ZDZISŁAW PAWLAK

MASZYNY MATEMATYCZNE

WARSZAWA
PAŃSTWOWE ZAKŁADY WYDAWNICTW SZKOLNYCH

Biblioteczka Matematyczna PZWS
i czasopisma „Matematyka"

Okładkę projektował
Alojzy Balcerzak

Redaktor
Urszula Drabarek

Redaktor techniczny
Stefania Rzęcka

WARSZAWA 1971
PAŃSTWOWE ZAKŁADY WYDAWNICTW SZKOLNYCH

Wydanie I. Nakład 4 000+250 egz. Ark. wyd. 2,76; ark druk. 3,5
Oddano do składania 27 X 1970 r. Podpisano do druku 23 VII
1971 r. Druk ukończono we wrześniu 1971 r. Cena zł 13.—
Papier druk. sat. 86×122 cm, 70 g, kl. V z fabryki w Skolwinie
Zam. nr 858/1033 D-6/185

ZAKŁADY GRAFICZNE PZWS W BYDGOSZCZY

SPIS TREŚCI

WSTĘP

Celem niniejszej pracy jest zapoznanie Czytelnika z podstawowymi problemami i pojęciami teorii maszyn matematycznych.

Posługiwanie się telefonem, telewizorem sprowadza się do kilku prostych manipulacji. Posługiwanie się maszynami matematycznymi nie jest już takie proste. Dobre wykorzystanie maszyny wymaga głębokiej jej znajomości. Utarło się nawet powiedzenie, że maszyna matematyczna jest tak mądra, jak jej użytkownik; użytkownik, nie rozumiejący dostatecznie maszyny i jej możliwości, niewiele z jej pomocą osiągnie; i odwrotnie. Maszyna matematyczna pod tym względem przypomina może bardziej instrument muzyczny aniżeli urządzenie techniczne.

Stąd w ostatnich latach zaczęła się gwałtownie rozwijać teoria maszyn liczących, mająca na celu głębokie wniknięcie w istotę ich działania i zastosowań. Urządzenie to bowiem, mimo 25 lat istnienia — intensywnego rozwoju i wszechstronnych zastosowań — nadal nie jest dostatecznie zbadane, co niewątpliwie wpływa ograniczająco na zakres jego zastosowań. Dzisiejszy użytkownik nie może ograniczyć się tylko do spraw związanych z programowaniem czy metodami numerycznymi. Niezbędna jest dla niego chociażby pobieżna znajomość problemów teorii maszyn matematycznych.

W książce omówiono podstawowe pojęcia, takie jak maszyna matematyczna, obliczenie maszyny, program obliczenia, schemat obliczenia, funkcje obliczalne przez maszynę i programy, język programowania oraz system programowania — ilustrując je bardzo prostym przykładem maszyny liczącej — sześcianem z ponumerowanymi ścianami (takim jak np. kostka do gry), którego położenia ilustrują różne stany maszyny.

W oparciu o wymienione pojęcia naszkicowano podstawowe problemy teorii maszyn matematycznych, jak np. równoważności maszyn, programów, języków programowania oraz systemów programowania.

Dzięki wprowadzonemu bardzo prostemu modelowi maszyny, Czytelnik, nie będący w tej dziedzinie specjalistą, może zrozumieć dokładnie istotę dyskutowanych zagadnień, a nawet niektóre z nich próbować rozwiązywać samodzielnie.

Do czytania i zrozumienia książki nie potrzebne są żadne wiadomości z matematyki — wystarczy odrobina zdrowego rozsądku.

Książeczka ta jest przeznaczona dla uczniów starszych klas liceum oraz nauczycieli — interesujących się maszynami matematycznymi i może stanowić rozszerzenie wiadomości z tego zakresu, przewidzianych w aktualnym programie dla IV klasy liceum matematyczno-fizycznego.

Mam nadzieję, że przeczytają ją również z pewnym pożytkiem specjaliści od maszyn matematycznych, zarówno inżynierowie jak i matematycy. Znajdą oni tu nieco inne spojrzenie na dobrze im znane sprawy. Dla tej grupy Czytelników ujęcie podanego materiału może być zbyt rozwlekłe i opisowe. Z myślą o nich podana jest na zakończenie literatura, gdzie poruszone tu szkicowo problemy są omówione wyczerpująco i zwięźle.

Na zakończenie chciałbym wyrazić podziękowanie drowi Antoniemu Mazurkiewiczowi. Wielokrotne z Nim dyskusje pozwoliły mi lepiej zrozumieć, czym jest w istocie maszyna matematyczna, i miały wpływ na przedstawione tu poglądy.

Rozdział I

PAMIĘĆ

§ 1. Ogólne pojęcie pamięci maszyny

Podstawową częścią każdej maszyny matematycznej jest pamięć[1]).

Nazwa ta jest nieco myląca, gdyż sugeruje podobieństwo do pamięci organizmów żywych. Jeżeli nawet podobieństwo takie istnieje, jest ono bardzo dalekie i powierzchowne. Właściwiej byłoby nazwać pamięć maszyny „magazynem", gdyż ten właśnie fragment maszyny gra rolę bardzo zbliżoną do „magazynu" w zakładzie produkcyjnym[2]).

W magazynie maszyny przechowuje się nie przedmioty, jak to ma miejsce w magazynie fabrycznym, lecz napisy składające się z cyfr bądź liter. Pamięć maszyny jest więc rodzajem kartoteki, jednakże sposób posługiwania się tą kartoteką bardzo przypomina pracę magazynu, z którego pobiera się i przyjmuje przedmioty. Wydawanie i przyjmowanie napisów z i do pamięci maszyny odbywa się podobnie jak przyjmowanie i wydawanie przedmiotów w zwykłym magazynie.

Dla zrozumienia podstawowych problemów omawianych w tej książeczce należy przede wszystkim zdać sobie sprawę z tego, czym jest pamięć maszyny matematycznej i jakie ma ona własności. Dlatego sprawie tej poświęcimy dużo uwagi.

---

[1]) Istnieją także maszyny bezpamięciowe, jednakże nie będziemy się nimi tu zajmować, gdyż są to maszyny bardzo prymitywne i z przyjętego tu punktu widzenia nieciekawe.

[2]) W niektórych krajach dla uniknięcia mylących skojarzeń używany jest właśnie termin „magazyn" a nie „pamięć". Ponieważ w Polsce przyjął się termin „pamięć" — przy nim pozostaniemy.

Z interesującego nas tu punktu widzenia nie wszystkie cechy pamięci są ważne. Dlatego ograniczymy nasze rozważania do bardzo uproszczonego modelu pamięci, pozwalającego jednakże prześledzić niemal bez żadnych istotnych uproszczeń wszystkie interesujące nas zagadnienia. Dla naszych celów nie musimy rozpatrywać maszyny jako magazynu, przechowującego określone przedmioty. Wystarczy, jeżeli przyjmiemy, że **pamięć maszyny** *to pewien przedmiot (urządzenie techniczne), którego zasadnicza własność polega na tym, że może on znajdować się w różnych stanach.* Nie będziemy tu ściśle wyjaśniać, co to jest stan pamięci, a przyjmiemy, że jest to pojęcie powszechnie znane. W fizyce np. mówimy, że jakieś ciało jest w stanie ciekłym bądź stałym, w organizacji produkcji mówimy o stanie magazynu itp., ogólnie biorąc stan przedmiotu jest więc pewną jego własnością, którą przedmiot może w danej chwili posiadać. Zbiór wszystkich takich własności różnych stanów przedmiotu w jakiś sposób charakteryzuje przedmiot.

Drugą cechą pamięci jest to, że może ona zmieniać stany w określony sposób. Na przykład jeżeli pamięć znajduje się aktualnie w stanie s_1, to w następstwie może znaleźć się w jednym z kolejnych stanów, np. s_3, s_8, s_{13}, jednakże nie może ona przejść ze stanu s_1 np. do stanu s_2.

A więc z każdym stanem pamięci są związane pewne stany „następne". W szczególności mogą to być wszystkie stany. Jednak, ogólnie biorąc, z każdego stanu pamięć może przejść tylko do niektórych stanów, które może ona przyjmować.

Tak więc pamięć maszyny będziemy charakteryzować

a) zbiorem jej stanów,

b) sposobem zmiany stanów.

Zbiór tych wszystkich stanów, które mogą nastąpić po stanie x, oznaczamy przez S_x i będziemy nazywali **otoczeniem stanu** x.

Przyjęliśmy tu, że zbiór stanów pamięci jest skończony. W wielu przypadkach wygodnie jest przyjmować, że zbiór stanów jest nieskończony. Łatwiej wtedy czasem sformułować problemy matematyczne. My jednakże pozostaniemy tu przy założeniu, że zbiór stanów w każdej pamięci jest skończony.

ĆWICZENIA

1. Podać otoczenie każdego stanu w następujących przypadkach:

a) Stanem ciała jest jego temperatura.

b) Stanem naczynia zawierającego ciecz (np. szklanki z herbatą) est ilość cieczy w naczyniu.

c) Stanem księgozbioru jest zbiór aktualnie znajdujących się w nim książek.

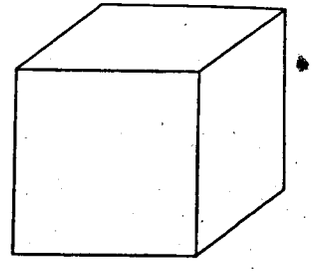
d) Położenie wskazówek zegara jest jego stanem.

e) Żarówka mogąca przyjmować dwa stany (zapalona, zgaszona).

Uwaga 1. We wszystkich przypadkach tam, gdzie zbiór stanów jest ciągły, przyjąć zbiór dyskretny, tzn. przyjąć, że ciało może zmieniać temperaturę tylko o jeden stopień; objętość wody w szklance może się zmieniać co 1 cm^3, zaś wskazówki zegara mogą zmieniać położenie skokowo, co 1 min.

§ 2. Przykład pamięci

W dalszym ciągu dla uniknięcia abstrakcyjnego toku myślenia przyjmiemy bardzo prosty model pamięci i na jego przykładzie będziemy formułować wszystkie problemy. Nie ograniczy to w niczym ogólności naszych rozważań — model ten jedynie ułatwi znakomicie zrozumienie istoty omawianych spraw. Moglibyśmy oczywiście z powodzeniem pominąć proponowany model, formułując zagadnienia w terminach ogólnych.

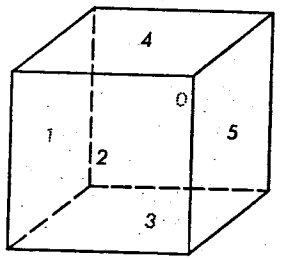


Rys. 1

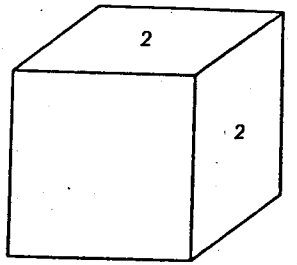
W naszych rozważaniach zwykła kostka sześcienna (rys. 1) będzie symbolizowała pamięć.

Stanem kostki nazwiemy jej położenie. A więc kostka może przyjmować sześć stanów. Przyjmijmy, że ściany kostki są w jakiś sposób ponumerowane, np. numerami od *0* do *5* (bądź literami *a, b, c, d, e, f*). Będziemy mówili, że kostka jest w stanie np. *3*, jeżeli kostka jest postawiona na ścianie oznaczonej numerem *3*, jak to pokazano na rysunku 2.

We wspomnianym przykładzie założyliśmy milcząco, że każda ściana kostki jest oznaczona odrębnym symbolem i że każde dwie



Rys. 2



Rys. 3

ściany są oznaczone różnymi symbolami. Założenia tego w dalszym ciągu nie będziemy przestrzegać, dopuszczając oznaczenie różnych ścian kostki takimi samymi symbolami; dopuszczamy również, że niektóre ściany mogą być nie oznaczone żadnymi symbolami, jak np. na rysunku 3.

