tutu matematycznego ...
Nr inw. 4931

Zdzisław Pawlak

ORGANIZACJA MASZYN MATEMATYCZNYCH - CZĘŚĆ II

Maszyny uniwersalne

Komentarz do wykładu dla studentów matematyki

ROZDZIAŁ III

Maszyny uniwersalne pierwszego rodzaju.

§ 7. Meta-język oraz język wewnętrzny maszyny.

Zanim przystąpimy do określenia maszyny uniwersalnej pierwszego rodzaju wprowadzimy najpierw pojęcie języka wewnętrznego i meta-języka maszyny.

Jeżeli Σ jest alfabetem maszyny $\mathcal{M}$, to jej językiem wewnętrznym jest $\Sigma_w \subseteq \Sigma^*$. Językiem wewnętrznym jest więc pewien podzbiór słów zbioru wszystkich słów nad alfabetem Σ. Mówiąc jeszcze inaczej język wewnętrzny tworzą te słowa nad alfabetem maszyny, które możemy wpisywać w pamięci maszyny. W szczególnym przypadku $\Sigma_w = \Sigma^*$.

Do opisu działania maszyny stosujemy też pewien język. Elementami tego języka są formuły przedstawiające instrukcje. W języku tym musimy też umieć opisać działanie pamięci oraz sterowanie maszyny. Język służący do opisu działania maszyny będziemy nazywali meta-językiem tej maszyny.

Ogólnie biorąc język, w którym opisujemy problem rozwiązywany przez maszynę i który to opis może być zapisany w pamięci maszyny - nazwiemy językiem wewnętrznym maszyny. Natomiast język, którym posługujemy się przy opi-

Tak przedstawione sterowanie będziemy nazywali <u>prog-</u>
<u>ramem</u> . Każdy wiersz programu jest więc piątką uporządkowaną

$$\langle j, r_{i_j}, K_{i_j}, n_{i_j}, n'_{i_j} \rangle \quad , \text{gdzie}$$

$$1 \leq j \leq p.$$

Elementy $\langle r_{i_j}, K_{i_j}, n_{i_j}, n'_{i_j} \rangle$ będziemy nazywali <u>instrukcją wewnętrzną maszyny uniwersalnej</u> lub krótko instrukcją, zaś liczbę j numerem tej instrukcji. A więc wiersz składa się z instrukcji wewnętrznej oraz jej numeru.

Jeżeli $\varrho = \langle r, K, n, n' \rangle$ jest instrukcją wewnętrzną, to jej numer będziemy oznaczali przez $A(\varrho)$, zaś r_ϱ, K_ϱ, n_ϱ, n'_ϱ będą oznaczały odpowiednio kolejne elementy instrukcji ϱ .

W opisie maszyny uniwersalnej będą występowały trzy rodzaje instrukcji :

1° Meta-instrukcje maszyn klasy $\mathcal{M}$; będziemy nazywali je meta-instrukcjami rzędu zero.

2° Meta-instrukcje maszyny uniwersalnej dla klasy maszyn $\mathcal{M}$. Będziemy je nazywali meta-instrukcjami rzędu 1.

3° Instrukcje wewnętrzne maszyny uniwersalnej

$$\varrho = \langle r, K, n, n' \rangle ,$$

gdzie r jest znaną instrukcją -meta rzędu zero.

Instrukcje $\varrho = \langle r,k,n,n' \rangle$ można czytać następująco: "Wykonaj meta-instrukcję rzędu zero r i sprawdź warunek K ; jeżeli warunek K jest spełniony wykonaj następnie instrukcje o numerze n, jeżeli warunek K nie jest spełniony, wykonaj następnie instrukcje o numerze n'".

2. <u>Pamięć maszyny uniwersalnej</u>.

Adresami tej pamięci będą liczby naturalne $A = \{0,1,\ldots\}$

Alfabet Σ będzie zawierał wszystkie symbole potrzebne do zapisu programów, tj. liczby naturalne, symbole potrzebne do zapisu warunków oraz symbole potrzebne do zapisu meta-instrukcji rzędu zero.

Zbiór wskaźników będzie zawierał jeden wskaźnik s , zwany wskaźnikiem rozkazów.

Wprowadzimy następujący warunek w pamięci maszyny uniwersalnej :

$$K_1 : el(s) = STOP,$$

gdzie STOP oznacza instrukcję wewnętrzną postaci $\langle x, i_0 \rangle$; i_0 - tożsamościowa meta-instrukcja rzędu zero, x - końcowy punkt programu.

Nie będziemy ściśle definiować funkcji wejścia i wyjścia dla pamięci maszyny uniwersalnej, zależą one bowiem od konkretnych maszyn. Stwierdzimy tylko ogólnie, że aby maszyna uniwersalna mogła "naśladować" działanie którejś z maszyn $\mathcal{M}_i$ należy sterowanie (raczej program) odpowiednio umieścić w pamięci maszyny uniwersalnej. Program umieścimy w ten sposób, że każda instrukcja wewnętrzna ϱ zostanie umieszczona w pamięci maszyny uniwersalnej pod adresem $A(\varrho)$[1].

1) Odstąpiliśmy tu od przyjętej poprzednio zasady, że w

Przyjmiemy ponadto, że dla każdego $x > p$, $c_i(x) = c_u(x)$, gdzie p jest liczbą wierszy programu, c_i - zawartością maszyny $\mathcal{M}_i$, zaś c_u - zawartością pamięci maszyny uniwersalnej. Natomiast dla każdego $x \leq p$ $c_i(x) = \Lambda$.

Przyjęte założenia wymagają, aby w programie nie występowały adresy mniejsze od p. Powodowałoby to bowiem, że sam program mógłby być pod wpływem własnych instrukcji zniszczany [1].

