CENTRUM OBLICZENIOWE POLSKIEJ AKADEMII NAUK
POLSKIE TOWARZYSTWO ELEKTROTECHNIKI TEORETYCZNEJ I STOSOWANEJ

NAUKOWE PROBLEMY MASZYN MATEMATYCZNYCH

MATERIAŁY Z I OGÓLNOPOLSKIEGO SYMPOZJUM 21–26 PAŹDZIERNIKA 1968

WARSZAWA 1970
PAŃSTWOWE WYDAWNICTWO NAUKOWE

Naukowe problemy maszyn
matematycznych. PWN. 1970

Zdzisław PAWLAK
Instytut Matematyczny PAN

Motto:

"Naukowcy, jak małe rybki, pływają ławicami"

"Istnieje realne niebezpieczeństwo, że nasza nauka może się udławić własnymi ekskrementami"

Erwin Chargaff, Essayas on Nucleic Acids, Elsevier, 1963.

PODSTAWY MATEMATYCZNE MASZYN CYFROWYCH

1. Organizatorzy tego sympozjum zwracając się do mnie z propozycją wygłoszenia referatu na temat podany w tytule postawili mi do rozwiązania bardzo łatwe zadanie. Cały mój referat można by zamknąć w jednym zdaniu: podstawy matematyczne maszyn cyfrowych nie istnieją. Takie stanowisko może wywołać zdziwienie. W każdym roku pojawia się bardzo wiele publikacji zawierających matematyczne rozważania dotyczące automatów skończonych, maszyn Turinga, języków bezkontekstowych itd. Uważam, że pojęcia automatu skończonego, maszyny Turinga czy języka bezkontekstowego mają duże znaczenie dla konstruktorów i użytkowników maszyn cyfrowych, jednakże nie stanowią one podstaw matematycznych tychże maszyn. Podstawy takie powinny tworzyć jakąś matematyczną teorię, w której sprecyzowane są pojęcia interesującej nas dyscypliny, a więc pojęcia takie, jak maszyna cyfrowa, pamięć rozkaz, program, obliczenia itd., a następnie w teorii tej powinny być do dyspozycji środki pozwalające na formułowanie i rozwiązywanie problemów dotyczących struktury maszyn cyfrowych oraz sposobów ich użytkowania. Jeżeli zgodzimy się na takie rozumienie podstaw matematycznych maszyn cyfrowych, a jest ono chyba zupełnie naturalne i nie odbiega od podstaw innych nauk, to przyznamy, że podstawy takie nie istnieją rzeczywiście. Nie umiemy, jak dotąd, podać zadowalającej matematycznej definicji maszyny cyfrowej, języka programowania itd., itd. W konsekwencji nie umiemy wykazać np., czy dwie jakieś maszyny są w pewnym sensie równoważne, czy dwa programy są równoważne, czy dwa języki programowania są równoważne itp. Pytania takie są ważne z praktycznego punktu widzenia, zresztą są to najprostsze pytania interesujące konstruktorów i użytkowników maszyn cyfrowych. Konstrukcja i użytkowanie maszyn stawia o wiele więcej bardziej złożonych pytań. Badania dotyczące automatów skończonych, maszyn Turinga czy też inne podobne nie tylko nie pozwalają na znalezienie odpowiedzi na takie pytania, ale nawet nie dysponują językiem pozwalającym na sformułowanie tego rodzaju problemów. Pojęcie automatu czy maszyny Turinga różni się bowiem tak dalece od maszyny cyfrowej, że przy próbie przeniesienia większości problemów z maszyn cyfrowych np. na maszyny Turinga problemy te po prostu znikają.