Jeżeli kilka ścian jest oznaczonych tym samym symbolem, to odpowiadające im stany (położenia) kostki będziemy utożsamiać, tzn. odpowiadające tym położeniom **stany** będziemy uważali za **identyczne.**

Natomiast położenie kostki odpowiadające ścianom nieoznaczonym nazwiemy **niedopuszczalnymi** i przyjmiemy, że kostka nie może znajdować się w stanie niedopuszczalnym.

Przyjmiemy dalej, że jeżeli kostka jest w stanie x, to może przejść tylko do stanu y, takiego że y jest dopuszczalny oraz ściany oznaczone x i y mają wspólną krawędź (tzn. ściany te są sąsiednie). Nie można więc zmienić położenia kostki w ten sposób, aby przeszła ona z położenia (stanu) x do y, jeśli ściany x i y są przeciwległe.

Z naszą kostką możemy więc związać co najwyżej sześcioelementowy zbiór stanów oraz ustalony sposób ich zmiany. Kostka taka jest więc w myśl przyjętej definicji pamięcią. Tym przykładem pamięci będziemy posługiwali się w dalszym ciągu.

Zwróćmy jeszcze uwagę, że nie jest sprawą istotną, jakimi symbolami oznaczymy ściany kostki. Mogą to być dowolne liczby, litery alfabetu a nawet kolory. Istotną jest tu liczba tych symboli oraz otoczenie każdego stanu.

ĆWICZENIA

1. Oznacz ściany kostki w ten sposób, aby realizowały one następujące pamięci:

a) $S = \{0, 1, 2, 3\}$
$S_0 = \{1, 2, 3\}$; $S_1 = \{0, 1, 2\}$; $S_2 = \{0, 3\}$; $S_3 = \{0, 2\}$
b) $S = \{0, 1, 2, 3\}$
$S_0 = \{1, 2, 3\}$; $S_1 = \{1, 2\}$; $S_2 = \{0, 1, 3\}$; $S_3 = \{0, 1, 2\}$
c) $S = \{0, 1, 2, 3, 4, 5\}$
$S_0 = \{1, 2, 3, 4\}$; $S_1 = \{0, 4, 5\}$; $S_2 = \{0, 1, 3, 5\}$;
$S_3 = \{0, 2, 4, 5\}$; $S_4 = \{0, 3, 1, 5\}$; $S_5 = \{1, 2, 3, 4\}$

2. Czy zadanie 1 jest zawsze wykonalne. Jeżeli nie, to dlaczego? Możesz to sformułować ogólnie?

3. Czy zawsze rozwiązanie zadania 1 jest jednoznaczne? Dlaczego?

4. Zamiast kostki przyjmij: a) czworościan, b) dwunastościan. Co możesz powiedzieć o pamięciach zbudowanych na takich bryłach? Czy możesz podać przykłady jeszcze innych brył, które można zastosować jako pamięci?

§ 3. Wykres (graf) stanów pamięci

Własności pamięci wygodnie jest przedstawić w postaci rysunku zwanego **grafem** albo **wykresem stanów pamięci,** bądź krótko — wykresem (grafem) stanów. Wykres (graf) stanów nazywany jest też wykresem (grafem) przejścia. Obu tych nazw będziemy używali zamiennie.

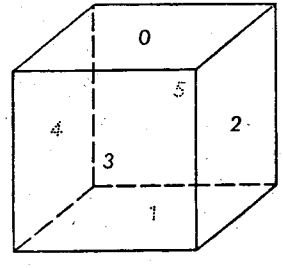
Wykres stanów wykonujemy następująco:

Stany pamięci przedstawiamy jako punkty na płaszczyźnie (zamiast punktów wygodnie jest rysować kółka). Jeżeli ze stanu x pamięć może przejść do stanu y i odwrotnie[1]), to punkt wykresu odpowiadający stanowi x (zwany dalej krótko punktem x) łączymy odcinkiem

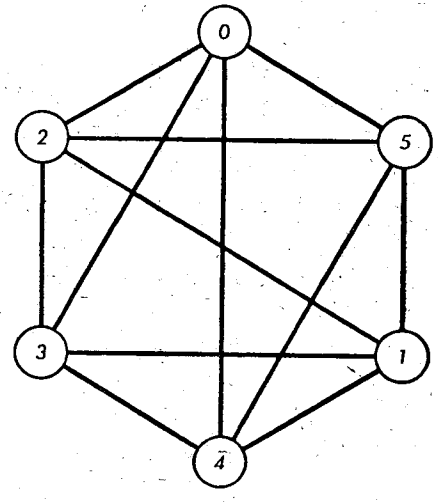
[1]) Czasem wygodnie jest rozpatrywać pamięci takie, w których możliwość przejścia od stanu x do y nie pociąga za sobą możliwości przejścia od stanu y do x. Wtedy należałoby punkty x i y łączyć na wykresie strzałką skierowaną od x do y.

z punktem y. Łącząc w ten sposób wszystkie punkty odpowiednio odcinkami otrzymamy wykres stanów (przejść) danej pamięci.

Na przykład wykres stanów dla kostki pokazanej na rysunku 4 jest pokazany na rysunku 5. Na rysunku 4 cyframi przerywanymi oznaczono stany przypisane ścianom niewidocznym. Wykres taki jest



Rys. 4



Rys. 5

przejrzystszy aniżeli np. rysunek kostki. Z wykresu stanów łatwo odczytać otoczenie każdego stanu, a więc sposób, w jaki pamięć może zmieniać swoje stany.

ĆWICZENIA

1. Podaj wykresy stanów dla pamięci podanych w ćwiczeniu 1, § 2.

2. Podaj wykresy stanów dla przedmiotów rozpatrywanych w ćwiczeniu 1, § 1.

3. Jak zmieni się wykres stanów dla kostki pokazanej na rysunku 4, jeżeli stany *1* i *3* zastąpimy stanem *0* (tj. na odpowiednich ścianach kostki zamiast *1*, *3* napiszemy *0*)?

4. Jak zmieni się wykres stanów dla kostki pokazanej na rysunku 4, gdy ściany oznaczone *4* i *3* nie będą zawierały żadnych symboli?

5. Jak zmieni się wykres stanów podany na rysunku 5, jeżeli symbole *0*, *1*, *2*, *3*, *4*, *5* zastąpimy odpowiednio przez symbole *3*, *4*, *2*, *0*, *7*, *1*? Czy sposób oznaczenia ścian kostki charakteryzuje istotne działanie pamięci?

6. Wykres stanów pamięci można otrzymać również w inny sposób. Oznaczmy ściany sześcianu punktami i połączmy ze sobą punkty, które reprezentują ściany sąsiednie (posiadające wspólne krawędzie). W przypadku gdy wszystkie ściany sześcianu są oznaczone różnymi symbolami, wykresy sporządzone tą metodą będą identyczne. Jeżeli jednak niektóre ściany zostaną oznaczone takimi samymi symbolami, wykresy będą się różnić. Na czym polega ta różnica? Zrobić wykresy stanów obiema metodami dla pamięci pokazanej na rysunku 5 przyjmując, że symbol *3* jest zastąpiony symbolem *5* oraz symbol *1* — symbolem *2*.

7. Podać wykres stanów dla czworościanu i dwunastościanu.

8. Gdybyśmy zamiast czworościanu rozpatrywali kulę jako model pamięci, co można by przyjąć jako stan takiej pamięci? W jaki sposób można opisać jej zmianę stanów?

9. Jak można by określić, kiedy dwie pamięci są jednakowe (równe)?

§ 4. Macierz stanów

Zamiast wykresu stanów (przejścia) wygodnie czasem jest stosować tzw. **macierz stanów** (albo macierz przejścia) **pamięci.**

Macierz ta ma postać, jak pokazano przykładowo niżej:

	0	*1*	*2*	*3*	*4*	*5*
0	·		×	×	×	×
1			×	×	×	×
2	×	×		×		×
3	×	×	×		×	
4	×	×		×		×
5	×	×	×		×	

Powyższa macierz odpowiada wykresowi stanów pokazanemu na rysunku 5. W wierszach i kolumnach podano stany pamięci. Jeżeli dopuszczalne jest przejście od stanu p do q, to na skrzyżowaniu kolumny p oraz wiersza q piszemy krzyżyk ×.

ĆWICZENIA

1. Podaj wykresy stanów odpowiadające następującym macierzom:

a)

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

b)

	0	1	2
0		×	×
1	×		×
2	×	×	

c)

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

§ 5. Ciągi stanów pamięci (obliczenia)

Powiedzieliśmy, że pamięć jest to urządzenie, które może posiadać pewną liczbę stanów (nie precyzując zresztą bliżej co to jest stan) i może te stany zmieniać według określonych reguł. O sposobie zmiany mówi wykres lub macierz stanów pamięci.

Pamięć o wykresie stanów pokazanym na rysunku 5 może zmieniać stany w następujący np. sposób:

0, 5, 4, 0, 2, 1, 2, 0
5, 1, 3, 2, 1, 4
4, 0, 4, 1, 5
3, 1, 2, 1, 2, 1, 2, . . .

Pierwsze trzy ciągi są skończone, ostatni zaś ciąg nie jest skończony.

Natomiast pamięć podana na rysunku 5 nie może zmieniać stanów w taki sposób:

0, 1, 4, 3, 2
5, 1, 2, 4, 0, 5
3, 1, 0, 5, 3,

gdyż po stanie *0* nie może bezpośrednio następować stan *1*, po stanie *2* — stan *4*, po stanie *1* — stan *0*, zaś po stanie *5* — stan *3*.

Ogólnie, *ciąg stanów pamięci P*

$$x_0, x_1, \ldots, x_i$$

taki, że stan x_{i+1} *w tym ciągu należy do otoczenia stanu* x_i, *nazwiemy* **obliczeniem w pamięci** P (ciąg ten może być skończony jak i nieskończony).

Czasem wygodnie jest operować także pojęciem obliczenia:

Ciąg

$$x_0, x_1, \ldots, x_i$$

jest **obliczeniem** *wtedy i tylko wtedy, gdy istnieje taka pamięć P, że ciąg ten jest obliczeniem w pamięci P.*

Obliczenie w pamięci P jest to więc jak gdyby taśma filmowa, na której zanotowano kolejne zmiany stanów pamięci P.

ĆWICZENIA

1. Podać pamięć (tzn. wykres stanów), która może zmieniać stany w następujący sposób:

a) *1, 2, 0, 3, 0, 1, 2*
1, 7, 3, 2, 1, 4
5, 3, 2, 0, 1, 2
b) *3, 4, 5, 0, 1*
2, 7, 3, 2, 1
5, 4, 5, 2

c) *a, b, c, d*
a, c, b, a, b
b, c, a, d
d) *0, 1, 0, 1, ...*

2. Czy następujące ciągi
0, 1, 5, 3, 2
2, 4, 1, 0
5, 1, 2, 4
mogą być obliczeniami w pamięci pokazanej na rysunku 5?

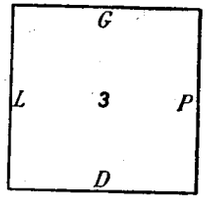
Rozdział II

MASZYNY MATEMATYCZNE

§ 1. Pojęcie maszyny oraz instrukcji maszyny

Dla wyjaśnienia pojęcia instrukcji posłużymy się wprowadzonym modelem pamięci — kostką.

Załóżmy, że na każdej ścianie kostki oprócz symbolu oznaczającego stan pamięci, wprowadzimy również oznaczenia krawędzi literami *G, D, P, L* (pierwsze litery słów Góra, Dół, Prawo, Lewo)[1]), jak to pokazano na rysunku 6.



Rys. 6

Każda ściana kostki jest więc teraz zorientowana tak, jak np. mapa, na której wyróżniamy cztery strony świata: północ, południe, wschód, zachód.

Jeżeli każda ze ścian kostki jest oznaczona inną literą, to krawędzie każdej ściany możemy oznaczyć zupełnie dowolnie[2]).

Jeżeli natomiast w kostce istnieją ściany oznaczone tą samą literą, to przyjmiemy, że oznaczenia krawędzi nie są dowolne, a spełniają pewien warunek, który omówimy nieco później — a tymczasem będziemy zajmowali się kostką, której ściany są oznaczone różnymi symbolami.

