ZDZISŁAW PAWLAK
Warszawa

519.004.14:517.004.14:681.322.0011

UWAGI O TEORII MASZYN CYFROWYCH

Autor przedstawia pewną próbę sprecyzowania pojęcia maszyny cyfrowej. Omawia ogólne pojęcia maszyny matematycznej, opierając się na pojęciach jej pamięci (stanów) oraz sterowania (funkcji przejścia). Następnie definiuje maszynę adresową oraz podaje ideę określenia maszyn programowanych.

Matematyczne podstawy maszyn cyfrowych — tak jak je rozumiem — powinny służyć do rozwiązywania i badania problemów związanych z konstrukcją i zastosowaniami współczesnych maszyn matematycznych. Aby podstawy maszyn matematycznych mogły służyć do realizowania takich właśnie celów, konieczne jest przede wszystkim sprecyzowanie podstawowych pojęć nauki o tych maszynach takich, jak np. samo pojęcie maszyny, program etc. Jeżeli zgodzimy się z takim określeniem teorii maszyn matematycznych, łatwo w konsekwencji przyznać, że jak dotąd teorii takiej właściwie nie ma. W ostatnich latach pojawiło się kilka prac, które można by zaliczyć do tak rozumianej teorii maszyn matematycznych — prace te dotyczą głównie sprecyzowania pojęcia programu — jednakże sporą liczbę tych prac cechuje brak wyraźnej myśli przewodniej: nie wiadomo, czemu mianowicie mają one służyć. Czy mają to być prace o wyraźnie teoretycznym charakterze, interesujące z matematycznego punktu widzenia, czy też prace mające jakieś znaczenie dla ludzi parających się maszynami. Są to jednakże dopiero początki i na pewno w ciągu najbliższych kilku lat sprawa zostanie rozwiązana całkowicie. Dlatego też uważam, że należy na próby te patrzeć przychylnie — nawet gdyby zawierały one pewne luki.

Znacznie gorzej przedstawia się sprawa z pojęciem maszyny cyfrowej. Tutaj można znaleźć może dwie, trzy publikacje zasługujące na uwagę. Mają one jednak tę wadę, że w ramach przyjętych w nich pojęć nie można nawet zdefiniować najprostszej maszyny, np. jednoadresowej. Trudno więc mówić tu o podstawach matematycznych maszyn cyfrowych. Ukazuje się wprawdzie bardzo dużo publikacji na temat maszyn Turinga, automatów skończonych, etc., ale ktoś, kto miał do czynienia z rzeczywistymi maszynami cyfrowymi niełatwo da się przekonać, że są to dobre modele do badania interesujących go problemów, związanych z prawdziwymi maszynami czy programami. Dlatego też, aby można było z sensem mówić o teorii maszyn cyfrowych należy przede wszystkim sprecyzować pojęcia maszyny cyfrowej i programu w ten sposób, żeby definicje te dawały możliwość formułowania i rozwiązania interesujących nas zagadnień.

W artykule tym chciałem krótko przedstawić pewną próbę sprecyzowania pojęcia maszyny cyfrowej. Podana definicja pozwala z niezłym przybliżeniem opisać istniejące maszyny cyfrowe (ściślej — ich organizację) oraz dowodzić różne twierdzenia dotyczące organizacji maszyn. Twierdzeń tych jednak nie będziemy tu podawać. Są one przygotowywane do druku i zainteresowany Czytelnik znajdzie je gdzie indziej.

Doskonale zdaję sobie sprawę z niedostatków proponowanej definicji. Wielu problemów nie da się w niej nie tylko rozwiązać ale nawet sformułować. (Np. zagadnienie przerywania programów w maszynach wieloprogramowych). Tym niemniej uważam, że tego rodzaju próby są niezbędne do stworzenia matematycznych podstaw maszyn cyfrowych.