Warto może dodać, że nie wszyscy widzą konieczność stworzenia podstaw matematycznych maszyn cyfrowych, uważając, że matematyka jest tutaj nieprzydatna. Stanowisko takie bierze się prawdopodobnie stąd, że po pierwsze znaczna część prac publikowanych na temat teorii maszyn nie tylko nie ma jakiegokolwiek znaczenia praktycznego, ale nawet żadnej interpretacji w rzeczywistych maszynach, po drugie zaś wiele istotnych z punktu widzenia maszyn matematycznych problemów nie udało się, jak dotąd, nawet sformułować w ramach istniejących pojęć matematyki. Wnoszą oni więc, że współczesna matematyka nie dysponuje zakresem pojęć przydatnym do celów maszynowych, a to, co da się zrobić w zakresie pojęć istniejących, jest nieprzydatne, jak to wykazuje znaczna część publikacji na tematy teorii maszyn matematycznych.

Moim zdaniem, stanowisko takie nie jest słuszne. Nie ulega dla mnie wątpliwości, że stworzenie odpowiednich podstaw matematycznych maszyn cyfrowych jest niezbędne dla dalszego rozwoju maszyn i ich zastosowań. Jestem przekonany, że matematyczna kodyfikacja wiedzy o maszynach matematycznych doprowadzi do powstania nowych działów matematyki, bardziej przydatnych do celów maszynowych niż matematyka obecna, uważam jednak, że niezależnie od badań nad stworzeniem takiej matematyki istniejące środki matematyczne mogą już oddać nieocenione usługi w tworzeniu matematycznej teorii maszyn cyfrowych.

2. Wiadomo, że wszystkie pojęcia matematyczne można sprowadzić do pojęć takich, jak zbiór, relacja, funkcja. Chcąc więc stworzyć matematyczne podstawy maszyn cyfrowych musimy umieć pojęcia podstawowe dla maszyn cyfrowych sprowadzić do pojęć zbioru, relacji, funkcji. Należy zdać sobie od razu sprawę, że zadanie to można wykonać na wiele sposobów. Nie należy więc oczekiwać, że znajdzie się kiedyś "idealną" definicję maszyny cyfrowej czy programu przydatną do rozwiązywania wszelkich problemów związanych z maszynami. Definicje te muszą być przystosowane do problemów, które za ich pomocą mają być rozwiązywane, chociaż nie wyklucza to, że wszystkie definicje np. maszyny cyfrowej mogą mieścić się w jakiejś ogólnej definicji maszyny, będąc każdorazowo szczególnym przypadkiem tej definicji ogólnej. Zastanówmy się dla przykładu nad pojęciem maszyny cyfrowej.

Zgodnie z tym, co powiedzieliśmy na początku tego paragrafu, pojęcie maszyny cyfrowej musimy wyrazić za pomocą pojęć zbioru, funkcji, relacji.

Niech T będzie zbiorem, którego natura nas w tej chwili bliżej nie interesuje. Zbiór ten nazwiemy pamięcią, a jego elementy będziemy nazywali stanami pamięci T. Jeżeli t jest stanem pamięci, to zapiszemy $t \in T$. Nazwy "pamięć" oraz "stan pamięci" nie mają chwilowo żadnego głębszego sensu: są to po prostu słowa oznaczające zbiór T oraz elementy tego zbioru. Innego sensu przypisywać im nie należy.

Niech π będzie funkcją częściową o dziedzinie D_π oraz przeciwdziedzinie R_π, takich że $D_\pi \subset T$ i $R_\pi \subset T$ - lub pisząc inaczej π: $T \rightarrow T$. Obliczeniem (przy ustalonym T oraz π) będziemy nazywali każdy ciąg postaci

$$t_0, t_1, \ldots, t_i \in T , \qquad (1)$$

taki że dla każdego i, $t_{i+1} = \pi(t_i)$.

Skończony ciąg

$$t_0, t_1, \ldots, t_k, \quad t_i \in T ,$$

taki że dla każdego i $(0 \leqslant i < k)$ spełniony jest warunek (1) oraz $t_k \notin D$, nazwiemy obliczeniem skończonym.