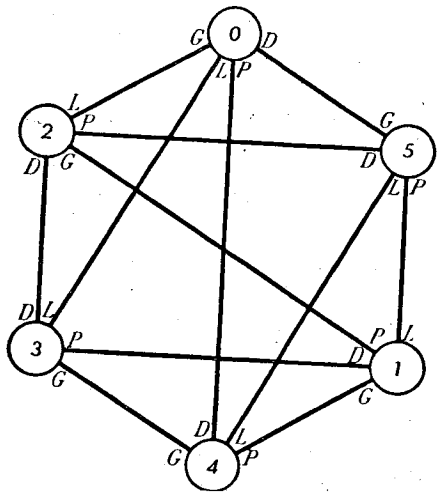
Tak oznaczoną kostkę nazwiemy **maszyną matematyczną** lub krótko maszyną, a symbole *G, D, P, L* — **instrukcjami maszyny.**

[1]) Zamiast liter *G, D, P, L* moglibyśmy użyć oznaczeń *N, S, W, E* symbolizujących strony świata.

[2]) Zauważmy, że w ten sposób każda krawędź kostki jest oznaczona dwoma symbolami, niekoniecznie jednakowymi.

Każdą maszynę definiujemy przez podanie jej pamięci oraz zbioru instrukcji.

Jeżeli instrukcje maszyny G, D, P, L naniesiemy na wykres stanów pamięci, jak to pokazano na rysunku 7 (który otrzymano z rysunku 5 pisząc przy każdym stanie w dowolny sposób litery G, D, P, L), to tak otrzymany nowy wykres będziemy nazywali **wykresem stanów maszyny,** a stany pamięci nazwiemy **stanami maszyny.**



Rys. 7

Możemy teraz wyjaśnić, co oznaczają litery G, D, P, L. Oznaczają one przewrócenie kostki wzdłuż odpowiedniej krawędzi, np. litera G oznacza przewrócenie kostki krawędzią górną (G), tzn. jeżeli maszyna o schemacie z rysunku 7 jest w stanie 0, to litera G będzie oznaczała przewrócenie kostki na ścianę oznaczoną liczbą 2; jeżeli pamięć znajduje się w stanie 5, to litera G będzie oznaczała przewrócenie kostki na ścianę 0 — itp.

Litery G, D, L, P oznaczają więc pewne czynności: zmianę położenia kostki (albo inaczej: zmianę stanu maszyny).

Sprawa ta jest bardzo prosta i nie powinna sprawić Czytelnikowi kłopotu[1]).

Wróćmy teraz do przypadku, kiedy niektóre ściany kostki są oznaczne tym samym symbolem. Aby instrukcje można było wykonać jednoznacznie, krawędzie poszczególnych ścian nie mogą być już oznaczone dowolnie (przeanalizować to na przykładzie pudełka

[1]) Tym, którzy dobrze nie zrozumieli dotychczasowych wywodów, proponuję przeanalizowanie tego problemu na pudełku od zapałek w następujący sposób: każdą ścianę pudełka oznaczmy, naklejając papierek z liczbą i literami określającymi odpowiednio stan oraz cztery krawędzie (patrz rys. 6). Następnie przeanalizujmy działanie każdej instrukcji przy dowolnych położeniach pudełka.

od zapałek). Gdybyśmy na przykład w maszynie o wykresie stanów podanym na rysunku 7 stany 2 i 0 utożsamili oznaczając oba symbolem 0, to nie byłoby dobrze, gdyż instrukcja np. D zastosowana do stanu 2 daje stan 3, zaś — do stanu 0, stan 5.

Sprawą tą nie będziemy się tu bliżej zajmować, chcielibyśmy jedynie zwrócić uwagę, że wprowadzenie instrukcji narzuca pewne ograniczenia na to, co możemy uważać za stan pamięci. Dla uproszczenia będziemy zakładali w dalszym ciągu, że każda pamięć jest zbudowana właściwie, tzn. wszystkie instrukcje są jednoznaczne. Wygodnie będzie również wprowadzić **macierz przejścia** (stanów) **maszyny.**

Macierz tę otrzymamy przez nieznaczną modyfikację macierzy stanów pamięci, mianowicie oznaczając każdą kolumnę odpowiednią instrukcją, zaś wiersz stanem oraz pisząc w wierszu p i kolumnie i stan, do którego przechodzi maszyna ze stanu p pod wpływem instrukcji i.

Przykład macierzy stanów dla maszyny o wykresie stanów podanym na rysunku 7 pokazano poniżej:

Stan \ Instrukcja	G	D	L	P
0	2	5	3	4
1	4	3	5	2
2	1	3	0	5
3	4	2	0	1
4	3	0	5	1
5	0	2	4	1

Macierz stanów jest właściwie tabelką pewnej funkcji φ.

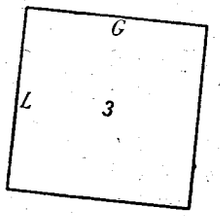
ĆWICZENIA

1. Czy można w wykresie stanów maszyny podanym na rysunku 7 tak zmieniać rozmieszczenie instrukcji G, L, P, D, aby

a) stany 0 i 2 były identyczne,

b) stany 0 i 3 były identyczne,

c) stany 0 i 4 były identyczne,



Rys. 8

a wszystkie pozostałe stany były różne? Czy można to zadanie rozwiązać, odrzucając ostatnie założenie?

2. Czy można zbudować czworościan lub dwunastościan posiadający dwa stany identyczne, a pozostałe różne?

3. Przyjmijmy, że w sześcianie tylko niektóre ściany są oznaczone literami G, D, L, P (jak np. na rys. 8) i że kostkę niezależnie od stanu można przewracać tylko przez krawędź oznaczoną jakąś literą. Czy wtedy można rozwiązać zadanie 1?

4. Sformułować ogólnie, kiedy dwie ściany dowolnej bryły mogą być utożsamione, i pokazać to na wykresie stanów maszyny.

§ 2. Instrukcje maszyny jako funkcje

Na wprowadzone instrukcje maszyny można spojrzeć nie tylko jako na opisy czynności zmiany położenia kostki, ale można im nadać również sens matematyczny.

Wszystkie instrukcje maszyny są po prostu funkcjami. Łatwo to zauważymy, jeżeli stan maszyny będziemy traktować jako argument funkcji, zaś stan, do jakiego maszyna przejdzie po zastosowaniu instrukcji — jako wartości funkcji.

Na przykład $G(0) = 2$, $G(4) = 3$ itp. (Patrz rys. 7).

Dla maszyny o wykresie stanów pokazanym na rysunku 7 instrukcje G, D, L, P będą funkcjami określonymi następującą tabelką:

x	0	1	2	3	4	5
$G(x)$	2	4	1	4	3	0
$D(x)$	5	3	3	2	0	5
$L(x)$	3	5	0	0	0	4
$P(x)$	5	2	5	1	1	1

ĆWICZENIA

1. Przyjmijmy, że pola szachownicy są kolejno ponumerowane liczbami od 0 do 63. Jaką funkcję definiuje pionek? Czy zawsze ta funkcja jest określona?

2. Podaj funkcje określone przez inne figury szachowe.

§ 3. Niezależność instrukcji maszyny

Analizując możliwe ruchy kostką zauważymy, że przejście z jednego stanu maszyny do innego stanu możemy wykonać na różne sposoby, np. ze stanu *0* do stanu *3* możemy przejść bezpośrednio wykonując instrukcję L (patrz rys. 7) albo też możemy najpierw zastosować instrukcję G, otrzymując stan $G(0) = 2$, zaś do stanu *2* zastosujemy następnie instrukcję D, otrzymując ostatecznie stan $D(2) = 3$. Istnieje jeszcze wiele innych możliwości przejścia od stanu *0* do *3*.

Powstaje więc pytanie, czy aby przejść od jakiegoś stanu do innego potrzebne są wszystkie wprowadzone instrukcje?

Łatwo zauważyć, że dla rozpatrywanego przykładu pamięci (rys. 7) wystarczą tylko instrukcje G, L. Wtedy, aby przejść np. ze stanu *0* do stanu *5*, zamiast instrukcji D możemy zastosować kolejno instrukcje L, G, G, L. Dla pozostałych stanów możemy postąpić podobnie i otrzymamy dla instrukcji D tabelkę

0	L, G, G, L
1	G, G
2	G, G, G
3	L, G
4	L, G
5	G, G

Podobnie możemy wyeliminować instrukcję P[1]).

[1]) Ciąg instrukcji, jakim zastępujemy instrukcję D w rozpatrywanym przykładzie, zależy od stanu pamięci. Ogólniej, ważniejszą rzeczą jest, abyśmy zawsze — niezależnie od stanu pamięci — tę samą instrukcję mogli zastąpić takim samym ciągiem innych instrukcji.

A więc instrukcje P, D są zbędne, gdyż można je zastąpić instrukcjami L i G. Mówimy w takim przypadku, że **zbiór instrukcji** G, D, L, P **maszyny jest zależny**. Jeżeli zaś żadnej instrukcji maszyny nie możemy zastąpić pozostałymi (tzn. nie możemy po jej wyeliminowaniu przejść od każdego stanu pamięci do dowolnego innego), to mówimy, że **zbiór instrukcji maszyny jest niezależny.**

ĆWICZENIA

1. Zbadać dla maszyny o wykresie stanów pokazanych na rysunku 7, które z instrukcji G, D, L, P są niezależne.

2. Czy można podać taką maszynę, aby zbiór jej instrukcji zawierał tylko jedną instrukcję?

3. Podać ogólnie na podstawie wykresu stanów, co to znaczy, że instrukcja maszyny jest zależna (oraz niezależna) od pozostałych instrukcji maszyny.

4. Czy można tak dobrać instrukcje G, D, L, P na rysunku 5, aby wszystkie one były niezależne?

5. Czy można tak dobrać na rysunku 5 instrukcje G, D, L, P, aby instrukcje zależne mogły być zastąpione jednakowym ciągiem innych instrukcji niezależnie od stanu maszyny?

Rozdział III

PROGRAMY OBLICZEŃ

§ 1. Pojęcie programu obliczenia

Zmodyfikujemy nieco pojęcie obliczenia podane w rozdziale I, § 5.

Ciąg (skończony albo nieskończony)

$$x_0, x_1, \ldots$$

nazwiemy **obliczeniem w maszynie** M *wtedy i tylko wtedy, gdy dla każdego* $i \geqslant 0$, x_i *jest stanem maszyny* M *oraz*

$$x_{i+1} = \varrho(x_i),$$

gdzie ϱ *oznacza instrukcję maszyny.*

Mówiąc prościej obliczenie w maszynie jest takim ciągiem stanów maszyny, że każdy następny stan w tym ciągu wynika ze stanu poprzedniego, przez zastosowanie odpowiedniej instrukcji maszyny.

Na przykład ciąg stanów

$$0, 5, 4, 0, 2, 1, 2, 0$$

maszyny o wykresie stanów podanym na rysunku 7 otrzymujemy stosując kolejno następujące instrukcje:

$$\begin{aligned} D(0) &= 5, \\ L(5) &= 4, \\ L(4) &= 0, \\ G(0) &= 1, \\ G(1) &= 1, \\ P(1) &= 2, \\ L(2) &= 0. \end{aligned}$$

Można to przedstawić jeszcze w inny sposób:

$$\begin{matrix} D & L & L & G & G & P & L \\ 0 \to 5 \to 4 \to 0 \to 2 \to 1 \to 2 \to 0 \end{matrix}$$

wpisując między stanami odpowiednie instrukcje.

Ciąg instrukcji

$$D, L, L, G, G, P, L$$

będziemy nazywać programem obliczenia $0, 5, 4, 0, 1, 1, 2, 0$. Zauważmy, że każde obliczenie w maszynie jednoznacznie definiuje pewien program.

