemy, że maszyny M_1 i M_2 są równoważne, symbolicznie $M_1 \equiv M_2$ wtedy i tylko wtedy, gdy $f_{M_1} = f_{M_2}$.

Bezpośrednio z definicji zawierania i równoważności maszyn wynikają następujące wnioski:

<u>Wniosek 4.</u> Jeżeli $M_1 \overset{\subset}{\underset{\varphi}{\leq}} M_2$, to

1^{o}. Dla każdego obliczenia $P_1(t_0) = t_0, \ldots, t_n$ w M_1 istnieje obliczenie $P_2(t_0') = t_0', \ldots, t_m'$ w M_2 takie, że:

$$\bigwedge_{0 \leq i \leq n} \bigwedge_{i \leq j \leq m} \left[t_j' = \varphi(t_i) \,\&\, t_0' = \varphi(t_0) \,\&\, t_m' = \varphi(t_n)\right],$$

$$2^{o}. \quad \bigwedge_{t \in D_{M_1}} d\left(P_1(t)\right) \leq d\left(P_2(\varphi(t))\right),$$

gdzie $d(P_1(t))$ oznacza długość obliczenia $P_1(t)$.

Jeżeli obliczenie jest nieskończone, to nie zachodzi trzeci człon koniunkcji.

<u>Wniosek 5.</u> Jeżeli $M_1 \underset{\varphi}{\sim} M_2$, to

1^{o}. Dla każdego obliczenia $P_1(t_0) = t_0, \ldots, t_n$ w M_1 istnieje obliczenie $P_2(t_0') = t_0', \ldots, t_n'$ w M_2 takie, że

$$\bigwedge_{0 \leq i \leq n} \left[t_i' = \varphi(t_i)\right],$$

$$2^{o}. \quad \bigwedge_{t \in D_{M_1}} d\left(P_1(t)\right) = d\left(P_2(\varphi(t))\right).$$

3. Maszyny programowane

Określimy najpierw pamięć maszyn programowanych.

Niech Σ i A będą dowolnymi zbiorami takimi, że $\Sigma \cap A \neq 0$. Σ będziemy nazywać alfabetem, zaś A zbiorem adresów. Zbiór funkcji $C \subset \Sigma^A$ nazwiemy pamięcią adresową. Ponieważ w dalszym ciągu będziemy zajmować się tylko pamięciami adresowymi, zamiast "pamięć adresowa" będziemy używali zwrotu "pamięć". Każdą funkcję $c \in C$ będziemy nazywać zawartością pamięci. Przyjmiemy w dalszym ciągu, że pamięć spełnia następujące warunki:

W 1. $\bigwedge_{c \in C}$ Dziedzina D_c jest skończona.

W 2. W zbiorze adresów A istnieje wyróżniony element l, zwany licznikiem rozkazów taki, że $\bigwedge_{c \in C} l \in D_c$.

W ten sposób określimy pamięć maszyn programowanych.

Zanim określimy sterowanie tego rodzaju maszyn, wprowadzimy najpierw kilka pojęć pomocniczych. Każdą funkcję postaci:

$$r : A^n \times C \longrightarrow C, \quad n \geq 0$$

nazwiemy schematem instrukcji n adresowej w pamięci C. Jeżeli w schemacie instrukcji adresy traktować będziemy jako

parametry, to funkcję

$$r_{\bar{a}_n} : C \longrightarrow C , \quad \bar{a}_n \in A^n$$

nazwiemy instrukcją n adresową w pamięci C . Instrukcje będziemy krótko oznaczali przez r^* .

Przyjmiemy w dalszym ciągu, że instrukcje spełniają następujący warunek:

W 3. $$\bigwedge_{r^*} \bigwedge_{c \in C} \bigvee_{a \in A} r^*(c) /A- l \cup a = c/A- l \cup a .$$

Znaczy to, że każda instrukcja może zmienić zawartość pamięci co najwyżej w dwu adresach l i a.