3. Meta-instrukcje rzędu 1.

Do opisu działania maszyny uniwersalnej będziemy potrzebowali dwu meta-instrukcji rzędu 1. Pierwsza z nich, to instrukcja tożsamościowa I_0. Drugą oznaczymy przez R i zdefiniujemy następująco:

$$R(m) = \langle c', l' \rangle = m' \text{ , gdzie}$$

$$c' = c'_\varsigma$$

$$l'(s) = \begin{cases} n_\varsigma & \text{, jeżeli } c' \in K_\varsigma \\ n'_\varsigma & \text{, jeżeli } c' \notin K_\varsigma \end{cases}$$

gdzie $\varsigma = c\ l(s)$, zaś c'_ς oznacza zawartość otrzymaną w wyniku zastosowania instrukcji r_ς do stanu pamięci m.

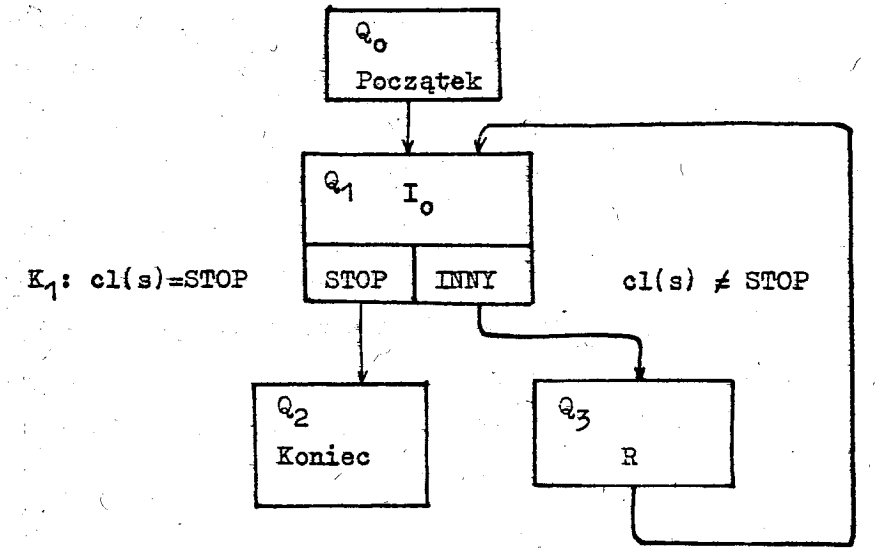
---

każdym miejscu pamięci może być zapisany tylko jeden symbol alfabetu Σ.

1) W przyszłości z tego założenia zrezygnujemy.

4. Sterowanie maszyny uniwersalnej.

Sterowanie to jest bardzo proste. Stany sterowania są oznaczane dużymi literami.



Sterowanie maszyny uniwersalnej sprawdza najpierw, czy odczytywany rozkaz jest rozkazem STOP. Jeżeli tak jest, to maszyna kończy działanie przechodząc do stanu Q_2. Jeżeli nie - sterowanie przechodzi do stanu Q_3 wykonując odpowiedni rozkaz programu i wyznaczają następny rozkaz, który ma być wykonany.

Bardziej szczegółowo rozpatrzymy działanie sterowania w następnym paragrafie przy omawianiu przykładów maszyn uniwersalnych.

§ 10. Przykłady maszyn uniwersalnych.

1. Trójadresowa maszyna uniwersalna.

Trójadresowa maszyna uniwersalna jest maszyną uniwersalną dla klasy maszyn trójadresowych. Jako klasę maszyn trójadresowych możemy przyjąć maszyny, które określiliśmy w poprzednim rozdziale. Założymy jedynie dodatkowo, że w instrukcjach wewnętrznych może występować tylko jeden warunek K_2 określony następująco :

Jeżeli $ca_3 = C$, powiemy, że warunek K_2 jest spełniony, gdy zaś $ca_3 \neq 0$, warunek K_2 jest niespełniony.

Ponieważ warunek jest jeden, można go w instrukcji wewnętrznej nie podawać rozumiejąc, że jest on zawsze sprawdzany. Tak więc każda instrukcja wewnętrzna będzie miała postać :

$$< f, a_1, a_2, a_3, n, n' > ,$$

gdzie f jest jedną z operacji określonych w poprzednim paragrafie ; a_1, a_2, a_3 - adresami obu argumentów operacji i wyniku.

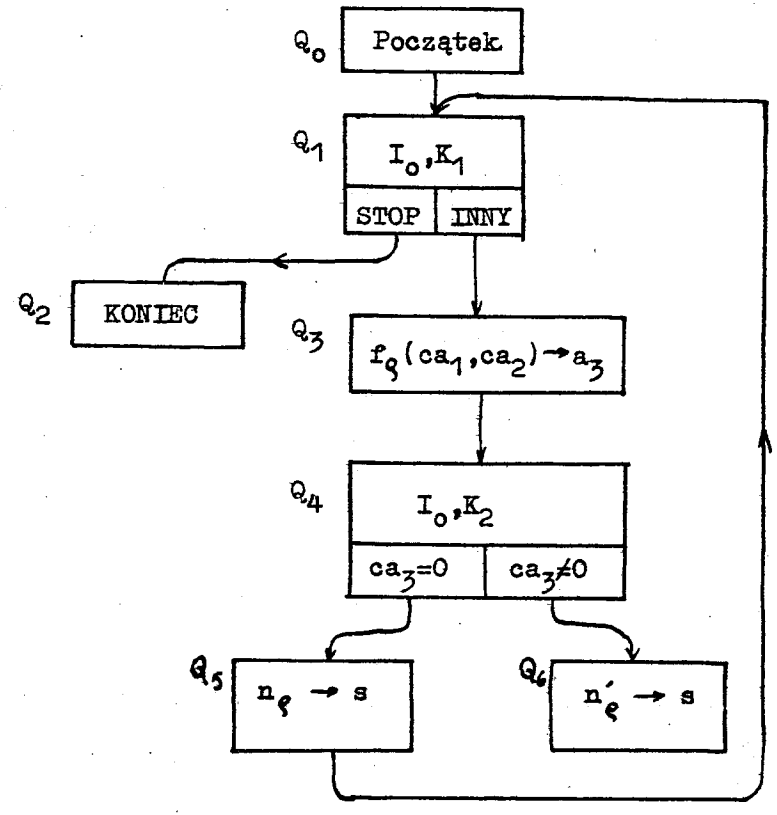
Meta-instrukcje tej maszyny uniwersalnej możemy więc określić :

$$R_3(m) = m' = < c', l' > \quad , \text{ gdzie}$$

$$c'(x) = \begin{cases} f_\varphi(ca_1, ca_2) & , \text{ jeżeli } x = a_3 \\ cx, & \text{jeżeli } x \neq a_3 \end{cases}$$

$$l'(s) = \begin{cases} n_\varphi & , \text{ jeżeli } c'\, a_3 = 0 \\ n'_\varphi & , \text{ jeżeli } c' a_3 \neq 0. \end{cases}$$