Ogólniej możemy powiedzieć, że **programem obliczenia** *w maszynie* M

$$x_0, x_1, \ldots, x_k,$$

(skończonego lub nieskończonego) *nazywamy taki ciąg instrukcji maszyny* M

$$\varrho_0, \varrho_1, \ldots, \varrho_{k-1},$$

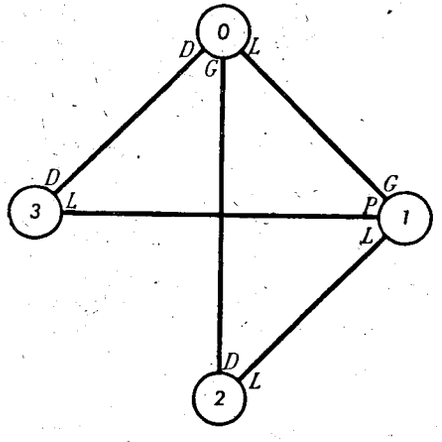
że dla każdego i

$$x_{i+1} = \varrho_i(x_i).$$

Ciąg instrukcji

1)
$$\varrho_0, \varrho_1, \ldots, \varrho_k$$

nazwiemy **programem,** *jeżeli istnieje taka maszyna* M *oraz takie obliczenie w maszynie* M



Rys. 9

2)
$$x_0, x_1, \ldots, x_k,$$

że ciąg 1) *jest programem obliczenia* 2) *w maszynie* M.

Łatwo zauważyć, że dla niektórych maszyn każdy ciąg instrukcji maszyny jest programem, dla innych zaś maszyn tak nie jest.

Pierwszy przypadek ma miejsce wtedy, gdy do każdego stanu maszyny możemy zastosować każdą z instrukcji maszyny, otrzymując za każdym razem nowy stan maszyny. Jeżeli zaś są tu

jakieś ograniczenia, tzn. do niektórych stanów nie możemy zastosować dowolnej instrukcji, wtedy nie każdy ciąg instrukcji maszyny będzie programem.

Łatwo sprawdzić, że dla maszyny o wykresie stanów podanym na rysunku 9 programami są ciągi

$$\begin{matrix} L\ G\ D\ D\ G \\ L\ G\ L\ P\ L \\ D\ D\ D\ \ldots, \end{matrix}$$

natomiast ciągi

$$\begin{matrix} D\ P\ L\ G\ D \\ P\ P\ G\ D\ L\ D \\ G\ D\ G\ L\ P \end{matrix}$$

nie mogą być programami w tej maszynie.

ĆWICZENIA

1. Czy ciągi

$$\begin{matrix} G\ L\ P\ D\ L\ G \\ D\ D\ L\ P\ G \\ L\ P\ D\ D \\ L\ L\ \ldots \\ P\ L\ P\ D \end{matrix}$$

są programami obliczeń w maszynie podanej na rysunku 9?

2. Czy można podać maszynę, dla której następujące ciągi są programami:

$$\begin{matrix} G\ L\ D\ P\ L\ P \\ P\ P\ L\ D\ G \\ P\ L\ L\ D\ G \end{matrix}$$

Czy zawsze to zadanie jest wykonalne? Uzasadnić odpowiedź.

§ 2. Zbiory obliczeń definiowane przez programy

Stwierdziliśmy w poprzednim paragrafie, że każde obliczenie jednoznacznie wyznacza pewien program. Można zadać pytanie odwrotne: czy każdemu programowi odpowiada dokładnie jedno obliczenie? Odpowiedź jest oczywista: nie.

Na ogół każdemu programowi odpowiada wiele obliczeń, które on

definiuje. Liczba tych obliczeń jest skończona, gdyż liczba stanów pamięci jest skończona.

Na przykład programowi w maszynie pokazanej na rysunku 7

$$D\,L\,L\,G\,G$$

odpowiadają obliczenia:

$$0\,5\,4\,5\,0\,2$$
$$1\,3\,0\,3\,4\,3$$
$$2\,3\,0\,3\,4\,3$$
$$3\,2\,0\,3\,4\,3$$
$$4\,0\,3\,0\,2\,1$$
$$5\,2\,0\,3\,4\,3$$

Zaś programowi w maszynie pokazanej na rysunku 9

$$L\,G\,L\,P\,L$$

odpowiada następujący zbiór obliczeń:

$$0\,1\,0\,1\,3\,1$$
$$2\,1\,0\,1\,3\,1$$
$$3\,1\,0\,1\,3\,1$$

ĆWICZENIA

1. Podać kilka programów dla maszyny pokazanej na rysunku 9 oraz dla tych programów wyznaczyć zbiory obliczeń.

2. Czy następujące ciągi:

$$0\,0\,2\,1\,2\,3\,0$$
$$3\,0\,1\,2\,3\,4\,5$$
$$1\,0\,1\,2\,3\,1\,0$$

tworzą zbiór wszystkich obliczeń jakiegoś programu?

§ 3. Programy i funkcje

Programom skończonym można przypisać funkcje, których argumentami i wartościami są stany pamięci — w sposób podobny jak to czyniliśmy dla instrukcji.

Funkcję wyznaczoną przez programy określimy w następujący sposób:

Jeżeli ciąg instrukcji

$$\varrho_0, \varrho_1, \ldots, \varrho_k$$

jest programem, to odpowiadająca mu funkcja będzie złożeniem funkcji odpowiadających poszczególnym instrukcjom programu, tzn. po zastosowaniu instrukcji ϱ_0 do otrzymanego stanu, stosujemy instrukcję ϱ_1 itd. aż do instrukcji ostatniej ϱ_k, tzn. jeżeli x jest stanem początkowym, to po zastosowaniu do niego programu $\varrho_0, \varrho_1, \ldots, \ldots, \varrho_k$ otrzymamy stan

$$y = (\varrho_k \ldots \varrho_1(\varrho_0(x))) \ldots).$$

Obliczenie

$$x$$
$$\varrho_0(x)$$
$$\varrho_1(\varrho_0(x))$$
$$\varrho_k(\ldots \varrho_1(\varrho_0(x)) \ldots)$$

będziemy nazywali **obliczeniem wartości** y **dla argumentu** x **przez program** $\varrho_0, \varrho_1, \ldots, \varrho_k$ *lub krótko* — **obliczeniem programu** $\varrho_0, \varrho_1, \ldots, \varrho_k$.

Oznaczmy przez π dowolny program skończony. Wtedy Ω_π będzie oznaczać zbiór wszystkich obliczeń programu π, zaś Φ_π — funkcję przyporządkowaną programowi π.

Funkcja Φ_π jest w prosty sposób związana ze zbiorem Ω_π, mianowicie pierwsze elementy każdego obliczenia programu są argumentami funkcji Φ_π, zaś odpowiadające im ostatnie elementy obliczenia — wartościami funkcji Φ_π. Na przykład jeżeli program $D\,L\,L\,G\,G$ ma następujący zbiór obliczeń

$$0\,5\,4\,0\,2\,1$$
$$1\,3\,0\,3\,4\,3$$
$$2\,3\,0\,3\,4\,3$$
$$3\,2\,0\,3\,4\,3$$
$$4\,0\,3\,0\,2\,1$$
$$5\,2\,0\,3\,4\,3,$$

to funkcja odpowiadająca temu programowi będzie określona tabelką:

x	0	1	2	3	4	5
$\Phi_\pi(x)$	1	3	3	3	1	3

Funkcję Φ_π będziemy nazywali **znaczeniem programu** π[1]) albo **funkcją obliczaną przez program** π.

Program (rozumiany jako ciąg skończony instrukcji) oznacza więc wykonywanie obliczenia wartości pewnych funkcji.

ĆWICZENIA

1. Podać znaczenie następujących programów:

$$P\,L\,L\,G\,D\,G$$
$$L\,L\,D\,G\,P$$
$$D\,G\,L\,P\,P$$

dla maszyny pokazanej na rysunku 7.

2. Podać znaczenie programów

$$D\,L\,D\,D\,G\,L\,G$$
$$L\,G\,D\,D\,G$$
$$L\,G\,L\,P\,L$$

dla maszyny pokazanej na rysunku 9.

§ 4. Równoważność programów

Powiemy, że *dwa* **programy są równoważne** *wtedy, jeżeli odpowiadające im funkcje są identyczne.* Pisząc to symbolicznie mamy:

$$\pi \sim \pi' \text{ wtedy i tylko wtedy, gdy } \Phi_\pi = \Phi_{\pi'}.$$

Znak $\sim$ oznacza tu równoważność programów.

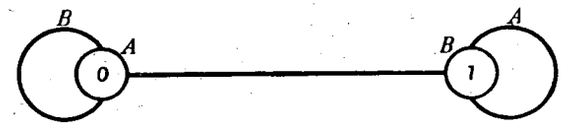
Zbiór wszystkich programów danej maszyny jest nieskończony, gdyż po ostatniej instrukcji każdego programu możemy napisać nową instrukcję, otrzymując nowy program. Natomiast zbiór funkcji obliczanych przez programy jest skończony, gdyż skończony jest zbiór stanów pamięci. Wiadomo, że na skończonym zbiorze można zbudować tylko skończoną liczbę funkcji. A więc pewne programy są na pewno równoważne.

Ważnym problemem w teorii programowania jest zagadnienie,

[1]) Czasem wygodniej znaczeniem programu π nazywać zbiór jego wszystkich obliczeń Ω_π, jednakże takim rozumieniem znaczenia programu nie będziemy się tu zajmowali.

kiedy dwa programy są równoważne. Dla skończonej pamięci posiadającej niewielką liczbę stanów zagadnienie to jest proste do rozwiązania. Wystarczy wtedy po prostu ułożyć tabelki obu funkcji obliczanych przez badane programy i stwierdzić, czy są one identyczne. Jeżeli liczba stanów pamięci jest bardzo duża (a tak właśnie jest w praktyce), to ułożenie takich tabelek jest faktycznie niemożliwe. Wtedy musimy się uciec do jakichś metod, które pozwoliłyby rozwiązać ten problem innym sposobem. Jednakże jak dotąd, w ogólnym przypadku zagadnienia tego nie udało się rozwiązać.

Dla ilustracji tego zagadnienia rozpatrzymy bardzo prosty przykład maszyny pokazanej na rysunku 10.



Rys. 10

Dwie instrukcje A, B występujące w tej maszynie są funkcjami stałymi

$$A(x) = 1,$$
$$B(x) = 0,$$

jak to łatwo stwierdzić na podstawie rysunku 10.

A więc wynika stąd, że funkcja obliczana przez dowolny program w tej maszynie zależy jedynie od ostatniej instrukcji programu, czyli

$$\varrho_0, \varrho_1, \ldots, \varrho_k \sim \varrho_k.$$

Inaczej mówiąc, mając dwa dowolne programy możemy łatwo sprawdzić ich równoważność przez zestawienie ostatnich instrukcji w porównywanych programach. Możemy to wyrazić ogólniej: W maszynie pokazanej na rysunku 10 dowolne programy

$$\varrho_0, \varrho_1, \ldots, \varrho_k; \quad \delta_0, \delta_1, \ldots, \delta_l$$

są równoważne:

$$\varrho_0, \varrho_1, \ldots, \varrho_k \sim \delta_0, \delta_1, \ldots, \delta_l$$

wtedy i tylko wtedy, gdy ich ostatnie instrukcje są identyczne, tj.

$$\varrho_k = \delta_l.$$

Aby więc stwierdzić równoważność dowolnych programów π π' w tej maszynie, nie musimy pisać tabelek funkcji obliczalnych

przez te programy, a wystarczy przeanalizować odpowiednio strukturę samych programów (w tym przykładzie w bardzo prosty sposób).

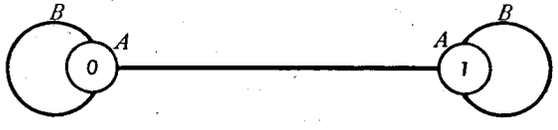
Takie właśnie metody mieliśmy na myśli mówiąc o praktycznym badaniu równoważności programów.

Mając sprecyzowane pojęcie równoważności programów w maszynie, możemy łatwo już wprowadzić działania na programach pozwalające na ich upraszczanie. W naszym przykładzie każdy program możemy zastąpić ostatnią jego instrukcją, otrzymując program mu równoważny.