Wprowadźmy relację binarną $M \subset T \times T$ zdefiniowaną następująco: $\langle t, t' \rangle \in M$ wtedy i tylko wtedy, gdy istnieje skończone obliczenie $t_0, \ldots, t_k$, takie że $t_0 = t$ oraz $t_k = t'$. Bardzo łatwo pokazać, że tak zdefiniowana relacja M jest funkcją, tzn. dla każdego t istnieje co najwyżej jedno t', takie że $\langle t, t' \rangle \in M$. Możemy wobec tego używać powszechnie przyjętego zapisu $t' = M(t)$. Funkcję M będziemy nazywać maszyną. Do wielu zastosowań zidentyfikowanie maszyny z pojęciem funkcji M jest wystarczające. Maszynę możnateż utożsamić ze zbiorem jej wszystkich obliczeń. Tym drugim modelem maszyny nie będziemy się tu zajmować. Maszynę M można też utożsamiać z parą $M = \langle T, \pi \rangle$.

Definicja ta, mimo swej ogólności, oddaje dość dobrze istotę działania wszelkich maszyn matematycznych. Każda maszyna ma pamięć, która może znajdować się w jednym z wielu możliwych stanów. Obliczenie czegoś przez maszynę polega na tym, że maszyna od jakiegoś ustalonego na początku stanu pamięci przechodzi kolejno do następnych stanów według zadanego z góry schematu. Schemat, w jaki sposób maszyna przechodzi od stanu do stanu, wyrażony jest tu poprzez funkcję π, zaś w rzeczywistej maszynie jest on uwarunkowany sterowaniem maszyny. Funkcję π możemy więc uważać za opis działania sterowania maszyny i będziemy ją w dalszym ciągu utożsamiali ze sterowaniem. Przechodzenie od stanu do stanu następuje w trakcie obliczenia tak długo, aż pamięć znajdzie się w takim stanie, dla którego funkcja π jest nieokreślona. Wtedy maszyna przestanie działać i obliczenie jest zakończone. Może oczywiście istnieć taki przypadek, że maszyna w trakcie obliczenia nigdy nie natrafi na stan, dla którego funkcja π jest nieokreślona i wtedy będzie ona ciągle zmieniała swe stany nigdy się nie zatrzymując.

Określimy jeszcze, co to znaczy, że maszyna M oblicza wartość funkcji f. Dla uproszczenia przyjmijmy, że f jest funkcją jednoargumentową $f: X \longrightarrow X$. Powiemy, że maszyna M oblicza funkcję f wtedy i tylko wtedy, gdy dla każdego $x \in X$ zachodzi

$$f(x) = \varphi\{M[\delta(x)]\}, \tag{2}$$

gdzie funkcje $\delta: X \to T$, $\varphi: T \to X$ zwane są odpowiednio kodowaniem i dekodowaniem (funkcję δ moglibyśmy też nazwać programem obliczenia). Obliczanie wartości funkcji $f(x)$ przez maszynę polega więc na tym, że argument x tej funkcji interpretujemy jako stan początkowy pamięci, następnie puszczamy maszynę w ruch. Maszyna wykonuje skończone obliczenie, tj. zatrzymuje się po pewnej liczbie zmiany stanów w takim stanie, który zdekodowany daje wartość liczonej funkcji. A więc to, co maszyna liczy, zależy nie tylko od samej maszyny, ale w znacznym stopniu również od tego, w jaki sposób my interpretujemy jej działanie poprzez funkcję kodującą i dekodującą. Obie te funkcje są to po prostu ustalone zasady zapisywania danych początkowych w pamięci maszyny oraz odczytywania wyników obliczenia z pamięci po zatrzymaniu się maszyny. Tym jedynie można wyjaśnić fakt, że maszyny budowane do obliczeń mogą być używane do celów, które z obliczaniem nie mają wiele wspólnego, jak np. tłumaczenie z jednego języka na inny. Jeżeli chcemy, aby maszyna tłumaczyła zdania jednego języka na zdania w innym języku, znaczy to, że musimy umieć znaleźć dla każdego zdania w jednym języku taki stan początkowy pamięci, że jeżeli w tym stanie puścimy maszynę w ruch i maszyna się zatrzyma, to otrzymany w ten sposób stan końcowy przy ustalonej metodzie jego interpretowania będzie przedstawiał zdanie przetłumaczone. W przykładzie tym x we wzorze (2)

oznacza zdanie w jednym języku, zaś $f(x)$ - zdanie przetłumaczone na inny język.