Sterowanie tej maszyny możemy przedstawić następująco :



Meta-instrukcję R_3 przedstawiliśmy tu nieco dokładniej przedstawiając ją jako złożenie czterech meta-instrukcji :

$$f_\varphi(ca_1, a_2) \to a_3$$
$$I_0$$
$$n_\varphi \to s$$
$$n'_\varphi \to s.$$

Pierwsza z nich powoduje wykonanie operacji określonej w instrukcji wewnętrznej ; druga - sprawdza spełnie-

nie warunku ; następne dwie powodują odpowiednie ustawienie wskaźników rozkazów s.

Zwróćmy uwagę, że w instrukcjach programu występuje tylko jeden warunek K_2 : $ca_3 = 0$, zaś w sterowaniu - jeszcze warunek K_1 sprawdzający, czy rozpatrywana instrukcja jest instrukcją STOP , tj. el(s) $\leq$ STOP.

2. Dwuadresowa maszyna uniwersalna.

Jako maszyny dwuadresowe przyjmiemy również maszyny określone również w poprzednim rozdziale [1]. Warunek K_2 będzie obecnie spełniony, jeżeli $ca_2 = 0$, gdy $ca_2 \neq 0$, to oczywiście warunek K_2 jest niespełniony. Instrukcje wewnętrzne będą miały obecnie postać :

$$< f, a_1, a_2, n, n' > ,$$

zaś meta-instrukcję maszyny uniwersalnej określimy w ten sposób :

$$R_2(m) = m' = < c', l' > \quad , \text{gdzie}$$

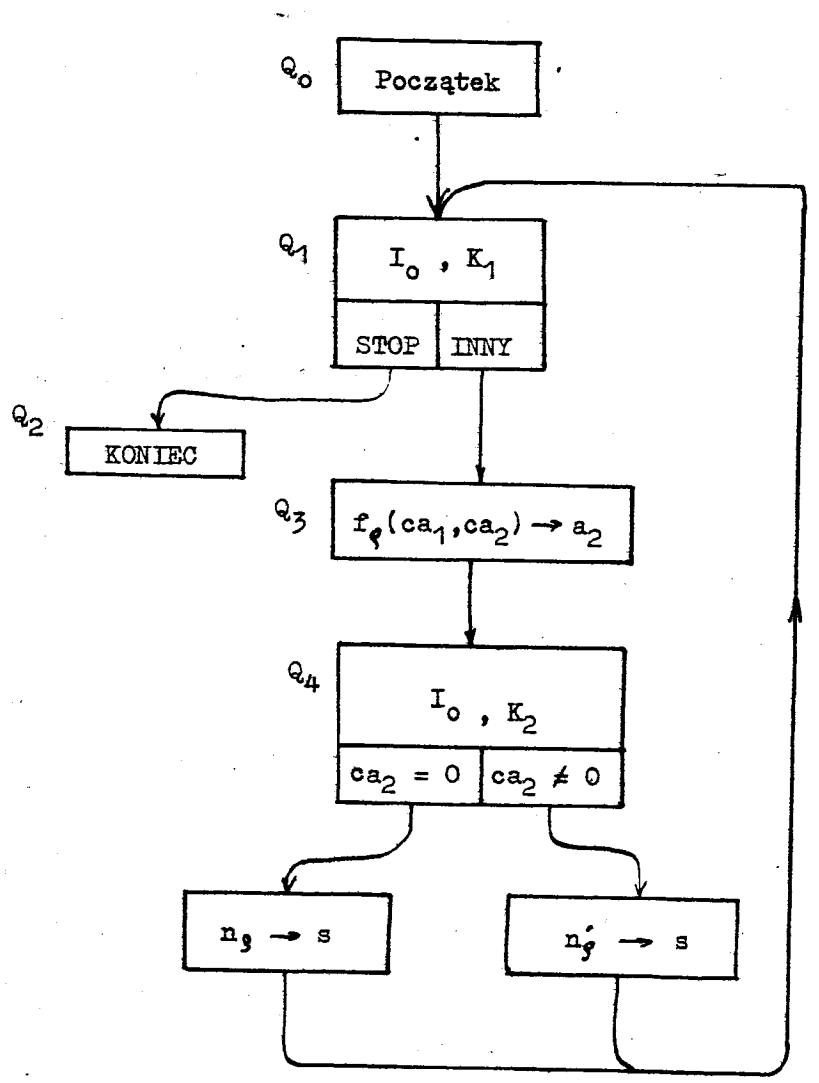
$$c'(x) = \begin{cases} f_\varphi (ca_1, ca_2) \rightarrow a_2, & \text{jeżeli } x = a_2 , \\ c(x), & \text{jeżeli } x \neq a_2. \end{cases}$$

$$l'(s) = \begin{cases} n , & \text{jeżeli } c'a_2 = 0, \\ n', & \text{jeżeli } c'a_2 \neq 0. \end{cases}$$

Sterowanie dwuadresowej maszyny uniwersalnej będzie miało więc postać :

---

1) Dla uproszczenia pominiemy tu instrukcję $\leftarrow a_1, a_2$,



3. Jednoadresowa maszyna uniwersalna.

Przyjmiemy poprzednio podane definicje maszyny jednoadresowej ustalając, że warunek K_2 ma obecnie postać c 0 = 0, zaś instrukcja wewnętrzna jest zredukowana do postaci :