ĆWICZENIA

1. Rozpatrzmy następującą maszynę (patrz rys. 11)

Instrukcja		A	B
Stan	0	1	0
	1	0	1



Rys. 11

Wykazać, że dla dowolnych programów w tej maszynie zachodzą następujące równoważności:

a) $\varrho_0, \varrho_1, \ldots, \varrho_k, B \sim \varrho_0, \varrho_1, \ldots, \varrho_k,$

b) $\underbrace{AA \ldots A}_{2n} \sim B, \quad n = 1, 2,$

c) $\underbrace{AA \ldots A}_{2n+1} \sim A,$

d) $BA \sim A.$

2. Uprościć na podstawie równoważności a), b), c), d) następujące programy:

$$B\,A\,A\,B\,B\,B\,A\,B$$
$$A\,A\,A\,B\,A\,B$$
$$B\,B\,B\,B\,A\,A\,B$$

Rozdział IV

WŁASNOŚCI MASZYN

§ 1. Funkcje obliczalne przez maszynę

Stwierdziliśmy, że każdy program w maszynie oblicza jakąś funkcję, której argumentami i wartościami są stany maszyny.

Ponieważ z każdą maszyną związany jest pewien zbiór programów, każdy program zaś wyznacza obliczalną przez niego funkcję, więc każda maszyna określa pewien zbiór funkcji, które mogą być obliczane przez programy tej maszyny. Funkcje te będziemy nazywali **funkcjami obliczalnymi** przez tę maszynę.

Wyznaczenie w ogólnym przypadku funkcji obliczalnych przez maszynę jest sprawą trudną, dlatego ograniczymy się tu jedynie do rozważenia prostych przykładów ilustrujących to zagadnienie.

Łatwo na przykład sprawdzić, że zbiór funkcji obliczalnych przez maszynę podaną na rysunku 10 składa się tylko z dwu funkcji stałych

$$A(x) = 1,$$
$$B(x) = 0.$$

Podobnie zbiór funkcji obliczalnych przez maszynę pokazaną na rysunku 11 składa się również z dwu funkcji

$$A(x) = \sim x,$$
$$B(x) = x.$$

Symbol $\sim x$ oznacza negację, która jest określona następująco:

$$\sim 0 = 1,$$
$$\sim 1 = 0.$$

Jedna z tych funkcji jest negacją, druga zaś tożsamością.

W rzeczywistości funkcje obliczalne przez maszyny są oczywiście o wiele bardziej skomplikowane i jest ich bardzo dużo.

Podane przykłady mają tylko w prosty sposób ilustrować główną myśl tego paragrafu.

ĆWICZENIA

1. Podać maszynę o pamięci dwustanowej, której zbiór funkcji składa się z czterech funkcji, dwu funkcji stałych — jedna przyjmująca zawsze wartość 0, druga wartość 1 — oraz funkcji identycznościowej i negacji.

2. Podać maszynę o pamięci dwustanowej, której zbiór funkcji przez nią obliczalnych składa się z dowolnych trzech funkcji podanych w ćwiczeniu 1.

§ 2. Zawieranie i równoważność maszyn

Będziemy mówili, że **maszyna M jest zawarta w maszynie M',** *co zapiszemy symbolicznie*

$$M < M',$$

gdy $F_M \subset F_{M'}$, *gdzie* F_M *oznacza zbiór wszystkich funkcji obliczalnych przez maszynę* M.

Jeżeli więc $M < M'$, to maszyna M jest w pewnym sensie „mniejsza" od maszyny M', za jej pomocą bowiem można obliczać mniejszą ilość różnych funkcji, aniżeli za pomocą maszyny M'. Stosunek ten wyraża więc jak gdyby, która maszyna ma większe możliwości zastosowań i w jakimś sensie charakteryzuje użyteczność maszyn.

Jeżeli maszyny M i M' obliczają te same funkcje, to powiemy, że są one **równoważne.** Symbolicznie:

$$M \sim M' \text{ wtedy i tylko wtedy, gdy } F_M = F_{M'}.$$

Oczywiście, jeżeli $M \sim M'$, to $M < M'$ oraz $M > M'$.

Pojęcia zawierania i równoważności maszyn pozwalają więc klasyfikować maszyny z punktu widzenia ich zdolności obliczeniowych.

ĆWICZENIA

1. Sprawdzić, które z następujących maszyn zawierają się, bądź są równoważne:

a)

S \ I	a	b
0	1	—
1	0	1

b)

S \ I	a	b
0	1	0
1	0	1

c)

S \ I	a	b
0	0	0
1	0	1

d)

S \ I	a	b
0	1	0
1	1	1

e)

S \ I	a	b
0	1	—
1	1	1

f)

S \ I	a	b
0	1	1
1	1	—

2. Można łatwo określić działania teoriomnogościowe na maszynach: sumę i iloczyn w następujący sposób:

a) **Iloczynem maszyn** M' i M'' nazwiemy maszynę M taką, że wszystkie obliczalne przez nią funkcje są jednocześnie obliczalne przez maszyny M' oraz M''.

b) **Sumą maszyn** M' i M'' nazwiemy maszynę M taką, że oblicza ona wszystkie funkcje obliczalne przez maszynę M' bądź M''.

Skonstruować z dwu dowolnych maszyn z ćwiczenia 1 sumę oraz iloczyn.

Czy zadania te są zawsze wykonalne?

§ 3. Naśladowanie działania maszyny przez inną maszynę

Pojęcia zawierania się oraz równoważności maszyn charakteryzowały możliwości obliczeniowe maszyn. Czasem jednakże takie porównywanie maszyn jest nie wystarczające. Nie zawsze chodzi nam bowiem tylko o to, czy dwie maszyny mają obliczyć te same funkcje, ale jeszcze możemy być zainteresowani tym, czy obliczenia te odbywają się w jednakowy czy też różny sposób. Dla porównywania maszyn z tego właśnie punktu widzenia wprowadzimy pojęcie naśladowania działania maszyny przez maszynę.

Będziemy mówili, że **maszyna M' naśladuje maszynę M** (symbolicznie $M \subset M'$), *gdy dla każdego obliczenia w maszynie M $x_0, x_1, \ldots, x_k, \ldots$ istnieje obliczenie w maszynie M' $y_0, y_1, \ldots, y_l, \ldots$ spełniające następujące warunki:*

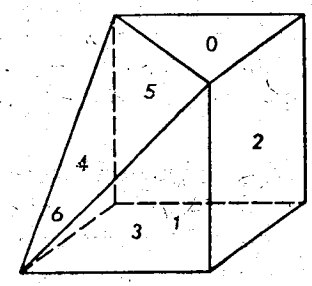
1° $x_0 = y_0$,

2° *jeżeli obliczenia są skończone oraz kończą się odpowiednio stanami x_k, y_l, to $x_k = y_l$,*

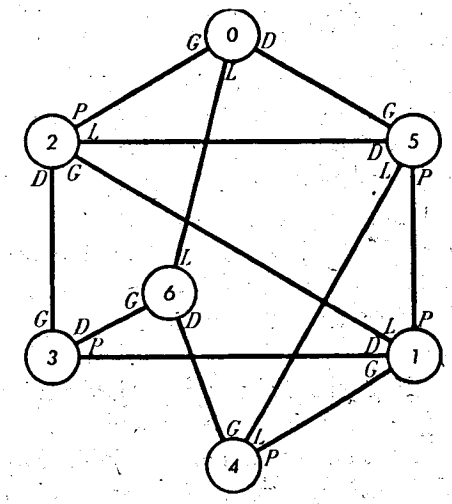
3° *dla każdych dwu stanów sąsiednich x_i, x_{i+1} w obliczeniu $x_0, x_1, \ldots, x_k, \ldots$ istnieją dwa stany y_p, y_q w obliczeniu $y_0, y_1, \ldots, y_l, \ldots$, takie, że $i \leqslant p < q$ oraz $y_p = x_i$, $y_q = x_{i+1}$.*

Znaczy to, że jeżeli maszyna M' naśladuje maszynę M, to działania obu maszyn są w pewnym sensie podobne. Podobieństwo to polega na tym, że jeżeli w trakcie jakiegokolwiek obliczenia maszyna M będzie znajdowała się kolejno w stanach $x_0, x_1, \ldots, x_k, \ldots$, i jeżeli maszyna M' rozpocznie działanie od stanu x_0, to również będzie przechodziła przez stany $x_0, x_1, \ldots, x_k, \ldots$ z tym że między kolejnymi stanami x_i, x_{i+1} może ona jeszcze przechodzić przez inne stany; jeżeli maszyna M zakończy działanie, to naśladująca ją maszyna M' również zakończy działanie i obie znajdą się w jednakowych stanach końcowych.

Wyjaśnimy to bliżej na przykładzie kostki, którą posługiwaliśmy się do tej pory w przykładach. Wyobraźmy sobie, że od kostki pokazanej na rysunku 4 odcięto czworościan otrzymując bryłę, jak to pokazano na rysunku 12. Numeracja ścian w tej bryle jest taka sama, jak w sześcianie na rysunku 4, z tym że nowo otrzymaną ścianę oznaczono liczbą 6. Wykres stanów wraz z odpowiednimi instrukcjami tej nowej maszyny pokazano na rysunku 13, a odpowiadającą mu macierz przejść podano niżej:



Rys. 12



Rys. 13

S \ I	G	D	L	P
0	2	5	6	—
1	4	3	2	5
2	1	3	5	0
3	2	6	—	1
4	6	—	5	1
5	0	2	4	1
6	3	4	0	—

Oznaczmy maszynę pokazaną na rysunku 7 przez M, zaś maszynę pokazaną na rysunku 13 przez M'. Łatwo sprawdzić, że $M \subset M'$.

Rozpatrzmy dla przykładu obliczenie w maszynie M:

$$0\,5\,4\,3\,0\,4\,1\,5$$

oraz obliczenie w maszynie M':

$$0\,5\,4\,6\,3\,2\,0\,5\,4\,1\,5.$$

Maszyna M mogła przejść od stanu *4* do *3* bezpośrednio, natomiast w maszynie M' od stanu *4* do *3* nie możemy przejść bezpośrednio, ale np. poprzez stan *6*. Podobnie ze stanu *3* do *0* maszyna M może przejść bezpośrednio, natomiast maszyna M' musi przejść jeszcze przez jakiś inny stan, np. *2*. Tak samo ze stanu *0* do stanu *4* maszyna M przechodzi bezpośrednio, zaś maszyna M' wymaga przejścia przez stan pośredni, np. *5*. Ze stanu np. *0* do *5* obie maszyny mogą przejść bezpośrednio. Definicja naśladowania taką sytuację dopuszcza również.

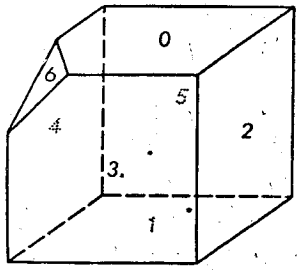
Maszyna M' jest więc jak gdyby bardziej złożona od maszyny M i jednocześnie może w pewien sposób „naśladować" działanie maszyny M, tzn. z każdego obliczenia maszyny M' można otrzymać obliczenie maszyny M przez odpowiednie pousuwanie stanów. W tym sensie właśnie należy rozumieć naśladowanie działania jednej maszyny przez inną.

Oczywiście każda maszyna naśladuje samą siebie...

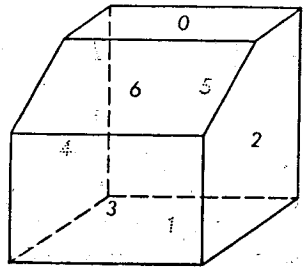
Łatwo również stwierdzić, że jeżeli $M \subset M'$, to $M < M'$, tzn. że maszyna M' może obliczyć wszystkie te funkcje, które liczy maszyna M.

ĆWICZENIA

1. Czy prawdą jest, że jeżeli $M < M'$, to $M \subset M'$?
2. Podać wykresy oraz macierze stanów dla maszyn pokazanych



Rys. 14



Rys. 15

na rysunkach 14 i 15 (instrukcje przyjąć dowolnie). Czy maszyny te naśladują maszynę pokazaną na rysunku 7 oraz na rysunku 13? Czy maszyny te mogą naśladować się nawzajem?