Już przy tak ogólnych założeniach można łatwo udowodnić wiele dość ciekawych własności maszyn matematycznych. Nie będą to oczywiście własności bardzo głębokie.

W ramach wprowadzonych pojęć możemy zdefiniować dowolną maszynę cyfrową. Aby określić jakąkolwiek maszynę, należy jedynie sprecyzować bliżej zbiór T, tzn. określić pamięć maszyny oraz funkcję π, tj. jej sterowanie. Zdefiniowanie dowolnej maszyny sprowadza się więc do określenia jej pamięci oraz sterowania. Ten sposób opisywania maszyn jest zgodny z ogólnie przyjętą praktyką. Techniczny opis każdej maszyny cyfrowej sprowadza się do wyliczenia jej pamięci, rejestrów oraz dokładnego opisu sterowania, tj. do podania sposobu przechodzenia od stanu do stanu.

Spróbujemy zdefiniować w ramach podanych pojęć, co to są maszyny adresowe. Dla odróżnienia od przypadku ogólnego pamięć maszyn adresowych będziemy oznaczać zamiast literą T symbolem C. Elementami zbioru C będą funkcje postaci $c: A \to \Sigma$, tzn. $C \subset \Sigma^A$ (w przypadku ogólnym nie precyzowaliśmy, czym są stany pamięci). Zbiór A będziemy nazywali zbiorem adresów pamięci C, zaś zbiór Σ alfabetem pamięci C. Każdą funkcję c nazywamy stanem pamięci C albo zawartością pamięci C. Przyjmiemy, że pamięć C spełnia następujące warunki:

1. $\Sigma \cap A \neq 0$.
2. Każda funkcja c jest określona dla skończonej liczby wartości argumentów.
3. W zbiorze adresów A istnieje element wyróżniony l, zwany w dalszym ciągu licznikiem rozkazów maszyny - taki że dla każdego c, $c(l)$ jest określone.

Tak rozumiana pamięć jest dość dobrym przybliżeniem pamięci rzeczywistych maszyn cyfrowych. Funkcja c, tj. zawartość pamięci mówi o tym, co jest zapisane w każdym miejscu pamięci. Zbiór wszystkich takich zawartości, które mogą mieć miejsce w pamięci, jest utożsamiany z samą pamięcią. Do wielu celów takie uproszczenie jest dopuszczalne.

Zanim określimy sterowanie π, wprowadzimy najpierw pojęcie schematu rozkazu i rozkazu.

Każdą funkcję postaci

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

będziemy nazywać schematem instrukcji n adresowej. Jeżeli ustalimy wszystkie adresy w schemacie instrukcji, to funkcję

$$r_{\bar{a}_n} : C \to C, \qquad \bar{a}_n \in A^n,$$

nazwiemy instrukcją n adresową.