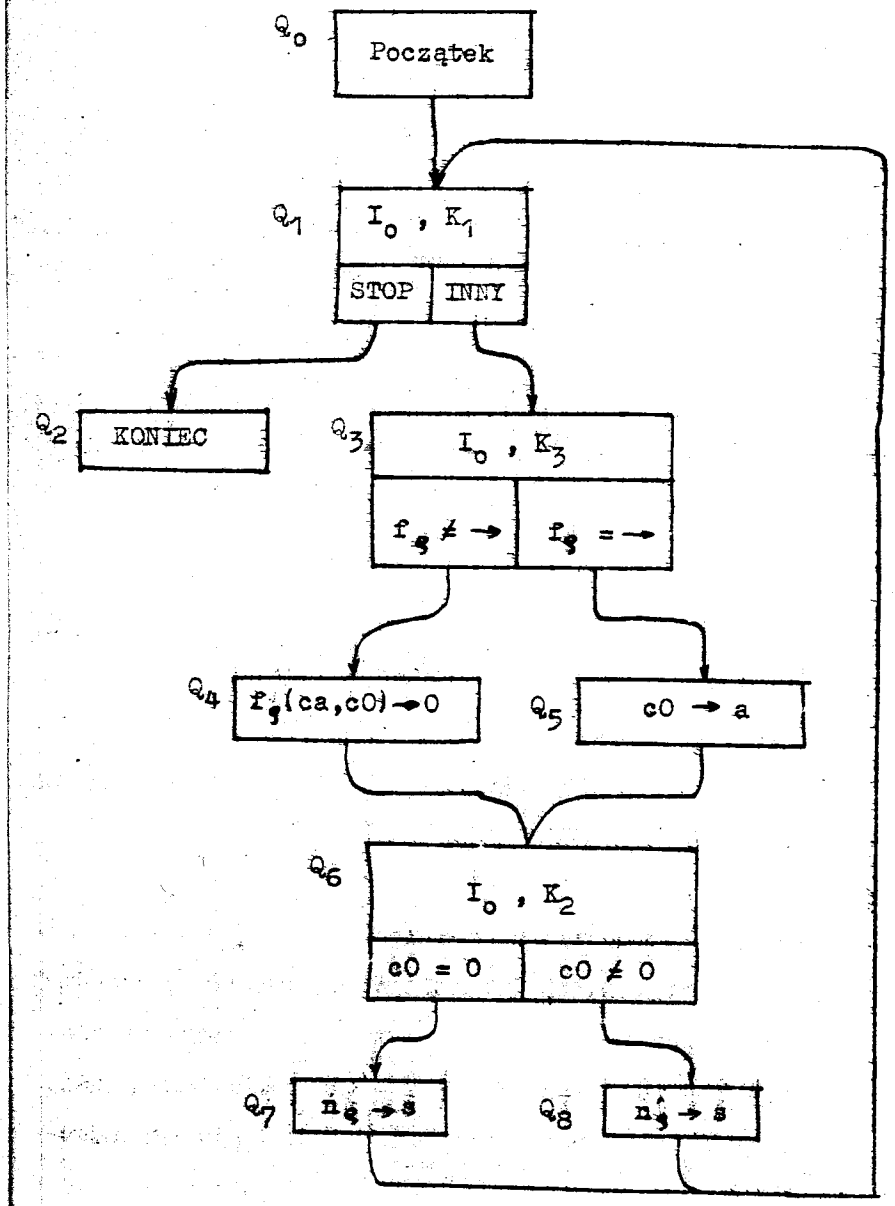
$$< f, a, n, n' > \ ,$$

W maszynie jednoadresowej wprowadzimy jeszcze jeden warunek : $K_3 : f_\varphi = \leftarrow$. Możemy teraz określić meta-instrukcje dla tej maszyny jako :

$$R_1(m) = m' = < c', l' > \quad , \text{gdzie}$$

$$c'(x) = \begin{cases} \left.\begin{array}{ll} f_\varphi(ca,c0), & \text{jeżeli } x = 0 \\ c(x), & \text{jeżeli } x \neq 0 \end{array}\right\} & \text{dla } f_\varphi \neq \rightarrow \\ \left.\begin{array}{ll} c0, & \text{jeżeli } x = a \\ c(x), & \text{jeżeli } x \neq a \end{array}\right\} & \text{dla } f_\varphi = \rightarrow \end{cases}$$

Sterowanie uniwersalnej maszyny jedno adresowej będzie więc miało postać :



4. Bezadresowa maszyna uniwersalna.

Jest to maszyna uniwersalna dla klasy maszyn bezadresowych [1]. Dla ustalenia uwagi przyjmiemy maszyny bezadresowe omówione w poprzednim rozdziale.

Wskaźnik w_1 dla tych maszyn będziemy interpretować jako adres 1 w pamięci maszyny uniwersalnej. W związku z powyższym instrukcje maszyn bezadresowych należy rozumieć w ten sposób, jak gdyby w_1 było adresem 1, zaś funkcję 1 należy zastąpić przez funkcje c. Wtedy np. instrukcje

$$c\,1\,w_1 + c0 \longrightarrow 0$$
$$c\,1\,w_1 + 1 \longrightarrow 1\,w_1$$
$$1\,w_1 + 1 \longrightarrow w_1$$

przyjmą postać

$$c\,c(1) + c(0) \longrightarrow 0$$
$$c\,c(1) + 1 \longrightarrow c(1)$$
$$c(1) + 1 \longrightarrow 1.$$

Oczywiście programy musimy umieszczać teraz od miejsca 2 w pamięci maszyny uniwersalnej.

---

1) Możemy tu również mówić o uniwersalnych maszynach bezadresowych w innym sensie. Jeżeli meta-instrukcje rzędu 1 są bezadresowe, to maszyna uniwersalna jest bezadresowa. Do tej pory omawiane maszyny są więc w tym drugim sensie adresowe, gdyż w ich meta-instrukcjach występowały adresy.