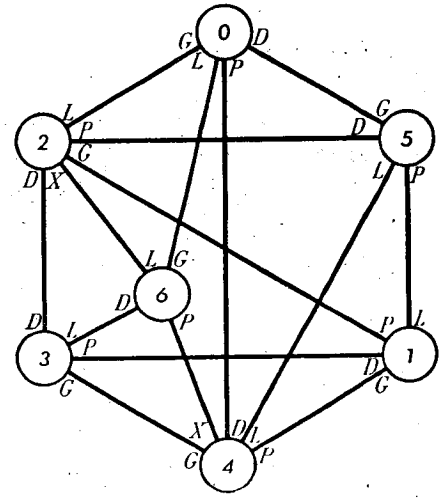
§ 4. Izomorfizm maszyn

Jeżeli maszyny M i M' naśladują się wzajemnie, to powiemy, że M i M' są **izomorficzne** *i zapiszemy $M \simeq M'$.*

Pisząc symbolicznie mamy:

$M \simeq M'$ wtedy i tylko wtedy, gdy $M \subset M'$ oraz $M' \subset M$.

Dla ilustracji izomorfizmu maszyn rozpatrzmy dwie maszyny pokazane na rysunku 13 oraz 16. Pierwszą z nich oznaczamy przez M, drugą zaś przez M'. Maszynę M' otrzymano z maszyny pokazanej na rysunku 7 przez wprowadzenie nowego stanu, jak to pokazano na rysunku 15. Maszyny M i M' posiadają jednakowe ilości stanów, a ponadto maszyna M' ma jedną instrukcję więcej (instrukcję X) aniżeli maszyna M.



Rys. 16

Łatwo stwierdzić, że obie te maszyny są izomorficzne. Najłatwiej prześledzić to na modelu geometrycznym (rys. 12 i 15).

Pokażemy najpierw, że $M \subset M'$. Z rysunku można stwierdzić, że dla każdego przejścia od stanu x do stanu y w maszynie M istnieje przejście od stanów x do y w maszynie M', co znaczy, że $M \subset M'$. Na przykład obliczeniom w maszynie M.

$$0\,6\,4^{1})$$
$$6\,0\,2$$

[1]) Zamiast obliczeń *0 6 4*, *6 0 2* możemy wziąć też inne obliczenia (np. *0 5 4* oraz *6 3 2*).

odpowiadają przejścia

$$\begin{matrix} 0\ 4 \\ 6\ 2 \end{matrix}$$

w maszynie M'.

Pozostałe zaś przejścia bezpośrednie od stanu do stanu są identyczne. A więc $M' \subset M$. Stąd wniosek, że dwie maszyny M i M' są izomorficzne.

Maszyny te mimo różnej budowy mają więc w pewnym sensie jednakowe możliwości obliczeniowe: to co może zrobić jedna z nich, może również zrobić i druga.

Można również łatwo stwierdzić, że jeżeli maszyny są izomorficzne, to są równoważne.

ĆWICZENIA

1. Czy prawdą jest, że jeżeli maszyny są równoważne, to są izomorficzne?

2. Czy maszyny o różnej liczbie stanów mogą być izomorficzne?

Rozdział V

JĘZYKI PROGRAMOWANIA

§ 1. Uwagi wstępne

Stwierdziliśmy poprzednio, że każde obliczenie polega na wykonaniu kolejnych instrukcji programu. Z każdym obliczeniem związany jest więc pewien program i z każdym programem związane są pewne obliczenia, które on określa.

Opisywanie obliczeń za pomocą programów ma jednak pewną istotną wadę. Wyobraźmy sobie, że chcemy ułożyć program obliczenia, które jest bardzo długie, tzn. zawiera np. 1 000 000 kroków[1]).

Program takiego obliczenia zawierałby prawie 1 000 000 instrukcji. W praktyce takie programy byłyby więc zupełnie nieużyteczne. Okazuje się, że jest wyjście z tej kłopotliwej sytuacji. Programy można zapisywać w pewien skrócony sposób, tak że programy bardzo długich nawet obliczeń są od nich samych wielokrotnie krótsze. Takie skrócenie zapisu nie jest możliwe dla wszelkich możliwych obliczeń, niemniej jednak prawie wszystkie obliczenia mające jakieś znaczenie praktyczne da się zapisać w taki właśnie skrócony sposób.

Do tego celu potrzebny jest specjalny system zapisu programów. System ten nazywany jest **językiem programowania.**

§ 2. Schematy obliczeń

Jak pamiętamy, programy służyły do opisu pewnych czynności, polegających na przewracaniu kostki w myśl określonych reguł (ściślej: do zmiany stanów pamięci maszyny).

[1]) W rzeczywistości występują obliczenia o wiele dłuższe.

Obecnie podamy specjalny język służący do tego samego celu. Wyrażenia tego języka będziemy nazywali **schematami obliczeń** lub krótko schematami. Schematy te będą opisami czynności, jakie należy wykonać, aby zmienić odpowiednio stan maszyny. Schematy obliczeń różnią się od programów tym, że są one bardziej zwarte; nie każda instrukcja musi występować w schemacie tyle razy, ile razy będzie ona wykonana — jak to ma właśnie miejsce w programie.

Dla określenia języka ustalimy najpierw **podstawowe symbole,** z jakich będą się składały wyrażenia języka programowania (schematy obliczeń). Zbiór tych symboli będziemy nazywali **alfabetem języka.**

Dla ustalenia uwagi przyjmijmy jako alfabet rozpatrywanego języka następujące symbole:

1) $$G,\ D,\ L,\ P,\ !,\ ?,\ Stop.$$

Symbole te będziemy nazywać **instrukcjami programowymi,** w odróżnieniu od instrukcji maszyny, którymi zajmowaliśmy się do tej pory[1]).

Schematem obliczeń lub krótko schematem będziemy nazywali następujące wyrażenie:

$$\begin{aligned} &0: \varrho_0, k_0;\\ &1: \varrho_1, k_1;\\ &\ldots\ldots\\ &n: \varrho_n, k_n, \end{aligned}$$

gdzie ϱ_i są instrukcjami programowymi oraz k_i są liczbami zawartymi między 0 i n $(0 \leqslant k_i \leqslant n)$.

Trójkę symboli p, ϱ_p, k_p nazwiemy **wierszem schematu.**

Liczby od 1 do n będziemy nazywali **numerami instrukcji lub wiersza**[2]).

Liczby $k_0, \ldots, k_n$ będziemy nazywali **numerami następnej instrukcji.**

Przyjmiemy jeszcze dla uproszczenia dodatkowe założenia odnoszące się do budowy schematów obliczeń. Mianowicie, jeżeli instrukcją programową jest jedna z liter D, G, L, P, *Stop*, to numeru następnej instrukcji przy nich nie piszemy. Natomiast przy instrukcjach $?$, $!$ piszemy dowolne liczby naturalne mniejsze od n.

Oto przykłady prostych schematów obliczeń:

$0: G$	$0: ?, 2$	$0: Stop$
$1: D$	$1: L$	$1: !, 3$
$2: L$	$2: P$	$2: G$
$3: ?, 0$	$3: ?, 1$	$3: D$
$4: ?, 5$	$4: !, 3$	$4: Stop$
$5: Stop$		$5: G$

Innych ograniczeń na schematy obliczeń nie nakładamy.

§ 3. Co oznaczają schematy obliczeń

Schematy obliczeń, podobnie jak programy, służą do opisu zmiany stanów pamięci maszyny.

Zasady postępowania według schematu są następujące:

Zaczynamy zawsze wykonywanie schematu od pierwszej instrukcji (tzn. od instrukcji oznaczanej numerem 0).

Załóżmy, że wykonaliśmy już instrukcję o numerze i w schemacie. Wtedy jako następną wykonujemy instrukcję o numerze j, którą określa następująca zasada:

a) Jeżeli w i-tym wierszu jest instrukcja *Stop*, to j jest nieokreślone, tzn. kończymy działanie;

b) Jeżeli i-ty wiersz ma postać $i: \varrho_i$, gdzie

$$\varrho_i = G,\ D,\ L,\ P, \quad \text{to} \quad j = i+1,$$

tzn. po wykonaniu i-tej instrukcji postaci G, D, L, P wykonujemy $i+1$ instrukcję schematu;

c) Jeżeli i-ty wiersz ma postać $i: !, k_i$, to przechodzimy do wykonania k_i-tej instrukcji;

d) Jeżeli i-ty wiersz ma postać $i: ?, k_i$ oraz stan pamięci po wykonaniu $i-1$ instrukcji-jest 0, to $j = i+1$, tzn. jako następną wykonujemy instrukcję $i+1$ w schemacie; jeżeli po wykonaniu $i-1$ instrukcji stan pamięci był różny od zera, to $j = k_i$, tzn. jako następną wykonujemy instrukcję zapisaną w wierszu k_i schematu.

Instrukcje programowe G, D, L, P będziemy nazywali **podstawo-**

[1]) Wszystkie instrukcje maszyny G, D, L, P są więc również instrukcjami programowymi, ale nie odwrotnie, tzn. nie wszystkie instrukcje programowe są instrukcjami maszyny.

[2]) Numery instrukcji są nazywane też etykietami lub adresami instrukcji, my jednakże pozostaniemy przy terminie „numer instrukcji".

wymi, pozostałe zaś — **pomocniczymi;** w szczególności instrukcje *!* — **przejściem bezwarunkowym,** a instrukcje *?* — **przejściem warunkowym.**

Będziemy też mówili, że instrukcja *?* jest **instrukcją warunkową,** zaś pozostałe instrukcje nazwiemy **instrukcjami bezwarunkowymi.**

Ponadto powiemy, że jeżeli pamięć jest w stanie zero, to spełnia warunek, a w przypadku, gdy pamięć jest w stanie różnym od zera, powiemy, że nie spełnia ona warunku.

Przykład

Rozpatrzmy następujący schemat:

0: *G*
1: *D*
2: *L*
3: *?*, 6
4: *G*
5: *Stop*
6: *G*
7: *!*, 1

i maszynę pokazaną na rysunku 7.

Jeżeli przyjmiemy, że maszyna znajduje się w stanie np. *3*, to po zastosowaniu do tego stanu powyższego schematu otrzymamy następujące obliczenie:

3, 4, 0, 3, 4, 0, 3, 4, ...

Kropki w tym obliczeniu oznaczają, że dalej będą się powtarzać cyklicznie liczby *0, 3, 4*. A więc jest to obliczenie nieskończone.

Gdybyśmy schemat ten zastosowali do stanu *4*, to otrzymalibyśmy obliczenie następujące:

4, 3, 2, 0, 2.

Obliczenie w tym przypadku jest skończone.

ĆWICZENIA

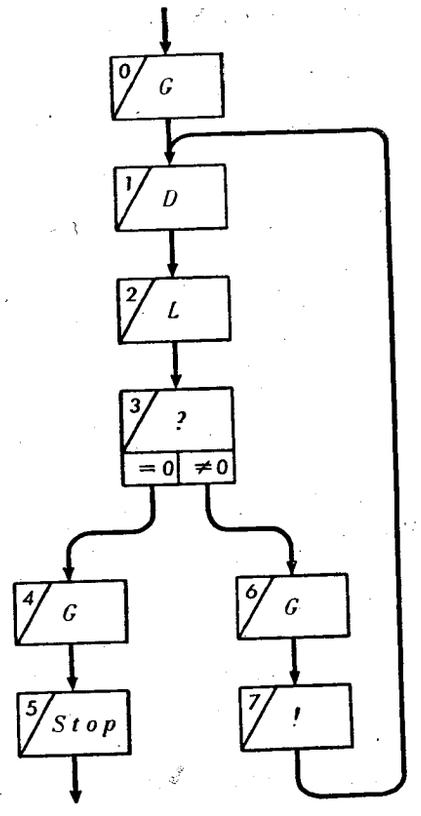
1. Podać obliczenia dla poprzednio rozpatrywanego schematu i maszyny oraz 4 stanów początkowych *1, 2, 3, 5*.