Pojęcie instrukcji odpowiada więc dość dobrze pojęciu instrukcji w rzeczywistej maszynie cyfrowej. Instrukcja przy zadanych adresach powoduje w określony sposób zmianę zawartości pamięci. Przyjmiemy, że z każdą maszyną związany jest skończony zbiór schematów instrukcji i instrukcje podpadające pod jeden schemat różnią się jedynie adresami. Ten zbiór schematów instrukcji maszyny jest nazywany listą instrukcji maszyny. Wprowadzimy jeszcze jedną funkcję $\varphi: \Sigma \to R$, gdzie R jest zbiorem wszystkich rozkazów maszyny. Przyjmiemy, że funkcja π jest częściowa i wzajemnie jednoznaczna. Funkcja ta w sposób wzajemnie jednoznaczny przyporządkowuje każdemu rozkazowi symbol alfabetu, który można uważać za pew-

nego rodzaju nazwę rozkazu. Teraz możemy już określić funkcje przejścia dla rozpatrywanych maszyn w następujący sposób:

$$c' = \left[\varphi(c(c(l)))\right](c) .$$

Funkcja ta mówi, że jeżeli mamy zadany jakikolwiek stan pamięci c, to nowy stan c' otrzymamy w ten sposób, że bierzemy adres wskazywany przez licznik rozkazów (tj. $c(l)$), a następnie rozpatrujemy symbol zapisany pod adresem $c(l)$ [tj. $c(c(l))$]. Symbol ten oznacza pewien rozkaz. Rozkaz ten właśnie stosujemy do stanu c i w ten sposób otrzymujemy nowy stan c'. O ile któraś z rozpatrywanych tu funkcji dla jakiegoś argumentu jest nieokreślona, to maszyna nie zmieni stanu, tzn. zatrzyma się. Tak właśnie przebiega zmiana stanu w maszynach cyfrowych.

Wprowadzone pojęcia zilustrujemy na przykładzie maszyny trójadresowej. Pamięć tej maszyny będzie miała postać $C = \sum^A$, gdzie $A = l \cup N$, $\sum = N$, zaś $N = \{0,1,2,\ldots\}$ jest zbiorem liczb naturalnych. Jako listę rozkazów maszyny trójadresowej przyjmiemy: $+a_1, a_2, a_3$; $-a_1, a_2, a_3$; $\cdot a_1, a_2, a_3$; $!\, a_1, a_2, a_3$; $!\, a_1, a_2, a_3$; $?\, a_1, a_2, a_3$ Stop a_1, a_2, a_3.

Pierwsze cztery schematy instrukcji oznaczają wykonywanie czterech działań arytmetycznych, symbol ! oznacza skok bezwarunkowy, symbol ? skok warunkowy, zaś Stop oznacza zatrzymanie maszyny.

Rozkaz dodawania zdefiniujemy następująco:

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

gdzie

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

Podobnie możemy zdefiniować pozostałe rozkazy arytmetyczne.

Rozkaz skoku zdefiniujemy następująco:

$$c' = \left[!\, a_1, a_2, a_3\right](c),$$

gdzie

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

Rozkaz warunkowy $?\, a_1, a_2, a_3$ zdefiniujemy jak niżej:

$$c' = \left[?\, a_1, a_2, a_3\right](c),$$

gdzie

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

Rozkaz Stop jest nie określony dla każdego c, a więc będzie on powodował zatrzymanie maszyny.

Dla całkowitego określenia działania maszyny należałoby podać jeszcze, w jaki sposób rozkazy są zapisywane w pamięci za pomocą symboli alfabetu. Inaczej mówiąc powinniśmy określić jeszcze funkcję φ. Dla prostoty nie będziemy jednakże tej sprawy rozpatrywali.

Podobnie możemy określić dowolną maszynę cyfrową. Nie tylko określić, ale również badać własności tak określonych maszyn. Możemy np. łatwo wprowadzić pojęcie równoważności maszyn cyfrowych i pytać, czy dodanie bądź odjęcie jakiegoś rozkazu do listy rozkazów maszyny zmienia maszynę w sposób istotny, czy też nie. Podobnie można sformułować wiele innych pytań dotyczących struktury maszyn cyfrowych i badać w prosty sposób ich własności.

W wielu miejscach czynione są próby sprecyzowania pojęcia maszyny, programu i na pewno w ciągu kilku najbliższych lat próby te zostaną uwieńczone powodzeniem.