Warunkiem K_2 będzie $c(0) = 0$; instrukcja wewnętrzna zaś będzie miała postać :

$$\langle f, n, n' \rangle .$$

Meta-instrukcję dla tej maszyny określimy w ten sposób :

$$R_b(m) = m' = \langle c', 1' \rangle$$

$$c'(x) = \begin{cases} f_\varphi(c(0), cc(1)), \text{ jeżeli } x = 0 \\ c(x), \text{ jeżeli } x \neq 0 \end{cases} \quad \text{dla } f_\varphi = +,-,\cdot,/,\leftarrow$$

$$\begin{cases} f_\varphi(c(0), cc(1)), \text{ jeżeli } x = c(1) \\ c(x), \text{ jeżeli } x \neq c(1) \end{cases} \quad \text{dla } f_\varphi = N, P, \rightarrow, 0$$

$$\begin{cases} c(1) \mp 1, \text{ jeżeli } x = 1 \\ c(x), \text{ jeżeli } x \neq 1 \end{cases} \quad \text{dla } f_\varphi = 1, p$$

$$1'(s) = \begin{cases} n, \text{ jeżeli } c(0) = 0 \\ n', \text{ jeżeli } c(0) \neq 0 . \end{cases}$$

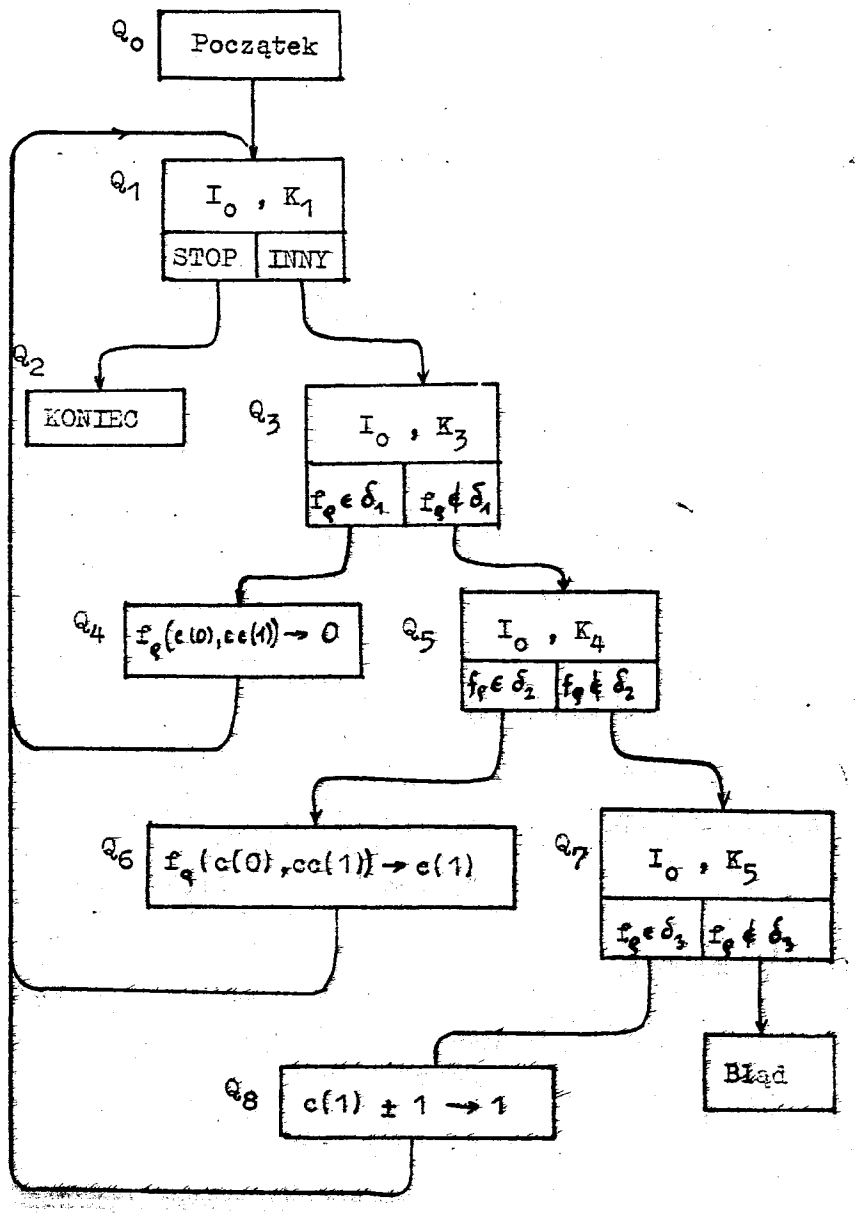
Wprowadzimy jeszcze dodatkowo trzy warunki :

$K_3 : f_\varphi \in \delta_1$, gdzie $\delta_1 = +,-,\cdot,/,\leftarrow$

$K_4 : f_\varphi \in \delta_2$, gdzie $\delta_2 = N, P, \rightarrow, 0$

$K_5 : f_\varphi \in \delta_3$, gdzie $1, p$.

Sterowanie zaś będzie :



§ 11. Własności maszyn uniwersalnych.

Maszyny uniwersalne $\mathcal{M}$ i $\mathcal{M}'$ są równoważne symbolicznie $\mathcal{M} \sim \mathcal{M}'$ wtedy i tylko wtedy, gdy $F(\mathcal{M}) = F(\mathcal{M}')$, gdzie $F(\mathcal{M})$ oznacza klasę funkcji obliczalnych przez maszynę $\mathcal{M}$.