W części 1 podano ogólne pojęcie maszyny matematycznej. Samo to pojęcie nie jest może zbyt ciekawe, jednakże wprowadzenia takiego ogólnego pojęcia maszyny wydaje się celowe z dwu powodów. Po pierwsze, przemawiają za tym względy dydaktyczne. Warto zdać sobie sprawę z tego, jak rozumiemy w najogólniejszych zarysach pojęcie maszyny. Pojęcie to powinno być takie, aby wszelkie szczególne przypadki maszyn były w nim zawarte. Łatwiej wtedy zadawać sensowne pytania odnośnie do maszyn matematycznych oraz łatwiej wyjaśnić działanie konkretnych maszyn na takim ogólnym modelu. Po drugie, przy badaniu konkretnych maszyn matematycznych zachodzi często potrzeba dowodzenia pomocniczych twierdzeń o tych maszynach, dość ogólnej natury. Dowody tych twierdzeń dla szczególnych przypadków maszyn zajmują po kilka stron. Natomiast jeżeli te same twierdzenia sformułować dla przypadku ogólnego maszyny, to ich dowody skracają się do kilku wierszy maszynopisu.

Część 2 zawiera pojęcie maszyny adresowej. Maszyny te działają według raz na zawsze ustalonego programu (zrealizowanego np. w postaci tablicy połączeń). Historycznie są to więc maszyny sprzed roku 1945, zanim J. Neuman wprowadził koncepcję maszyny z pamiętanym i modyfikowanym w trakcie liczenia programem. Maszyny te są omówione w 3 części zatytułowanej: maszyny programowane. Można pokazać (czego nie czynimy w tym artykule), że maszyny J. Neumana istotnie różnią się od maszyn, które nie posiadają możliwości modyfikowania instrukcji.

Ostatnia wreszcie część 4 zawiera definicję maszyny uniwersalnej dla pewnej klasy maszyn. Ściślejsze zbadanie pojęcia maszyny uniwersalnej pozwala na lepsze zrozumienie istoty pomysłu J. Neumana — maszyny programowanej.

1. Ogólne pojęcie maszyny matematycznej

Niech T będzie pewnym zbiorem (skończonym albo nieskończonym). Natura elementów tego zbioru chwilowo nas nie interesuje. Zbiór ten będziemy nazywać pamięcią, a elementy tego zbioru stanami pamięci i będziemy je oznaczać małymi literami t ewentualnie ze wskaźnikami u dołu. Dalej niech π oznacza funkcję częściową (tzn. nie wszędzie określoną) o argumentach należących do zbioru T i przyjmującą wartości ze zbioru T, co zapiszemy $\pi : T \to T$. Procesem obliczenia (albo krótko procesem lub obliczeniem) będziemy nazywali ciąg

$$t_0, t_1 \ldots \tag{1}$$

jeżeli spełnia on następujące warunki:

1°. Dla każdego i, $t_i \varepsilon T$, tj. elementami ciągu (1) są stany ustalonej pamięci.

2°. Dla każdego i, $t_{i+1} = \pi(t_i)$.

Obliczenie $t_0, t_1, \ldots, t_k$ nazwiemy skończonym wtedy i tylko wtedy, gdy $t_k \varepsilon D\pi$, gdzie $D\pi$ oznacza dziedzinę funkcji, tj. zbiór tych wszystkich elementów zbioru T, dla których funkcja π jest określona. Obliczenie polega więc na tym, że według ustalonej reguły (funkcji π) przechodzimy od jednego stanu pamięci do następnego, tak długo, jak jest to możliwe. Jeżeli natrafimy na taki stan pamięci, że funkcja π nie mówi nam już, co należy dalej robić (tzn. do jakiego następnego stanu pamięci należy przejść), to obliczenie jest zakończone; jeżeli na taki stan nie natrafimy, to obliczenie będzie trwało nieskończenie długo i nigdy w naszym postępowaniu się nie zatrzymamy. Przyjęta definicja obliczenia w ogólnym zarysie oddaje istotę wszelkich obliczeń, a ściślej mówiąc dowolnych manipulacji na symbolach [1].

[1] W podobny sposób może być przedstawiony też proces produkcyjny, ale taką interpretacją nie będziemy się zajmowali w tym opracowaniu. Zainteresowanego tym tematem Czytelnika odsyłam do książki Z. Pawlaka „Matematyczne aspekty organizacji produkcji", Państwowe Wydawnictwo Ekonomiczne, 1969.