Niech $R_C^* \subset R_C$, gdzie R_C jest zbiorem wszystkich instrukcji w pamięci C , zaś $\varphi : \sum \longrightarrow R_C^*$ niech będzie wzajemnie jednoznaczną funkcją, zwaną w dalszym ciągu kodowaniem. Każdą maszynę posiadającą pamięć C spełniającą warunki W 1 i W 2 oraz funkcję przejścia określoną jak niżej

$$\bigwedge_{c \in C} c' = [\varphi(c^*(1))](c) ,$$

gdzie $c^*(1) = c(c(1))$,

nazwiemy maszyną programowaną.

Przykład. Rozpatrzymy maszynę trójadresową o pamięci $C = \sum^A$, gdzie $A = 1 \cup N$, zaś $N = 0,1,2,\ldots$ o następującym zbiorze schematów instrukcji :

$+(\bar{a}_3)$, $-(\bar{a}_3)$, $\cdot(\bar{a}_3)$, $/(\bar{a}_3)$, $!(\bar{a}_3)$, $?(\bar{a}_3)$, $\mathrm{stop}(\bar{a}_3)$,

gdzie $\bar{a}_3 = a_1 a_2 a_3$, $a_i \in N$.

Schematy te są określane następująco:

$$c' = [+(\bar{a}_3)](c) ,$$

gdzie

$$c'(x) = \begin{cases} c(a_1) + c(a_2), & \text{gdy } x = a_3,\ x \in A \\ c(1) + 1, & \text{gdy } x = 1 \\ c(x), & \text{gdy } x \neq a_3 \text{ i } x \neq 1 . \end{cases}$$

Podobnie możemy określić instrukcje odejmowania, mnożenia i dzielenia. Instrukcję $!(\bar{a}_3)$ nazwiemy przejściem bezwarunkowym i określimy ją następująco:

$$c' = [!(\bar{a}_3)](c) , \text{ gdzie}$$

$$c'(x) = \begin{cases} c(a_3) , & \text{gdy } x = 1 \\ c(x) , & \text{gdy } x \neq 1. \end{cases}$$

Instrukcję $?(\bar{a}_3)$ zwaną przejściem warunkowym zdefiniujemy jak niżej:

$$c' = [?(\bar{a}_3)](c) , \text{ gdzie}$$

$$c'(x) = \begin{cases} c(1) + 1 , & \text{gdy } x = 1 \text{ oraz } c(a_1) = 0 , \\ c(a_3) , & \text{gdy } x = 1 \text{ oraz } c(a_1) \neq 0, \\ c(x) , & \text{gdy } x \neq 1. \end{cases}$$

Instrukcja $\mathrm{stop}(\bar{a}_3)$ ma postać

$$c' = [\mathrm{stop}(\bar{a}_3)](c)$$

i jest nieokreślona dla żadnego $c \in C$.

Dla pełnego określenia maszyny musimy podać jeszcze kodowanie φ . Gdy zbiór instrukcji jest nieskończony, możemy przyjąć tu Gödelowską zasadę numeracji instrukcji. Dla skończonego zbioru adresów sprawa jest o wiele prostsza; wystarczy symbolom

+ , - , . , / , ! , ? , stop przyporządkować numery od 1 do 7. Numerem instrukcji d_1, a_1, a_2 a_3 będzie liczba, której cyfry rozwinięcia dziesiętnego są kolejnymi cyframi liczb d_1, a_1, a_2, a_3 d_1 jest numerem symbolu +, - , . , / , ? , ! , stop - uzupełniając odpowiednio liczby a_1, a_2, a_3 zerami, tak aby maksymalna ilość pozycji przeznaczonych na jeden adres pozwalała na zapisanie każdego adresu.

Jeżeli maszyna M spełnia warunek

$$\text{W 4.} \quad \bigwedge_{(c_0,c_1,\ldots)\in M} \bigwedge_{0\leqslant i,j} \left[c_i(1) = c_j(1) \longrightarrow c_i^*(1) = c_j^*(1) \right], \quad i \neq j,$$

to maszynę M nazwiemy stałoprogramową.