Można dowieść następujących twierdzeń o maszynach uniwersalnych :

Twierdzenie 1. Dla każdej maszyny uniwersalnej $\mathcal{M}$ istnieje równoważna jej maszyna $\mathcal{M}'$ taka, że każda instrukcja wewnętrzna maszyny $\mathcal{M}'$ ma postać

$$\varphi = \langle r, K, A(\varphi) + 1, n' \rangle \quad ^{1)}.$$

Twierdzenie 2. Dla każdej uniwersalnej maszyny $\mathcal{M}$ istnieje równoważna jej maszyna uniwersalna $\mathcal{M}'$ taka, że każda instrukcja wewnętrzna maszyny $\mathcal{M}'$ ma postać

$$\varphi = \langle r, K, n, A(\varphi) + 1 \rangle \quad .$$

Twierdzenie 3. Dla każdej maszyny uniwersalnej $\mathcal{M}$ istnieje jej równoważna maszyna uniwersalna $\mathcal{M}'$ taka, że każda instrukcja wewnętrzna maszyny $\mathcal{M}'$ ma jedną z postaci :

$$\varphi = \langle r, A(\varphi) + 1, A(\varphi) + 1 \rangle \quad ,$$

bądź

$$\varphi = \langle r_0, K, n, n' \rangle \quad ,$$

---

1) tzn w przypadku spełnienia warunku sterowanie p
chodzi do wykonania instrukcji w następnym wierszu programu.

gdzie r_0 jest instrukcją tożsamościową.

Twierdzenie to dzieli instrukcje wewnętrzne na dwie klasy. Do pierwszej klasy należą instrukcje powodujące wykonanie jakiejś operacji, po czym następuje przejście do wykonania następnej instrukcji wewnętrznej bez sprawdzania warunku.

Instrukcje te są nazywane bezwarunkowymi.

Do drugiej klasy należą instrukcje, których się operacją tożsamościową i powoduje sprawdzenie warunku , który wyznacza następną instrukcję wewnętrzną. Instrukcje tego rodzaju nazwiemy instrukcjami przejścia. Instrukcje bezwarunkowe możemy zapisać krótko

$$\varrho = r,$$

zaś instrukcje przejścia

$$\varrho = \langle K, n, n' \rangle .$$

Zwróćmy uwagę, że jeżeli w instrukcji przejścia $n=n'$, to instrukcja ta właściwie nie zależy od warunku K; możemy ją więc krótko zapisać

$$\varrho = n.$$

Instrukcję tę będziemy nazywać przejściem bezwarunkowym. Powoduje ona przejście do wykonywania instrukcji o numerze n . Jeżeli zaś $n \neq n'$, to instrukcję taką nazwiemy przejściem warunkowym.

Zachodzi również

Twierdzenie 4. Dla każdej maszyny uniwersalnej $\mathcal{M}$ istnieje równoważna jej maszyna uniwersalna $\mathcal{M}'$, któ-

rej instrukcje przejścia są postaci :

$$\varrho = \langle K, A(\varrho) + 1, n' \rangle .$$

Podobnie zachodzi

Twierdzenie 5. Dla każdej maszyny uniwersalnej $\mathcal{M}$ istnieje równoważna jej maszyna uniwersalna $\mathcal{M}'$, której instrukcje przejścia są postaci :

$$\varrho = \langle K, n, A(\varrho) + 1 \rangle .$$

Oba rodzaje instrukcji przejścia warunkowego możemy krótko zapisać

$$\varrho = \langle K, n \rangle .$$

Maszyna uniwersalna $\mathcal{M}$ jest zawarta w maszynie uniwersalnej $\mathcal{M}'$, symbolicznie $\mathcal{M} \subset \mathcal{M}'$ wtedy i tylko wtedy, gdy $\mathcal{M}$ i $\mathcal{M}'$ posiadają jednakową pamięć oraz dla każdej instrukcji wewnętrznej r maszyny $\mathcal{M}$ istnieje program p w języku wewnętrznym maszyny $\mathcal{M}'$ taki , że dla dowolnego stanu pamięci m

$$r(m) = p(m).$$

Twierdzenie 6. Jeżeli $\mathcal{M} \subset \mathcal{M}'$ oraz $\mathcal{M}' \subset \mathcal{M}$, to

$$\mathcal{M} \approx \mathcal{M}'.$$

R O Z D Z I A Ł IV

Maszyny uniwersalne drugiego rodzaju

§ 12. Realizacja języka Łukasiewicza.

a) Maszyna z pamięcią wyników częściowych.

Maszyna ta posiada następujące pamięci :

D - pamięć danych

F - pamięć formuły

S - pamięć wyników częściowych

Z - skorowidz zmiennych.

Pamięci te są takie same, jak w maszynie realizującej język nawiasowy z tą jedynie różnicą, że Σ_F nie zawiera nawiasów (,) .

Warunki będą teraz tylko trzy :

K_1 jeżeli c(f) = zmienna

K_2 jeżeli c(f) = symbol operacji

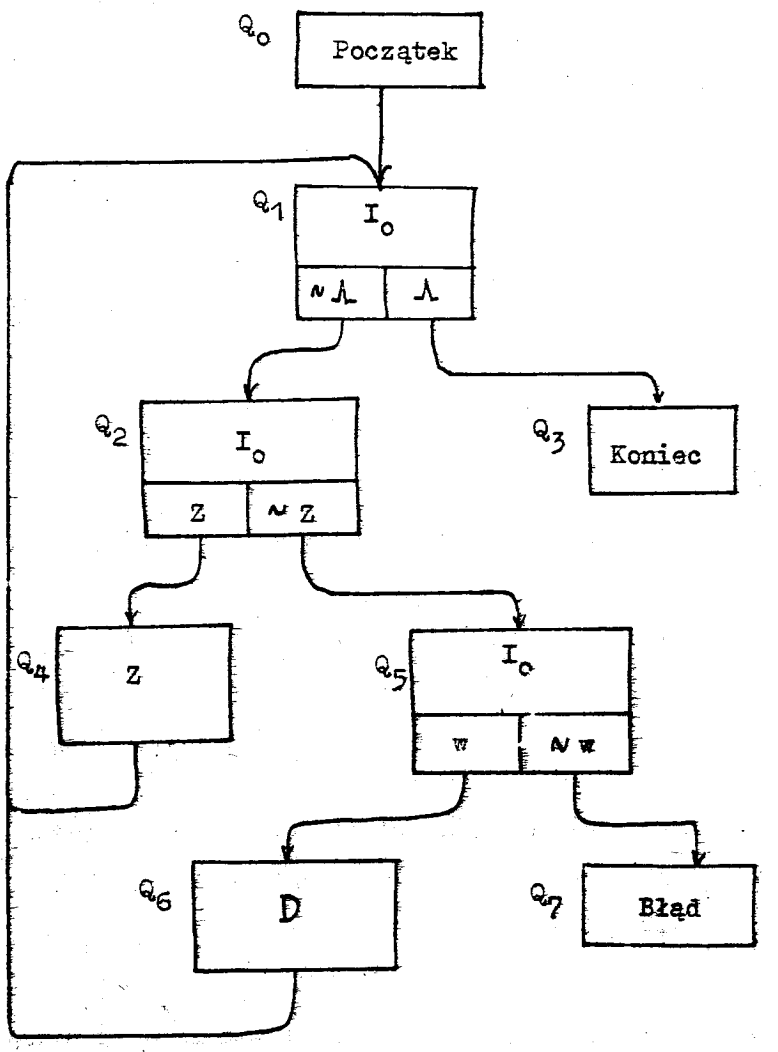
K_3 jeżeli c(f) = Λ .