Mając definicję obliczenia, możemy wprowadzić pojęcie maszyny. Wprowadźmy najpierw relację M między stanami pamięci w następujący sposób: powiemy, że dwa stany t, t' pamięci T są w relacji M wtedy i tylko wtedy, gdy istnieje skończone obliczenie $t_0, \ldots, t_k$, takie, że $t_0 = t$ oraz $t_k = t'$, tj. obliczenia zaczynające się od stanu t i kończące się w stanie t'. Okazuje się, że przy tak zdefiniowanej relacji M dla każdego t istnieje co najwyżej jedno t' (może też wcale nie istnieć) będące w relacji M z t. A więc stan początkowy obliczenia jednoznacznie wyznacza stan końcowy obliczenia (o ile oczywiście stan taki istnieje). Możemy mówić więc o funkcji M przyporządkowującej stanom pamięci inne stany pamięci, w ten sposób, że jako argument funkcji M bierzemy dowolny stan pamięci T, następnie wykonujemy obliczenie, przechodząc do kolejnych stanów za pomocą funkcji π, tak długo, aż otrzymamy stan końcowy (o ile to jest możliwe). Ten stan końcowy nazywamy wartością funkcji M. Funkcja M jest więc nie zawsze określona. Dla pewnych stanów t wartość funkcji $M(t)$ może być określona, dla innych zaś stanów pamięci wartość funkcji $M(t)$ może być nieokreślona. Funkcję M będziemy nazywać *maszyną*. Nie jest to sprzeczne z intuicyjnym rozumieniem maszyny matematycznej.

Każdą maszynę matematyczną można sobie wyobrazić jako urządzenie, które posiada jakieś stany (pamięć T); jeżeli urządzenie to ustawimy na początku w jakimś stanie, to przechodzi ono w ustalony sposób do następnego stanu itd., i albo zmienia ono stany w sposób nieskończony, albo też może się przy jakimś stanie zatrzymać. T będziemy nazywać pamięcią maszyny M, zaś π — funkcją przejścia albo sterowaniem maszyny M. Dla określenia więc jakiejkolwiek maszyny musimy podać więc jej pamięć T oraz sterowanie π. Ten sposób opisywania maszyn nie odbiega od metod stosowanych w praktyce. Nawet prospekt handlowy dowolnej maszyny, czy też jej opis techniczny zawierają przede wszystkim opis pamięci oraz listę rozkazów, która, jak się przekonamy, jest niczym innym jak właśnie funkcją przejścia π, dla maszyny [2]).

Dla pełnego zrozumienia działania maszyny musimy jeszcze określić, co to znaczy, że maszyna oblicza wartości jakiejś funkcji. Załóżmy, że mamy pewną maszynę M i chcemy za jej pomocą obliczać wartości funkcji $f: X \to X$. (Dla uproszczenia przyjmujemy tu funkcję jednoargumentową. Uwzględnienie funkcji wieloargumentowych nie przedstawia trudności).
Zgodnie z intuicyjnym rozumieniem obliczania wartości funkcji za pomocą maszyny można przyjąć następującą definicję:

Maszyna M oblicza wartość funkcji f wtedy i tylko wtedy, gdy dla każdego $x \varepsilon \varkappa$, $f(x) = \delta\{M[\varkappa(x)]\}$, gdzie $\varkappa$ oznaczają odpowiednio funkcję kodującą i dekodującą: funkcja kodująca $\varkappa$ przyporządkowuje argumentom funkcji f stany pamięci maszyny M, zaś funkcja δ (dekodująca) odwrotnie — stanom pamięci maszyny M — wartości funkcji f.

Obliczanie wartości jakiejś funkcji przez maszynę polega więc na odpowiednim „zapisaniu" (czy przedstawieniu) argumentu x — dla którego chcemy obliczyć wartość funkcji f — w pamięci maszyny, tj. przyporządkowaniu zadanemu argumentowi odpowiedniego stanu pamięci maszyny, następnie na uruchomieniu maszyny i jeżeli maszyna się zatrzyma — na odpowiednim zinterpretowaniu (poprzez dekodowanie δ) stanu końcowego pamięci. Oczywiście wymagamy tu, aby przy ustalonej funkcji f sposób kodowania i dekodowania był taki sam dla wszystkich argumentów funkcji f. Wymagamy więc, aby metoda interpretowania działania maszyny była jednolita dla wszystkich możliwych wartości argumentów — inaczej trudno byłoby bowiem mówić sensownie o obliczaniu wartości funkcji f.