2. Podać przykłady innych schematów i rozpatrzyć opisywane przez nie obliczenia dla różnych stanów początkowych maszyny.

§ 4. Sieci działań

Schematy obliczeń wygodnie jest przedstawić w postaci rysunków zwanych sieciami działań.

Rozpatrywany w poprzednim paragrafie schemat będzie miał sieci działań pokazane na rysunku 17. Na rysunku tym każdy wiersz schematu jest przedstawiony w prostokącie, po którego lewej stronie podany jest numer wiersza, a w środku wpisana instrukcja. Jeżeli po instrukcji *i* wykonujemy instrukcję *j*, to odpowiadające im na schemacie prostokąty łączymy strzałką skierowaną od prostokąta *i* do prostokąta *j*. A więc z prostokąta, w którym wpisana jest instrukcja bezwarunkowa, wychodzi zawsze jedna strzałka; natomiast w przypadku gdy w prostokącie wpisana jest instrukcja warunkowa, wychodzą z niego dwie strzałki: jedna wskazuje następną instrukcję, gdy warunek jest spełniony, druga — wskazuje następną instrukcję, gdy warunek jest nie spełniony.



Rys. 17

Ostatnia instrukcja w schemacie jest oznaczona strzałką z napisem „*Stop*".

Tak skonstruowana sieć działań znakomicie ułatwia czytanie w układzie schematów programów.

1. Podać sieci działań dla następujących schematów:

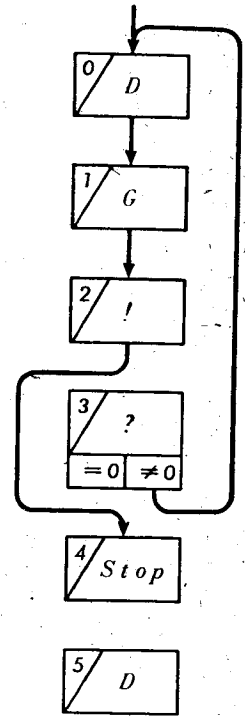
0: P	0: G	0: ?, 3
1: !, 3	1: D	1: G
2: ?, 0	3: !, 0	3: !, 1
3: *Stop*	3: !, 0	3: !, 1
	4: *Stop*	4: *Stop*
	5: ?, 2	5: G

§ 5. Schematy obliczeń zbudowane poprawnie

Rozpatrzmy następujący przykład schematu:

0: D
1: G
2: !, 4
3: ?, 0
4: *Stop*
5: D

Odpowiadająca mu sieć działań jest pokazana na rysunku 18. Zauważmy, że do prostokąta nr 3 oraz 5 nie wchodzą żadne strzałki. A więc jeżeli przyjmiemy, że zawsze zaczynamy wykonywanie schematu instrukcji nr 0, to widać, że nigdy nie dojdziemy do wykonania instrukcji 3 ani 5. Są one więc w tym schemacie zbędne. Możemy je więc opuścić bez naruszenia istotnych własności schematu.



Rys. 18

Taki schemat będziemy nazywać **niepoprawnie zbudowanym.** Natomiast schemat, który nie zawiera żadnych zbędnych instrukcji w wyżej podanym sensie, nazwiemy **schematem zbudowanym poprawnie.**

Nie będziemy pojęcia poprawnego schematu definiować ściślej, podany przykład wyjaśnia sens tego terminu dostatecznie jasno. W dalszym ciągu przez schemat będziemy zawsze rozumieli schemat poprawnie zbudowany.

1. Które z podanych niżej schematów są zbudowane poprawnie?

0: G	0: G	0: D
1: D	1: L	1: G
2: L	2: ?, 0	2: L
3: !	3: *Stop*	3: ?, 1
4: *Stop*	4: !	4: L
5: ?, 1	5: *Stop*	5: *Stop*

§ 6. Schematy obliczeń i programy obliczeń

Każdy schemat obliczenia możemy traktować jako skrócony zapis pewnego zbioru programów.

Rozpatrzmy dla przykładu schemat, którego sieć działań jest pokazana na rysunku 17. Łatwo sprawdzić, że wszelkie programy, jakie otrzymamy postępując według tego schematu, będą miały jedną z postaci:

G, D, L, G
G, D, L, G, D, L, G
$G, D, L, G, D, L, G, D, L, G$
$G, D, L, G, D, L, G, D, L, G, D, L, G$
. . . . . .
$G, D, L, G, \ldots, G, D, L, G, \ldots, G, D, L, G.$

To znaczy, że po wykonaniu pierwszej instrukcji G, dalej wykonaliśmy instrukcje D, L, a następnie w zależności od spełnienia warunku wykonamy czwartą instrukcję G i na tym zakończymy działanie albo też przy niespełnionym warunku wykonamy szóstą instrukcję (też G) i później przejdziemy do wykonywania pierwszej instrukcji (D) itd. Możemy to zapisać krótko: programy otrzymane ze schematu pokazanego na rysunku 17 mogą mieć postać

1) $$G, D, L, (G, D, L)^*, G,$$

gdzie

$$(G, D, L)^*$$

oznacza dowolny z następujących programów:

$\emptyset$ — program pusty
G, D, L
G, D, L, G, D, L
$G, D, L, G, D, L, G, D, L$
. . . . . ,

tj. dowolne powtórzenie programu zawartego w nawiasie (również powtórzenie puste).

Wyrażenia postaci 1) są nazywane **wyrażeniami regularnymi.**

Na podstawie dowolnego schematu i jego sieci działań możemy napisać odpowiadające mu wyrażenia, dające wszystkie możliwe programy wyznaczone przez zadany schemat programów.

ĆWICZENIA

1. Podać wyrażenia regularne dla następujących schematów:

0: *G*	0: *D*	0: *D*
1: *D*	1: *G*	1: *G*
2: *?*, 1	2: *P*	2: *!*, 4
3: *L*	3: *L*	3: *L*
4: *G*	4: *?*, 2	4: *?*, 0
5: *Stop*	5: *Stop*	5: *Stop*

§ 7. Schematy obliczeń i funkcje

Każdy schemat — podobnie jak i program — wyznacza pewną funkcję. Na przykład schemat

0: *G*
1: *D*
2: *L*
3: *?*, 6
4: *G*
5: *Stop*
6: *G*
7: *!*, 1,

którego sieć działań pokazano na rysunku 17, wraz z maszyną pokazaną na rysunku 7, wyznacza następującą funkcję:

x	0	1	2	3	4	5
$f(x)$	2	—	2	—	2	3

Wartość tej funkcji dla argumentów *1* i *3* jest nieokreślona, co zaznaczono w tabelce kreskami.

Kolejne wartości tej funkcji obliczono w następujący sposób:

Dla początkowego stanu pamięci *0* otrzymamy na podstawie rysunku 17 i rysunku 7 następujące obliczenie:

$$\begin{aligned} &0,\\ &G(0) = 2,\\ &D(2) = 3,\\ &L(3) = 0,\\ &G(0) = 2, \end{aligned}$$

tj. $f(0) = 2$.

Dla stanu początkowego *1* obliczenie będzie miało następującą postać:

$$\begin{aligned} &1,\\ &G(1) = 4,\\ &D(4) = 0,\\ &L(0) = 3,\\ &G(3) = 4,\\ &D(4) = 0,\\ &L(0) = 3,\\ &\ldots\ldots \end{aligned}$$

i dalej będą się nieskończenie powtarzać stany *4, 0, 3*. A więc funkcja $f(x)$, dla $x = 1$, jest nieokreślona.

W podobny sposób można wyznaczyć pozostałe wartości funkcji.

Zauważmy, że zawsze wtedy, gdy funkcja jest nieokreślona, „chodzimy w kółko" po jakiejś zamkniętej drodze w sieci działań oraz wykresie stanów maszyny.

ĆWICZENIA

1. Podać funkcje zdefiniowane przez maszynę na rysunku 7 oraz następujące schematy:

0: *G*	0: *L*	0: *G*
1: *G*	1: *D*	1: *L*
2: *!*, 4	2: *?*, 0	2: *L*
3: *L*	3: *D*	3: *?*, 5
4: *?*, 3	4: *L*	4: *Stop*
5: *Stop*	5: *Stop*	5: *!*, 1.

§ 8. Równoważność schematów obliczeń

Jeżeli dwa schematy definiują jednakową funkcję, to powiemy, że są one **równoważne** (zakładamy tu, że w obu przypadkach mamy tę samą maszynę).

Sprawdzanie równoważności schematów metodą, w której szukamy funkcji definiowanych przez oba schematy, jest uciążliwe nawet dla rozpatrywanych przez nas prostych przykładów. Dlatego byłoby rzeczą wielce wskazaną, opracowanie innej metody sprawdzania równoważności — łatwiejszej w użyciu. Na przykład metoda taka mogłaby polegać na tym, że dla zbadania, czy dwa schematy są równoważne, czy nie, dowiadujemy się na podstawie analizy postaci schematów.

Byłoby rzeczą pożyteczną, gdybyśmy potrafili, mając zadaną pewną liczbę schematów — o których wiemy, że są równoważne — wyprowadzić wszelkie inne schematy równoważne. Jednakże zadanie to jest nierozwiązalne.

ĆWICZENIA

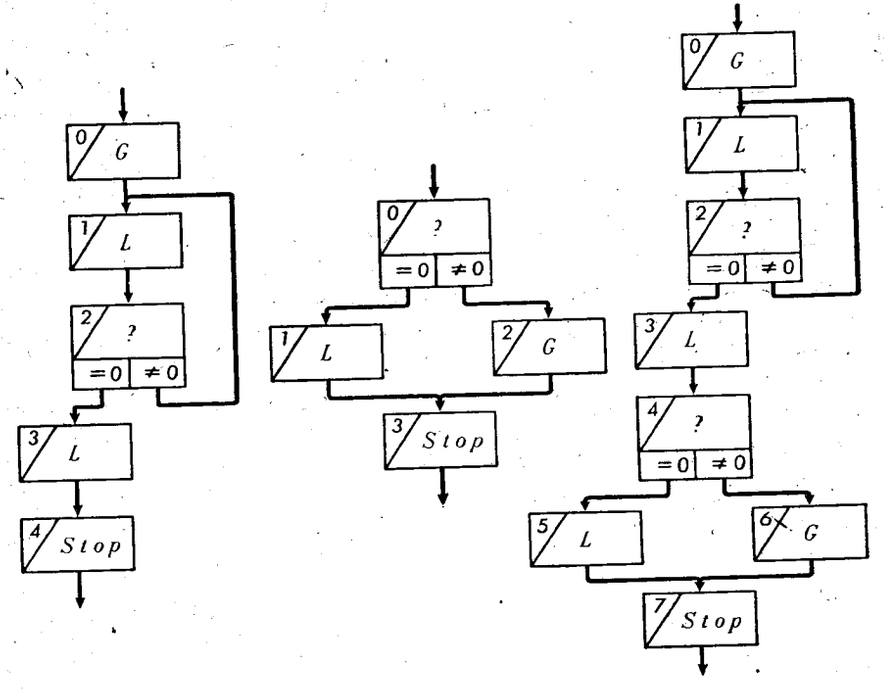
1. Sprawdzić, czy następujące schematy są równoważne (przyjąć maszynę podaną na rys. 7):

0: *G*	0: *G*
1: *L*	1: *?*, 0
2: *?*, 0	2: *L*
3: *Stop*	3: *Stop*

§ 9. Operacje na schematach obliczeń

Schematy obliczeń można ze sobą łączyć otrzymując nowe schematy. Łączenie to najlepiej wyjaśnić na podstawie sieci działań.

Jeżeli początek jednej sieci działań połączymy z końcem innej sieci działań w ten sposób, że instrukcję *Stop* w drugiej sieci zastąpimy pierwszą instrukcją pierwszej sieci, to tak otrzymana sieć jest również siecią działań jakiegoś schematu, o ile tylko odpowiednio przenumerujemy instrukcje w drugim schemacie. Ilustruje to np. rysunek 19.