Jako zbiór meta-instrukcji przyjmiemy

$$I = \{ I_0, Z, D \} .$$

Wszystkie te meta-instrukcje są takie same jak w maszynie nawiasowej z pamięcią wyników częściowych.

Sterowanie będzie miało postać :



Działanie tej maszyny jest oczywiste.

F	D	Z	S
x	2	x	
y	4	xy	
+		*	6
z	1	* z	
•		*	6
x	2	* *	6
z	1	* xz	6
-		* *	6,1
/		*	6

b) Maszyna z pamięcią roboczą.

Maszyna ta posiada tylko trzy pamięci :

D - pamięć danych

F - pamięć formuły

R - pamięć robocza.

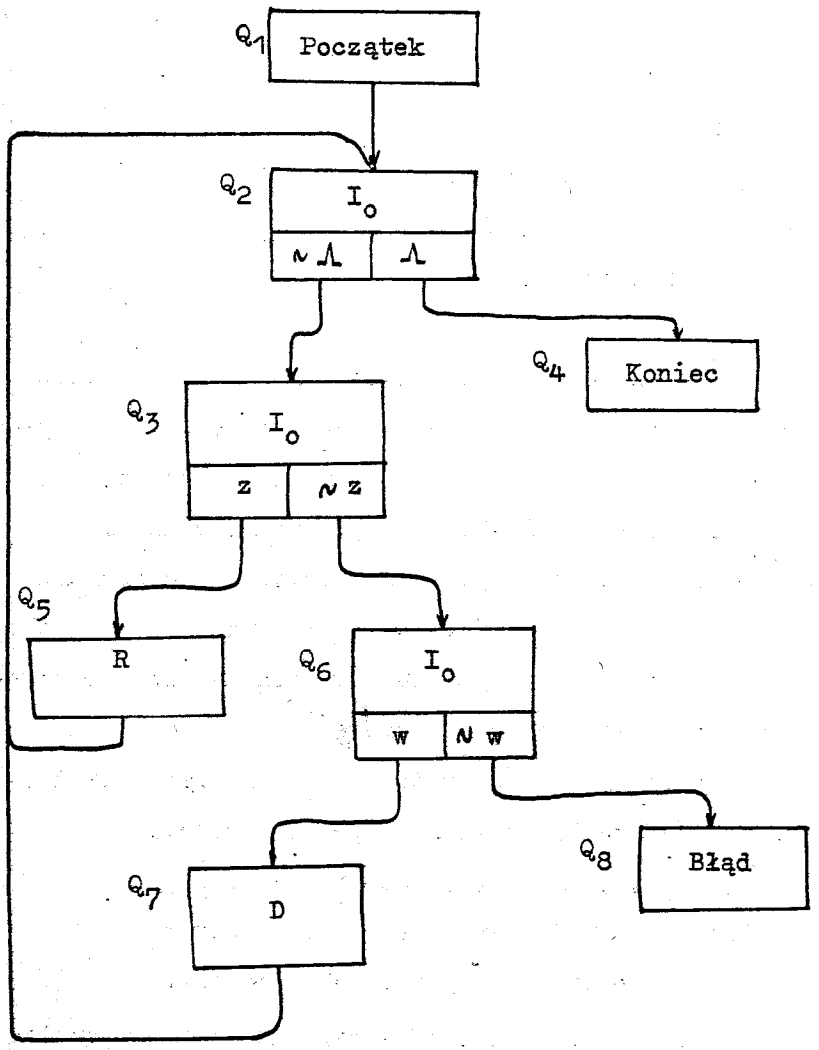
Są one identyczne jak w odpowiedniej maszynie nawiasowej. Warunki są takie same jak w maszynie poprzedniej, zaś meta-instrukcje

$$I = I_o , R , D .$$

Ich znaczenie jest takie samo jak w maszynie nawiaso-

wej z pamięcią roboczą.

Sterowanie będzie miało postać :



§ 13. Realizacja języka nawiasowego.

a) Maszyna z pamięcią wyników częściowych.

W maszynie tej wyróżniamy następujące pamięci :

D - pamięć danych

F - pamięć formuły

S - pamięć wyników częściowych

Z - skorowidz zmiennych

O - skorowidz operacji

które kolejno omówimy.

Pamięć danych służy do magazynowania wartości zmiennych występujących w formule, które mamy liczyć. Przyjmiemy $\Sigma_D = \{0,1,\dots\}$ [1], zaś A_D jest zbiorem zmiennych, które dopuściliśmy w pisaniu formuł oraz $W_D = \{d\}$. D jest więc pamięcią adresową, adresowana zmiennym i magazynująca wartości zmiennych. Funkcje wejścia i wyjścia są takie, jak to podawaliśmy przy opisywaniu pamięci adresowych w rozdziale I.