Zwróćmy tu przy okazji uwagę na dwie sprawy. Przy podanym sposobie rozumienia liczenia funkcji f, maszyna wykonuje tylko pewną część pracy: znaczną część pracy wykonuje również jej użytkownik przez ustalenie kodowania i dekodowania. Zależnie od tego, jak praca związana z obliczeniem wartości funkcji f jest podzielona między funkcje $\varkappa$, δ i M, więcej pracy może przypadać na obsługującego, bądź też na maszynę. Druga uwaga dotyczy tego, że właściwie każda maszyna liczy w istocie tylko jedną (!) funkcję M. To, że możemy za pomocą tej samej maszyny obliczać wiele różnych funkcji, polega na tym, że możemy różnie interpretować działania maszyny poprzez dobór odpowiednio funkcji kodującej i dekodującej. Mówiąc ściślej, każdą funkcję, której wartość chcemy liczyć, przedstawiamy jako złożenie trzech funkcji $\varkappa$, δ, M, z których tylko jedna jest liczona przez maszynę — a mianowicie funkcja M. Przy ustalonej funkcji M dobierając odpowiednio funkcję kodującą i dekodującą możemy więc otrzymywać różne funkcje. Gdybyśmy wzięli np. za M funkcję tożsamościową, cała praca związana z obliczeniem wartości funkcji f spadałaby na nas, poprzez odpowiedni dobór kodowania i dekodowania. Mielibyśmy więc wtedy do czynienia ze zwykłym rachunkiem za pomocą papieru i ołówka.

Warto może jeszcze zwrócić uwagę na fakt, że f niekoniecznie może oznaczać funkcję liczbową. Jako zbiór argumentów funkcji f możemy przyjąć np. zdania w jednym języku, funkcje f traktować jako algorytm tłumaczenia zdań w jednym języku na zdania w innym języku — wtedy wartością funkcji f będzie zdanie przetłumaczone. Tłumaczenie z języka na język za pomocą maszyn polega więc na tym, że umiemy znaleźć taki sposób interpretowania działania maszyny, że dla pewnych zdań (niekoniecznie wszystkich w danym języku) interpretowanych jako stan początkowy pamięci, maszyna się zatrzyma i otrzymany stan końcowy przy tej właśnie interpretacji jest zdaniem przetłumaczonym. Na tym właśnie polega fakt, że maszyny cyfrowe, budowane pierwotnie do obliczeń numerycznych, mogą być stosowane do spraw, które z rachunkiem nie mają wiele wspólnego.

2. Maszyny adresowe

Jak to stwierdziliśmy w poprzednim rozdziale, określenie każdej maszyny sprowadza się do podania jej pamięci oraz sterowania. Również maszyny, którymi zajmujemy się w tym rozdziale określimy w ten właśnie sposób.

Najpierw zdefiniujemy pamięć maszyn adresowych. Zanim przystąpimy do opisania pamięci, podamy najpierw kilka pojęć pomocniczych.

Niech A będzie zbiorem skończonym albo nieskończonym (przeliczalnym); elementy A będziemy nazywali adresami. W dalszym ciągu przyjmiemy, że A jest zbiorem liczb naturalnych 0, 1, 2, 3, ... W przypadku gdy będziemy czynili odstępstwo od tego założenia wyraźnie to zaznaczymy.

Przez Σ będziemy oznaczali skończony albo nieskończony zbiór (przeliczalny), którego elementy nazwiemy symbolami, zaś sam zbiór Σ — alfabetem. Przyjmiemy, że alfabet również jest zbiorem liczb naturalnych 0, 1, 2, 3, ... W przypadku odstąpienia od tego założenia wyraźnie to zaznaczymy. Będziemy rozpatrywać funkcje $c: A \to \Sigma$. Każdą taką funkcję będziemy nazywać zawartością pamięci. Przyjmiemy, że dowolna zawartość jest zawsze określona dla skończonej liczby wartości argumentów (adresów). Zawartość pamięci możemy więc traktować jako tabelkę, jak to pokazano np. niżej:

adres	0	1	2	3	4	5	6	7	...
symbol			2	0		3			

2) Nie podajemy tu odpowiednika wejścia i wyjścia maszyny. Można je w razie potrzeby również uwzględnić. Jednakże do celów, którymi będziemy zajmować się w tej notatce, uwzględnienie wejścia i wyjścia maszyny nie jest konieczne.