Jeżeli maszyna M spełnia warunek

$$\text{W 5.} \quad \bigvee_{(c_0,c_1,\ldots)\in M} \bigvee_{0\leqslant i,j} \left[c_i(1) = c_j(1) \,\&\, c_i^*(1) \neq c_j^*(1) \right], \quad i \neq j,$$

to maszynę M nazwiemy zmiennoprogramową.

Niech K_s i K_z oznaczają odpowiednio klasy maszyn stało- i zmiennoprogramowych. Powiemy, że klasa K_s jest zawarta w klasie K_z wtedy i tylko wtedy, gdy

$$1) \quad \bigwedge_{M\in K_s} \bigvee_{M'\in K_z} \bigvee_{\varphi} \left(M \underset{\varphi}{\subset} M' \right).$$

Przyjmiemy, że dla maszyn programowanych odwzorowanie φ spełnia następujące warunki:

$$1. \quad \bigwedge_{C,C'} C(x) = C'(x) \Leftrightarrow \varphi C\left(\varphi'(x)\right) = \varphi C'\left(\varphi'(x)\right),$$

gdzie $\varphi' : A \to A'$ jest wzajemnie jednoznacznym odwzorowaniem zbioru adresów pamięci maszyn M i M',

2. $1' = \varphi'(1)$, gdzie 1 i 1' są odpowiednio licznikami rozkazów w maszynach M i M'.*)

Twierdzenie 5. Klasa maszyn stałoprogramowych jest istotnie zawarta w klasie maszyn zmiennoprogramowych.

Dowód. Pokażemy najpierw, że dla każdej maszyny stałoprogramowej istnieje maszyna zmiennoprogramowa spełniająca warunek 1).

Niech M_3 będzie maszyną stałoprogramową z pamięcią C, zbiorem instrukcji R_s^* oraz funkcją przejścia π_s, zaś M_2 niech będzie maszyną zmiennoprogramową o pamięci C, zbiorze instrukcji R_z^* takim że $R_z^* \supset R_s^*$ i funkcji przejścia

$$\pi_z = \varphi_z\left(c\left(c\left(1\right)\right)\right),$$

gdzie

$$\varphi_z : \Sigma \longrightarrow R_z^*$$

takiej, że jeżeli $\varphi(a) \in R^*$, to $\varphi(a) = \varphi_z(a)$.

Maszyna M_s jest wtedy oczywiście zawarta w maszynie M_z.

Pokażemy teraz, że jeżeli M_z jest maszyną zmiennoprogramową, to nie istnieje maszyna stałoprogramowa M_s taka, że $M_z \underset{\varphi}{\subset} M_s$. Mówiąc inaczej, jeżeli M_z jest maszyną zmiennoprogramową oraz $M_z \underset{\varphi}{\subset} M'$, to M' jest też maszyną zmiennoprogramową dla dowolnych maszyn M_z, M' i odwzorowania φ.

Ponieważ maszyna M_z jest zmiennoprogramowa, możemy więc napisać

$$1) \quad \bigvee_{c_0,c_1,\ldots\in M_z} \bigvee_{i,j} \left[c_i(1) = c_j(1) \,\&\, c_i^*(1) \neq c_j^*(1) \right].$$

Na podstawie założenia, że $M_z \underset{\varphi}{\subset} M'$, wniosku 4 warunków 1 i 2 spełnianych przez odwzorowanie φ - 1) możemy przepisać w postaci

*) Na konieczność umieszczenia tych warunków zwrócili mi uwagę dr G. Rozenberg oraz mgr Z. Sozańska.

2) $$\bigvee_{c'_0, c'_1, \ldots \in M'} \bigvee_{p,q} \left[c'_p(1') = c'_q(1') \wedge c'^{*}_p(1') \neq c'^{*}_q(1'), \right]$$

$$p \neq q$$

gdzie

$$c'_p = \varphi(c_i) \quad \text{oraz} \quad c'_q = \varphi(c_j) \quad \text{zaś} \quad 1' = \varphi'(1) .$$

A więc M' jest zmiennoprogramowa.