3. W matematycznym formułowaniu podstaw maszyn cyfrowych należy postępować ostrożnie i z rozwagą. Jak wykazuje doświadczenie, niewłaściwa matematyzacja nie tylko nie daje odpowiedzi na stawiane pytania, ale wywołuje lawinę prac poprawnych formalnie, ale pozbawionych wartości z punktu widzenia maszyn matematycznych.

Rozpatrzmy dla przykładu pojęcie maszyny. Powiedzieliśmy, że maszyna jest funkcją. Taka definicja maszyny może przyczynić się do zbadania wielu interesujących własności maszyn, może też jednak być źródłem wielu nieporozumień. Jest rzeczą naturalną zapytać np., jaką klasę funkcji stanowią maszyny. Jeżeli przyjmiemy, że będziemy rozpatrywali funkcje w dziedzinie liczb naturalnych, pytamy więc, jak się ma pojęcie maszyny w stosunku do funkcji obliczalnych. Odpowiedź na to pytanie, z maszynowego punktu widzenia, nie jest zresztą zbyt interesująca. Ze względów dydaktycznych zajmowanie się takim problemem może być jednakże częściowo usprawiedliwione. Ale bardzo łatwo popaść tu w działalność czysto mechaniczną, polegającą na definiowaniu coraz to innych szczególnych przykładów maszyn i powtarzaniu ciągle tego samego pytania dotyczącego stosunku rozpatrywanych maszyn do klasy funkcji obliczalnych.

Inny przykład. W lingwistyce matematycznej język definiuje się jako pewien zbiór. Ponieważ na zbiorach możemy wykonywać operacje teoriomnogościowe, takie same operacje możemy więc wykonywać na językach, pytając, czy wyprowadzają one poza klasę określonych języków. Na pewno pytania tego typu są interesujące, ale znów bardzo łatwo pójść tutaj po najmniejszej linii oporu: można w nieskończoność definiować coraz nowe klasy języków i do znudzenia powtarzać to samo pytanie - czy operacje teoriomnogościowe wyprowadzają poza klasę badanych języków?

Przykładów tego rodzaju działalności można znaleźć pod dostatkiem w bieżącej literaturze fachowej. Podstawy matematyczne maszyn cyfrowych na pewno nie zyskują na takiej twórczości.

4. Dla prawidłowego rozwoju podstaw matematycznych maszyn cyfrowych powinny być spełnione następujące warunki:

1. Teoria maszyn matematycznych powinna się głównie zajmować problemami ważnymi dla rozwoju maszyn matematycznych i ich zastosowań.

2. Należy próbować znaleźć metody matematyczne pozwalające na właściwe formułowanie i rozwiązywanie problemów maszynowych.

3. Do jednych z pierwszych zadań teorii maszyn matematycznych powinno należeć sprecyzowanie podstawowych pojęć dla tej teorii, takich jak maszyna cyfrowa, program itd.

4. Nie należy wykluczać stosowania w teorii maszyn matematycznych pojęć niedostatecznie ugruntowanych w matematyce, chociaż sytuacji takich należałoby w miarę możliwości unikać.

Jako wnioski z powyższych warunków wynikają następujące dezyderaty:

1. Teorią maszyn matematycznych powinni się głównie zajmować ludzie znający dobrze problematykę maszynową i posiadający odpowiednie przygotowanie matematyczne.

2. Należy unikać podejmowania prac badawczych jedynie na podstawie literatury. Tematy takie z reguły w chwili podejmowania są już przestarzałe i po zakończeniu wywołują zdziwienie ich autorów, że problem ten już dawno został rozwiązany.

3. Należy szukać problematyki badawczej w zasadzie w konstrukcji oraz zastosowaniach maszyn matematycznych.

4. Należy zachęcać do publikowania rozwiązań konstrukcyjnych i programowych.

Uważam, że spełnienie powyższych warunków może przyczynić się do szybszego powstania podstaw matematycznych maszyn cyfrowych.