Należy to rozumieć w ten sposób, że pod pewnymi adresami zapisane są symbole ustalonego alfabetu, przy czym zapisana jest ich zawsze skończona liczba, natomiast dla pozostałych adresów funkcja zawartości jest nieokreślona. Zbiór wszystkich możliwych zawartości maszyny będziemy nazywać *pamięcią zmienną* maszyny i będziemy ją oznaczać przez C [3].

Określimy obecnie pamięć stałą maszyny wprowadzając uprzednio kilka potrzebnych do tego celu pojęć.

Schematem instrukcji będziemy nazywali funkcję postaci:

$$r : A^m \times C \to C, \qquad m = 0, 1, 2, \ldots$$

Jeżeli w schemacie instrukcji ustalimy adresy (tzn. adresy traktujemy jako parametry), to tak otrzymaną funkcję nazwiemy instrukcją m-adresową (m jest ustaloną liczbą naturalną dla danej instrukcji, $m = 0, 1, 2, 3, \ldots$). Instrukcją jest więc funkcja

$$r(\bar{a}_m) : C \to C, \qquad \bar{a}_m = \{a_1, \ldots a_m\},$$

która jedną zawartość pamięci zmiennej przeprowadza w inną zawartość tejże pamięci.

Np. instrukcje dodawania w maszynie trójadresowej możemy rozumieć jako funkcję następującą:

$$[+(a_1, a_2, a_3)]\,(c) = c',$$

gdzie

$$c'(x) = \begin{cases} c(\bar{a}_1) + c(a_2) & \text{gdy } x = a_3,\ x \varepsilon A \\ c(x) & \text{gdy } x \neq a_3. \end{cases}$$

Niech $Q = \{q_0, q_1, \ldots, q_l\}$ będzie pewnym skończonym zbiorem, zaś s funkcją, która każdemu elementowi zbioru Q przyporządkowuje instrukcję. Zbiór Q możemy traktować jako zbiór adresów pamięci stałej, natomiast s jako zawartość tej pamięci.

W pamięci stałej oznaczanej dalej przez S istnieje tylko jedna funkcja zawartości s raz na zawsze ustalona dla danej pamięci. Pod adresami Q są więc tu zapisane instrukcje, które powodują zmianę zawartości pamięci zmiennej. Pamięć maszyny adresowej składać się więc będzie z dwu pamięci — pamięci zmiennej C oraz pamięci stałej S, co zapiszemy $T = C \times S$. Ponieważ w pamięci stałej istnieje tylko jedna funkcja zawartości s i jest ona jednoznacznie wyznaczona przez podanie zbioru Q, zamiast więc pisać $T = C \times S$ wygodniej będzie w dalszym ciągu stosować zapis $T = C \times Q$.

Określiliśmy więc pamięć maszyn adresowych. Teraz przystąpimy do zdefiniowania jej sterowania, tj. funkcji przejścia π. Sterowanie jest funkcją przypisującą jednym stanom pamięci inne stany, tj. $\pi : T \to T$. Ponieważ $T = C \times Q$, więc sterowanie ma postać $\pi : T \times Q \to T \times Q$. Sterowanie π dla maszyn adresowych wygodnie jest traktować jako parę funkcji π_Q, π_C, takich, że

$$\pi_Q : C \times Q \to Q,$$
$$\pi_C : C \times Q \to C.$$

Dla dokładniejszego określenia obu tych funkcji należy wprowadzić kilka pojęć pomocniczych.
Warunkiem w pamięci zmiennej, albo krótko *warunkiem*, będziemy nazywali dowolny, ustalony podzbiór W pamięci zmiennej. Będziemy mówili, że stan pamięci zmiennej c spełnia warunek W wtedy i tylko wtedy, gdy $c \varepsilon W$; w przypadku przeciwnym powiemy, że stan c nie spełnia warunku W [4].

Np. w maszynie jednoadresowej często spotykanym warunkiem jest badanie, czy zawartość akumulatora jest równa zeru, czy też nie. Tzn. warunek ten w podanej tu terminologii wyrazimy w ten sposób, że W jest zbiorem tych wszystkich stanów pamięci zmiennej, dla których zawartość akumulatora jest równa zeru.