W podobny sposób można udowodnić wiele innych twierdzeń o maszynach programowych. Zostaną one pokazane w dalszych pracach.

*

Rozpatrywany tu zbiór stanów T można interpretować zależnie od potrzeb, jako zbiór liczb naturalnych skończony lub nieskończony , zbiór liczb rzeczywistych, zbiór funkcji czy też nawet bardziej złożonych pojęć matematycznych. Interpretacja zbioru T jako zbioru liczb rzeczywistych może być przydatna do opisu maszyn analogowych. Dla dobrego opisu istniejących maszyn wydaje się rzeczą konieczną wprowadzenie jeszcze bardziej złożonych pojęć matematycznych do opisu stanu maszyny.

Rozpatrzmy dla przykładu pamięć rzeczywistej maszyny cyfrowej. Pamięć taką można uważać za pewnego rodzaju graf, punkty którego są interpretowane jako komórki pamięci. Ponieważ w każdej komórce może znajdować się pewna liczba, możemy mówić o funkcji, której dziedziną jest zbiór punktów grafu, zaś przeciwdziedziną są liczby naturalne. Taki model pamięci tj. graf oraz funkcja zawartości określona na grafie dość dobrze oddaje techniczną strukturę wielu rodzajów pamięci.

Ponieważ techniczne przesyłanie zawartości komórek pamięci może odbywać się jedynie pomiędzy tymi komórkami, które są bezpośrednio połączone - każdą instrukcję maszyny możemy uważać za funkcję, która zmienia zawartość pamięci w ten sposób, że nowa zawartość zależy co najwyżej od zawartości komórek z nią sąsiadujących tj. komórek, które są z nią bezpośrednio połączone.

Dla dobrego określenia maszyny konieczne jest więc przyjęcie, że w zbiorze adresów określona jest pewna relacja binarna określająca strukturę połączeń w pamięci maszyny.

Ten sposób opisu pamięci wydaje się jednak dość niewygodny do formalnego traktowania, dlatego rzeczą wygodniejszą będzie przyjąć, że pamięć jest pewną przestrzenią ciągłą, w której jest określone pojęcie odległości. Na przestrzeni tej określona jest funkcja zawartości. Instrukcjami będą wtedy pewne operatory zmniejszające zawartość pamięci w ten sposób, że nowa zawartość dowolnego punktu przestrzeni może zależeć co najwyżej od zawartości punktów nie dalszych niż pewna z góry ustalona odległość.

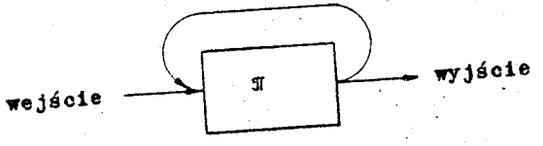
Inaczej mówiąc działanie instrukcji jest lokalne: zmiana zawartości dowolnego miejsca pamięci może zależeć jedynie od zawartości miejsc najbliższych. Miejsca dostatecznie dalekie nie mają bezpośrednio wpływu na zmianę zawartości miejsc najbliższych. Miejsca dostatecznie dalekie nie mają bezpośrednio wpływu na zmianę zawartości. Program powoduje propagację zmiany zawartości pamięci od punktu sąsiedniego do punktu sąsiedniego. Przypomina to rozchodzenie się zaburzenia w przestrzeni fizycznej.

Wydaje się, że zastosowanie modelu ciągłego dla maszyn cyfrowych może doprowadzić do głębszych i bliższych praktyce wyników, chociaż maszyny te są w swej istocie nieciągłe. Próba takiego ujęcia zostanie przedstawiona w jednym z najbliższych artykułów.

Na rozpatrywane tu sprawy można spojrzeć z nieco innego punktu widzenia. Zbiór T możemy interpretować jako alfabet skończony albo nieskończony , którego elementami mogą być liczby naturalne, liczby rzeczywiste, funkcje, przestrzenie itp. zależnie od potrzeby. Słowem w alfabecie T nazwiemy każdy ciąg skończony lub nieskończony elementów zbioru T . Będziemy mówili, że słowo W jest akceptowane przez maszynę o sterowaniu π wtedy i tylko wtedy, gdy $W = P_{\pi}(t)$, gdzie t jest pierwszym elementem słowa W . W ten sposób każda maszyna wyznacza pewien język.