Rys. 19

Do każdego numeru instrukcji w drugim schemacie dodaliśmy liczbę 4 (numer ostatniej instrukcji w pierwszym schemacie). W ten sposób uległ również zmianie numer następnej instrukcji w instrukcji warunkowej w drugim schemacie.

Otrzymanej w ten sposób sieci działań odpowiada schemat jak niżej:

0: G
1: L
2: $?$, 1
3: L
4: $?$, 6
5: L
6: G
7: *Stop*

Można łatwo sprawdzić, że złożeniu schematów obliczeń odpowiada złożenie funkcji, tj. jeżeli schemat P definiuje funkcję $f_P(x)$, zaś schemat P' — funkcję $f_{P'}(y)$, to złożenie schematów P, P' będzie definiowało funkcję $f_P(f_{P'}(y))$, przy założeniu, że do końca schematu P dołączyliśmy początek schematu P'.

Drugą ważną operacją na schematach obliczeń jest **iteracja** (powtarzanie) **danego schematu.** Operację tę również najłatwiej wyjaśnić na sieci działań.

Iteracja schematu polega na zastąpieniu w schemacie instrukcji *Stop* instrukcją warunkową oraz w przypadku spełnienia warunku przejścia do instrukcji *Stop*, zaś w przypadku niespełnienia warunku, przejścia do instrukcji pierwszej (tj. zerowej) schematu.

Oprócz tych dwu operacji można określić jeszcze inne operacje na schematach obliczeń, które pozwalają na tworzenie najpierw prostych schematów, a następnie łączenie ich w większe całości.

Na przykład schemat obliczania funkcji $\sqrt{\lg(x)}$ można złożyć z dwu schematów: schematu obliczania $\lg(x)$ oraz $\sqrt{y}$.

Możliwość działania na schematach upraszcza znacznie posługiwanie się maszynami.

ĆWICZENIA

1. Podać schematy będące złożeniami następujących schematów:

0: G	0: D	0: D
1: D	1: G	1: $!$, 3
2: L	2: L	2: G
3: $!$, 5	3: $?$, 2	3: $?$, 2
4: *Stop*	4: G	4: *Stop*
5: $?$, 1	5: *Stop*	
6: *Stop*		

§ 10. Równoważność języków programowania

Każdy język programowania przy ustalonej maszynie wyznacza pewien zbiór funkcji, które są zdefiniowane przez schematy obliczeń tego języka.

Mając zadane dwa języki i ustaloną maszynę możemy pytać, czy oba te języki wyznaczają jednakowe klasy funkcji, w takim przypadku możemy mówić o równoważności języków programowania.

Może się zdarzyć również taka sytuacja, że jeden język określa szerszą klasę funkcji niż drugi; ten pierwszy język jest więc w pewnym sensie „bogatszy" od pierwszego. W praktyce ważne jest pytanie, w jaki sposób rozstrzygnąć, czy zadane dwa języki programowania są równoważne, czy któryś z nich jest bogatszy.

Zagadnienie to do tej pory nie jest jednakże rozwiązane.

ĆWICZENIA

1. Czym się różni klasa funkcji definiowana przez maszynę od klasy funkcji definiowanej przez język programowania i tę samą maszynę?

2. Czy klasy te są zawsze równe (różne)? Od czego to zależy?

§ 11. Systemy programowania

W poprzednim paragrafie dla porównania języków programowania przyjmowaliśmy, że maszyna jest ustalona. Nic jednakże nie stoi na przeszkodzie, abyśmy rozpatrywali parę: język — maszyna, jako samodzielną jednostkę pojęciową.

Para taka będzie nazywana **systemem programowania.** Oczywiście, każdy system programowania wyznacza pewną klasę funkcji. Podobnie jak poprzednio możemy więc pytać o równoważność systemów rozumiejąc przez **systemy równoważne** takie systemy, które wyznaczają tę samą klasę funkcji.

Pozostaje więc pytanie: który z dwu równoważnych systemów programowania jest lepszy (przy założeniu, że maszyny w obu systemach są różne)?

Możemy bowiem przy ustalonej klasie funkcji, które chcemy w sy-

stemie obliczać, zbudować prostą maszynę, natomiast posługiwać się skomplikowanym językiem programowania wyrównującym niedostatki maszyny. Możemy też postąpić odwrotnie — budując złożoną maszynę i stosunkowo prosty język, otrzymując w rezultacie identyczne jak poprzednio możliwości obliczeniowe.

Jest to znany od początku istnienia maszyn matematycznych problem — do tej pory zresztą nie rozstrzygnięty — jaką część możliwości obliczeniowych systemu zrealizować maszynowo, a jaką programowo — lub jak to mówią fachowcy, co realizować w „hardwarze", a co w „softwarze".

Jak już wspominaliśmy, sprawy te nie są do tej pory dostatecznie zbadane ani z praktycznego, ani też z teoretycznego punktu widzenia.

ĆWICZENIA

1. Na podstawie rozpatrywanych do tej pory przykładów maszyn i języków podać dwa systemy programowania. Co możesz powiedzieć o ich równoważności?

ZAKOŃCZENIE

Podsumujmy krótko problemy i wyniki, które rozważaliśmy w tej książce.

Najpierw określiliśmy bardzo ogólnie pamięć maszyny — jako urządzenie mogące przyjmować różne stany. Takie rozumienie pamięci jest wystarczające do badania wielu istotnych własności maszyn matematycznych. Określonej w ten sposób pamięci maszyny odpowiada w matematyce pojęcie relacji, która wyznacza „następne" stany pamięci.

Wprowadzając pojęcie instrukcji maszyny rozumianej jako funkcja, która każdemu stanowi pamięci przyporządkowuje następny stan pamięci, mogliśmy określić maszynę matematyczną jako parę złożoną ze zbioru stanów oraz zbioru instrukcji. W związku z tym jest jednoznacznie wyznaczona przez podanie jej pamięci oraz zbioru instrukcji maszyny.

Następnie rozpatrywaliśmy pojęcie obliczenia w maszynie, programu obliczenia oraz funkcje obliczalne przez maszyny i programy.

Dalej rozpatrywaliśmy języki programowania i systemy programowania.

Zrozumienie powyższych pojęć stanowi klucz do rozumienia problematyki teorii maszyn matematycznych.

Na proste, zdawałoby się, pytania związane z problematyką maszyn matematycznych — wciąż nie ma odpowiedzi.

Znalezienie odpowiedzi na te pytania (jak np. równoważność programów, języków programowania czy też systemów programowania) ma duże znaczenie praktyczne jak i poznawcze. Dlatego w ostatnich latach obserwuje się bardzo intensywny rozwój badań teoretycznych w dziedzinie maszyn liczących.

Dokładne zrozumienie pojęcia maszyny i rozwiązanie związanych z tym problemów może mieć poważne znaczenie nie tylko dla maszyn matematycznych.

Dla wielu osób interesujących się matematyką i jej zastosowaniami otwiera się, również w Polsce, nowe ciekawe i ważne pole działalności twórczej.

LITERATURA

N. Van Ba: *Maszyny adresowe i ich programy, praca doktorska.*
H. Barański: *Programy statyczne i dynamiczne maszyn adresowych*, praca doktorska.
R. Bartoszyński: *Some Remarks on Extensions of Stochastic Automata*, Biuletyn PAN, w druku.
A. Blikle: *On the Notion of Process*, Z. Math. Logik Grundlagen Math, 11/1965/, 257—271;
Formalisation of Parenthesis-free Languages, ibid., 12/1966, 177—186;
Investigations in the Theory of Addressless Computers, Bull. Acad. Polon. Sci., Ser. Sci. Math. Astronom. Phys., 14/1966/, 203—208;
About the Meaning of Programs, ibid., 18/1970/, 341—344;
Functions Definable by Means of Programs, ibid., 18/1970/, 391—393;
An Algebra of Flow-algorithms, ibid., 18/1970, 395—398;
Addressless Units for Carrying out Loop-free computations, Prepr 1, 1970;
Algorithmically Definable Functions, Preprint 2, 1970.
P. Dembiński: *On the Nation of Machine Instructions*, Bull. Acad. Polon. Sci., Ser, Sci. Math. Astronom. Phys., 16/1968/;
Some Relations in the Set of Abstract Machines and their Properties, ibid., 17/1969/, 761—764;
An Operation of Composition in the Set of Abstract Machines, ibid., 18/1969/, 765—768.
J. Ejsmund: *O pewnej klasie maszyn m-adresowych*, Studia Logica, w druku.
A. Góraj, M. Mirkowska, A. Paluszkiewicz: *On the Nation of the Description of the Program*, Bull. Acad. Pol. Sci., Ser. Sci. Math., 18/1970/, 499—505.
Z. Grodzki: *On the Equivalence of Markov Normal Algorithms*, Bull. Acad. Pol. Sci. Ser. Sci. Math. Astronom. Phys., 16/1968/, 333—336;
The k-machines, ibid., 18/1970/, 399—402;
The Sets of Computation, ibid., w druku;
The Degree of Complexity of the Sets of Computations II, ibid., w druku;
Z. Grodzki: *The Degree of the Sete of Computations* II, ibid., w druku;
On Some Class of the k-machines, maszynopis;
The Complexity of Some Class of the k-machines; w druku.
The Complexity of Markov Normal Algorithms, maszynopis.
W. Kwasowiec: *On Some Properties of Machines*, Algorytmy 10 (1969), 21—24;
Generable Sets, Information and Control, w druku;
Some Properties of Machines, Z. Math. Log. Grundlagen Math, w druku;
Machines and Their Computations, Bull. Acad. Pol. Sci., Ser. Sci. Math. Astronom. Phys, 18 (1970), 403—406;
Relational Machines, ibid., 18 (1970), 545—549;
Operations on Relational Machines, ibid., 18 (1970), w druku.
B. Lesisz: *Maszyny ciągłe*, maszynopis.
A. W. Mazurkiewicz: *A Note Enumerable Grammars, Information and Control* 14 (1969), 555—558;
Proving Algorithms by Tail Functions, Information and Control, w druku

Z. Pawlak: *On the Notion of a Computer*, Logic. Math. and Phil. Sci., 3 (1968), 255—267;
Maszyny programowane, *Algorytmy* 10 (1969), 5—19.
Z. Raś: *On the Inclusion and the Equivalence of Computing Machines*, Bull. Acad. Pol. Sci., Ser. Sci. Math., 18 (1970), 407—411. Errata, ibid., 18 (1970);
On a Algebraic Properties of Computing Machines, ibid., w druku;
Deductive Systems of Computing Machine, ibid., w druku;
On Applications of Deductive Systems of Computing Machines to Context Languages, w druku.
H. Rossner: *Formalizacja pojęcia programu*, *Algorytmy* 10 (1969), 25—43.
G. Rozenberg: *Finite Memory Address Machines are Universal*, Bull. Acad. Pol. Sci., Ser. Sci. Math., 17 (1969), 401—403;
Constant-program Machines are Universal for a Class of Countable Machines, ibid.;
The Unsolvability of an Isomorphism Problem for Address Machines, w druku;
A. Salwicki: *Formalized Algorithmic Languages*, Bull. Acad. Pol. Sci., Ser. Sci. Math., 18 (1970), 227—232;
On the Equivalence of FS-expresions and Programs, ibid., 18 (1970), 275—278;
On the Predicate Calculi with Iteration Quantifiers, ibid., 18 (1970), 279—286.
A. Skowron: *Semantic Translation of Programing Languages*, ibid., 18 (1970);
A Syntactic Translation of the Machine-determined Languages, ibid., 18 (1970);
A Representability of Languages Determined by Machine Systems, ibid., w druku;
Included Machine Systems, ibid., w druku;
Semantic Translation, Zeitsch. Math. Log., w druku.
Z. Sozańska: *Some Properties of m-address Machines*, Studia logica 25, w druku.
Z. Sozańska, W. Kwasowiec: *Organizacja maszyn matematycznych*. Komentarz do wykładu doc. dra Z. Pawlaka.
A. Walat: *Własności k-maszyn*, w druku.