Wprowadzimy jeszcze dwie funkcje pomocnicze φ_1, φ_2 przyporządkowujące elementom zbioru Q elementy tego samego zbioru. Możemy teraz już określić sterowanie w następujący sposób:

$$\pi_C(c, q) = c',$$

gdzie

$$c' = [s(q)](c)$$

zaś

$$\pi Q(c, q) = \begin{cases} \varphi_1(q), & \text{gdy } c \varepsilon W, \\ \varphi_2(q), & \text{gdy } c \varepsilon \bar{W}. \end{cases}$$

Sterowanie opisuje więc jednoznacznie działanie maszyny. Gdy zadamy jej stan początkowy c, q, sterowanie powoduje wykonanie instrukcji przyporządkowanej q, tzn. zmienia zawartość pamięci zmiennej C, a następnie sprawdza, czy otrzymana zawartość spełnia warunek W, czy też nie — i zależnie od tego przechodzi do wykonania następnej instrukcji z pamięci stałej. Jeżeli funkcje φ_1 i φ_2 przyporządkowują jakiemuś q tę samą wartość q', to znaczy, że wykonanie instrukcji następnej nie zależy faktycznie od spełnienia warunku W. Mówimy wtedy, że przejście do następnego stanu jest *bezwarunkowe;* w przypadku przeciwnym mówimy o przejściu *warunkowym*. Nie powiedzieliśmy jeszcze, w jaki sposób maszyna się zatrzymuje. W tym celu najwygodniej jest wprowadzić instrukcje Stop. Możemy wprowadzić dwa rodzaje takich instrukcji: instrukcje stopu statycznego lub instrukcje stopu dynamicznego. Instrukcje te określimy następująco:

$\text{Stop}_s(c)$ = nie określone dla dowolnych c,

natomiast

$\text{Stop}_d(c) = c$ dla dowolnych c.

W przypadku pierwszego stopu instrukcja jest nieokreślona, a więc również funkcja przejścia jest nieokreślona i maszyna zakończy obliczenie przy natrafieniu na taką instrukcję; w drugim natomiast przypadku obliczenie będzie nieskończone, jednakże od pewnego miejsca stany pamięci przestaną się zmieniać. Te nie zmieniające się stany możemy też uważać za stany końcowe (co wszakże nie jest zgodne z definicją obliczenia skończonego podaną w pierwszym rozdziale).

3. Maszyny programowane

Definicja maszyny programowanej jest żmudniejsza, niż maszyny adresowej, dlatego nie będziemy jej przytaczać dokładnie, a podamy tylko ogólną ideę takich maszyn.

Maszyny programowane są szczególnym przypadkiem maszyn adresowych. Pamięć ich składa się również z dwu pamięci, pamięci zmiennej oraz pamięci stałej. O ile w maszynach adresowych nie nakładaliśmy szczególnych warunków na pamięci, to dla maszyn programowanych pamięci te muszą posiadać dodatkowe własności. Odnośnie do pamięci zmiennej wymagamy mianowicie, aby wśród jej adresów znajdował się adres wyróżniony, który oznaczać będziemy literą l, a nazywać go będziemy licznikiem rozkazów. Pamięć stała też jest szczególnej budowy. Nie będziemy jej dokładnie omawiać, a podamy tylko w zarysie najważniejsze jej cechy. W zbiorze Q wyróżnimy element początkowy i oznaczymy go przez q_0. Każde obliczenie, takie że $c_0, q_0, c_1, q_1 \ldots c_k, q_k$, nazwiemy cyklem pracy maszyny programowanej, zaś ciąg instrukcji $s(q_0), s(q_1), s(q_2), \ldots, s(q_{k-1})$ nazwiemy makroinstrukcją maszyny programowanej [5].

Każda makroinstrukcja jest też oczywiście funkcją o argumentach należących do C i wartościach należących również do C. Przyjmiemy ponadto, że każda

[3] Czasem wygodnie jest rozpatrywać pamięć zmienną jako podzbiór wszystkich możliwych zawartości. Jednakże dla rozpatrywanych tu celów, pierwsze określenie pamięci jest wystarczające.

[4] Wprowadziliśmy tu tylko jeden warunek. Czasem wygodnie jest wprowadzić więcej niż jeden warunek w pamięci i badać spełnienie jednego z nich.