Wynika stąd, że maszynę można zdefiniować nie jako funkcję, jak to uczyniliśmy na początku niniejszej pracy, a jako język zdefiniowany przez zadaną funkcję przejścia może sensowniej byłoby wtedy nazywać maszyną nie język a samą funkcję przejścia. Taka definicja może być do wielu celów nawet lepsza niż poprzednia. Maszyna jest bowiem wtedy utożsamiana ze zbiorem wszystkich wykonywanych przez nią obliczeń, również nieskończonych, i na takim zbiorze możemy wykonywać operacje teorio-mnogościowe otrzymując nowe maszyny. W ten sposób otrzymujemy pełniejszą charakterystykę pracy maszyny, gdyż wyłączone są również przypadki, kiedy maszyna się nie zatrzyma. Natomiast przy rozumieniu maszyny jako funkcji możemy rozpatrywać tylko przypadki pracy maszyny prowadzące do zatrzymania. Dla wielu zastosowań jest nie wystarczające.

Warto tu zwrócić jeszcze uwagę na następującą interpretację pojęcia maszyny : każdą maszynę można przedstawić w postaci jak niżej



Prostokątne pudełko symbolizuje urządzenie realizujące funkcję przejścia. Podany na wejściu pudełka sygnał reprezentuje stan początkowy maszyny. Urządzenie realizujące funkcję przejścia po pewnym czasie na wyjściu wyprodukuje sygnał reprezentujący następny stan maszyny. Sygnał ten ponownie jest poprzez sprzężenie zwrotne kierowany na wejście i jeżeli należy on do dziedziny funkcji przejścia produkowany jest następny stan maszyny.

W ten sposób na wyjściu maszyny pojawiają się kolejne stany procesu. Jest rzeczą ciekawą, że podany schemat maszyny jest na tyle ogólny, że mieści się w nim zarówno pojedynczy przerzutnik jak i cała maszyna całą maszynę można zresztą uważać za bardzo złożony przerzutnik . W schemacie tym mieszczą się także inne urządzenia elektroniczne, jak np. filtry, wzmacniacze, a także maszyny analogowe.

Wydaje się, że badania tego rodzaju mogą mieć również znaczenie dla dowodzenia twierdzeń, bowiem proces dowodzenia twierdzeń mieści się również w podanym schemacie maszyny i obliczenia.

Z podanych tu interpretacji płynie bardzo bogata problematyka badawcza, którą trudno omówić w tak krótkiej notatce.

STORED PROGRAM COMPUTERS

Summary

This note contains general definition of a computer and on this basis stored program computer is introduced. As a primitive notion we use the set of states T of a computer and a transition function $\mathfrak{M} : T \to T$. /$\mathfrak{M}$ is partial function/. Sequence $t_0, t_1, \ldots$ is called process if for all i $t_{i+1} = \mathfrak{M}(t_i)$. The process is finite if it has the form $t_0, t_1, \ldots, t_k$ and t_k does not belong to the domain of a transition function $\mathfrak{M}$. Some elementary properties of processes are proved. Than the relation $M \subset T \times T$ is introduced by all finite process /by fixed transition function $\mathfrak{M}$ / and it is shown that this relation is a function. This function is called machine. Some elementary properities of machines are shown, the inclusion /the simulation of one machine by another machine/ is defined, as well as the equivalence of machines is introduced.

In the second part of this note the stored program computer is defined. The notion of adressable memory, the notion of scheme of instruction and the instruction itself are defined for such kind of machines. It is shown that the class of stored program computers with modification of instructions is essentialy wider /in the sense of the concept of inclusion/ than the class of stored program computers without the modification. Thus the idea of von Neumann computer is essentialy different from those not containing the ability of modification.

Some further problem related to the theory of computers are outlined.

Warszawa, ul. Krzywickiego 34.

Z.P.