[5] Czasem zamiast makroinstrukcja używany jest termin instrukcja, natomiast zamiast instrukcja mówimy mikroinstrukcja.

makroinstrukcja zmienia co najmniej zawartość licznika rozkazów w pamięci zmiennej. Przyjmiemy dalej, że każdej makroinstrukcji odpowiada symbol alfabetu pamięci zmiennej; odwzorowanie to oznaczymy literą Ψ. Funkcję przejścia (sterowanie) dla maszyny programowanej możemy wtedy określić w sposób następujący:

$$c' = \{\Psi [c\,(c\,(l))]\}\,(c)$$

Wykonanie każdej makroinstrukcji polega więc na odczytaniu symbolu zapisanego w miejscu pamięci zmiennej wskazywanym przez aktualną zawartość licznika rozkazów, następnie symbol ten jest interpretowany jako ciąg instrukcji zapisanych w pamięci stałej, zaczynający się i kończący w miejscu wyróżnionym q_0. Łatwo zauważyć, że tak określona maszyna programowana jest jednoznacznie wyznaczona przez podanie jej pamięci oraz zbioru wszystkich jej makroinstrukcji (ściślej: schematów makroinstrukcji). Zbiór makroinstrukcji maszyny programowanej nazywamy jej *listą rozkazów*. Zauważmy różnicę między maszynami programowanymi a maszynami adresowymi. Maszyna adresowa nie była wyznaczona przez jej pamięci i listy instrukcji. Działanie maszyny adresowej zależało jeszcze od tego, jaka była zawartość pamięci stałej. Podobnie jest w istocie w maszynach programowanych, jednakże zawartość pamięci stałej jest jednoznacznie wyznaczona poprzez listę makroinstrukcji, a więc lista ta determinuje działanie maszyny we wszystkich możliwych przypadkach.

Uściślenie podanej definicji nie przedstawia trudności i w ten sposób możemy dokładnie określić pojęcie maszyny programowanej, programu wewnętrznego maszyny, etc.

Pozwala to w konsekwencji stawiać i rozwiązywać różne problemy związane z maszynami programowanymi, jak np. jakie należy dobrać instrukcje, aby można z nich złożyć zadaną listę makroinstrukcji, czy też pytanie odwrotne: jaki zbiór makroinstrukcji możemy otrzymać przy zadanym zbiorze instrukcji. Możemy też sformułować różne pojęcia równoważności maszyn i badań, kiedy dwie maszyny czy klasy maszyn są równoważne.

4. Maszyny uniwersalne

W celu lepszego zrozumienia pojęcia współczesnych maszyn matematycznych wygodnie jest wprowadzić pojęcie maszyny uniwersalnej. Pojęcie to pozwoli dokładniej zrozumieć, czym są w istocie maszyny programowane.

Wprowadzimy najpierw pojęcie naśladowania (symulowania, zawierania) maszyny M' przez maszynę M. Powiemy, że maszyna M naśladuje maszynę M', wtedy i tylko wtedy, gdy spełnione są następujące dwa warunki:

1. Maszyny M oraz M' posiadają jednakową pamięć [6].
2. Dla każdego obliczenia $t_0, t_1, \ldots, t_k$ w maszynie M istnieje obliczenie $t'_0, t'_1, \ldots, t'_l$ w maszynie M' takie, że $t_0 = t'_0$, $t_k = t'_l$ oraz dla każdego i $(0 \leqslant i \leqslant k)$ istnieje takie j $(i \leqslant j \leqslant l)$, że $t_i = t'_j$.

Naśladowanie maszyny przez maszynę polega na tym, że jeżeli obie maszyny będą miały jednakowe stany początkowe pamięci, to również skończą obliczenie w jednakowych stanach pamięci oraz każdemu stanowi pamięci w obliczeniu maszyny M będzie odpowiadał stan w obliczeniu maszyny M'.

Definicja ta jest naturalna i odpowiada ona technicznemu naśladowaniu działania jednej maszyny przez inną maszynę. Jeżeli maszyna M naśladuje maszynę M', to zapiszemy $M'CM$. Powiemy, że maszyna M jest maszyną uniwersalną dla klasy maszyn K, jeżeli maszyna M może naśladować działanie każdej maszyny należącej do klasy K. Inaczej, M jest maszyną uniwersalną dla klasy K, jeżeli dla każdej maszyny $M \varepsilon K$, $M'CM$ [7].

Określmy teraz, co to jest klasa maszyn adresowych. Powiemy, że dwie maszyny adresowe należą do tej samej klasy, jeżeli posiadają one jednakową pamięć zmienną oraz ich instrukcje należą do ustalonego zbioru instrukcji. Klasa takich maszyn jest więc wyznaczona przez pamięć zmienną oraz zadany zbiór instrukcji. Oczywiście istnieje nieskończenie wiele maszyn w tak określonej klasie maszyn, gdyż każda taka maszyna jest wyznaczona przez pamięć stałą, a tych właśnie pamięci, nawet przy skończonej liczbie instrukcji, jest nieskończenie wiele. Można pokazać, że dla każdej klasy maszyn adresowych istnieje maszyna programowana, która jest maszyną uniwersalną dla tej klasy maszyny i ta maszyna uniwersalna nie należy do tej klasy maszyn adresowych. Inaczej mówiąc, programowana maszyna uniwersalna dla klasy maszyn adresowych jest maszyną istotnie różną od maszyn, które naśladuje. Naśladowanie przez tę maszynę uniwersalna dowolnej maszyny adresowej z ustalonej klasy polega na tym, że musimy umieć odpowiednio „zapisać" działanie dowolnej maszyny adresowej w pamięci maszyny uniwersalnej. Ponieważ działanie maszyny adresowej jest jednoznacznie wyznaczone przez zawartość jej pamięci stałej, wystarczy odpowiednio „przenieść" zawartość pamięci stałej maszyny, której działanie chcemy naśladować, do pamięci zmiennej maszyny uniwersalnej. Maszyna uniwersalna musi oczywiście posiadać makroinstrukcje, które pozwolą symulować instrukcje maszyny adresowej. Zawartość stałej pamięci maszyny adresowej gra więc w maszynie programowanej rolę programu!

Jeżeli natomiast zdefiniujemy klasę maszyn programowanych jako klasę maszyn posiadających jednakowa pamięć oraz makroinstrukcje należące do skończonego, ustalonego zbioru schematów instrukcji, to ponieważ każdy zbiór schematów instrukcji jednoznacznie wyznacza maszynę programowaną, więc klasa taka może zawierać tylko skończona liczbę maszyn programowanych. Łatwo znów wykazać, że dla każdej klasy maszyn programowanych istnieje programowana maszyna uniwersalna dla tej klasy i maszyna ta należy do tej klasy maszyn. Co więcej, każda maszyna programowana jest maszyną uniwersalną dla pewnej klasy maszyn.

Pojęcie maszyny uniwersalnej dokładnie więc podaje różnice między maszynami adresowanymi i maszynami programowanymi, wprowadzonymi przez von Neumana. Maszyny programowane istotnie się różnią od maszyn adresowych nie tylko z praktycznego punktu widzenia, ale i własności abstrakcyjne tych maszyn są różne.

Tak sobie wyobrażam teorię matematyczną maszyn cyfrowych. Uważam, że przede wszystkim powinna ona sprecyzować pojęcie maszyny cyfrowej, a następnie pozwalać na badanie różnych własności tych maszyn. Oczywiście zależnie od tego, jakie własności maszyn chcemy badać, definicje maszyny cyfrowej mogą być różne, uwzględniające różne cechy maszyn. Podany tu przykład definicji maszyny miał na celu zilustrowanie myśli przewodniej artykułu i nie stanowi oczywiście rozwiązania całego problemu. Tym niemniej nawet w ramach tej definicji można udowodnić szereg interesujących twierdzeń o maszynach cyfrowych.

Jestem przekonany, że w przyszłości teoria maszyn matematycznych nie tylko będzie nadążała za rozwojem technologii, co obecnie niestety nie ma miejsca, ale nawet będzie wyprzedzać postęp technologiczny — wytyczając nowe kierunki rozwoju tej dyscypliny nauki.

---

[6] Warunek o jednakowych pamięciach nie jest konieczny. Przyjęliśmy go tutaj jedynie dla uproszczenia.

[7] Pojęcie naśladowania możemy też zastosować do określenia równoważności maszyn (albo równoważności klas maszyn) w następujący sposób. Powiemy, że maszyny M oraz M' są równoważne wtedy i tylko wtedy, gdy M naśladuje M' oraz M' naśladuje M. Zagadnieniem równoważności nie będziemy się tu jednakże zajmowali.