Pamięć formuły F służy do przechowywania formuły, której wartość ma być obliczana przez maszynę.

$$\Sigma_F = \{(,),\text{zmienne},\text{symbole operacji}\} .$$

W dalszym ciągu dowolną zmienną będziemy oznaczali przez z , zaś operacje - przez ω (omega).

$A_F = \{0,1,\dots\}$, a więc adresami są liczby naturalne

zaś $W_F = \{f\}$. $\Sigma_D = \{0,1\}$.

1) Gdybyśmy liczyli np. wartości formuł logicznych, to

Pamięć F jest liniowa [1] . Pamięć S służy do przechowywania wyników częściowych otrzymywanych w trakcie liczenia.

$A_S = \Sigma_S = \{0,1,\dots\}$, zaś $W_S = \{s\}$. Jest to pamięć rewersyjna (patrz rozdział I).

Pamięci Z oraz O mają charakter pomocniczy. Nazywamy je skorowidzami i służą one do magazynowania zmiennych oraz symboli operacji w trakcie liczenia. Obie te pamięci są rewersyjne.

$\Sigma_Z = \{\text{zmienna}\}$, $A_Z = \{0,1,\dots\}$, $W_O = \{z\}$.

$\Sigma_O = \{\text{symbole operacji}\} \cup *$, $A_O = \{0,1,\dots\}$, $W_O = \{s\}$.

Zbiorem stanów całej pamięci będzie iloczyn

$$M = M_D \times M_F \times M_S \times M_Z \times M_O ;$$

gdzie poszczególne elementy iloczynu są stanami odpowiednich pamięci oznaczonych przez wskaźnik u dołu.

Będziemy rozróżniali następujące warunki :

K_1 jeżeli c(f) = (

K_2 jeżeli c(f) =)

K_3 jeżeli c(f) = zmienna

K_4 jeżeli c(f) = symbol operacji

K_5 jeżeli c(f) = Λ .

A więc warunek polega na sprwdzeniu w pamięci F zawartości miejsca l(f) .

Obecnie opiszemy meta-instrukcje tej maszyny. Zbiór

---

1) Funkcje wejścia i wyjścia patrz rozdział I.

meta-instrukcji będzie następujący : $I = \{\mathcal{I}_o, \mathcal{P}, \mathcal{Z}, \mathcal{O}, \mathcal{D}\}$
(Nie należy mylić tych oznaczeń z oznaczeniami pamięci).

$\mathcal{I}_o$ - jest instrukcją tożsamościową.

$\mathcal{P}$ - jest instrukcją przesuwania o jedno miejsce w prawo wskaźnika f w pamięci F. Możemy więc tę instrukcję zapisać w ten sposób :

$$\mathcal{P}(m) = m' = \langle m_D , m'_F , m_S , m_Z , m_O \rangle ,$$

gdzie $m = \langle m_D , m_F , m_S , m_Z , m_O \rangle$, zaś

$$m'_F = \langle c_F , l'_F \rangle \quad \text{natomiast}$$

$$l'_F(f) = l_F(f) + 1 \quad ^{1)} \qquad l'(f) = l(f) + 1$$

$\mathcal{Z}$ - powoduje wpisanie zmiennej wskazywanej przez wskaźnik f do skorowidza zmiennych i przesuwanie wskaźnika f w prawo.

$$\mathcal{Z}(m) = m' = \langle m_D , m'_F , m_S , m'_Z , m_O \rangle ,$$

$$m'_F = \langle c_F , l'_F \rangle , \quad \text{gdzie } l'(f) = l(f)+$$

zaś $m'_Z = I_r \left[c_F\, l_F(f), m_Z \right]$, gdzie I_r jest funkcją wejścia pamięci rewersyjnej. Dla uproszczenia w dalszym ciągu zamiast $c_F\, l_F(f)$ będziemy pisać $c(f)$ i podobnie dla innych wskaźników.

$\mathcal{O}$ - powoduje wpisanie symbolu operacji wksazywanej przez wskaźnik f do skorowidza operacji i prze-

---

1) Przyjmiemy raz na zawsze, że odpowiednie litery bez kreski u góry oznaczają poprzednie wartości stanów, a z kreseczką - stany po zastosowani instrukcji.

sunięcie f o jedno miejsce w prawo.

$\mathcal{O}(m) = m' = \langle m_D , m'_F , m_S , m_Z , m'_O \rangle$, gdzie m'_F jest w instrukcji poprzedniej, zaś

$$m'_O = I_r \left[c(f), m_O \right].$$

$\mathcal{D}$ - jest instrukcją wykonania działania. Powoduje ona wykonanie operacji na odpowiednich argumentach i umieszczenie wyniku w pamięci S.

$\mathcal{D}(m) = m' = \langle m'_D , m'_F , m'_S , m'_Z , m'_\sigma \rangle$

m'_F określona jak w poprzedniej instrukcji.

$$m'_Z = \langle c'_Z , l'_Z \rangle \text{ , gdzie}$$

$$c'_Z(x) = \begin{cases} * & \text{, jeżeli } x = l(z) - 1, \\ c_Z(x) & \text{, jeżeli } x \neq l(z) - 1 \end{cases}$$

$$l'(z) = l(z) - 1$$

$m'_o = \langle c_o, l'_o \rangle$ gdzie $l'(0) = l(0) - 1$.

Wreszcie

$$m'_S = I_r \left[\omega(n_1, n_2), m_S \right], \text{ gdzie } \omega = O_2(m_o) \quad ^{1)}$$

$$n_1 = \begin{cases} O_2 \left[O_2(O_1(m_Z)), m_D \right], & \text{jeżeli } O_2(O_1(m_Z)) \neq * \\ O_2(O_1(m_S)), & \text{jeżeli } O_2(O_1(m_Z)) = * . \end{cases}$$

---

1) Jeżeli

$$O(m) = \langle m', \sigma \rangle,$$

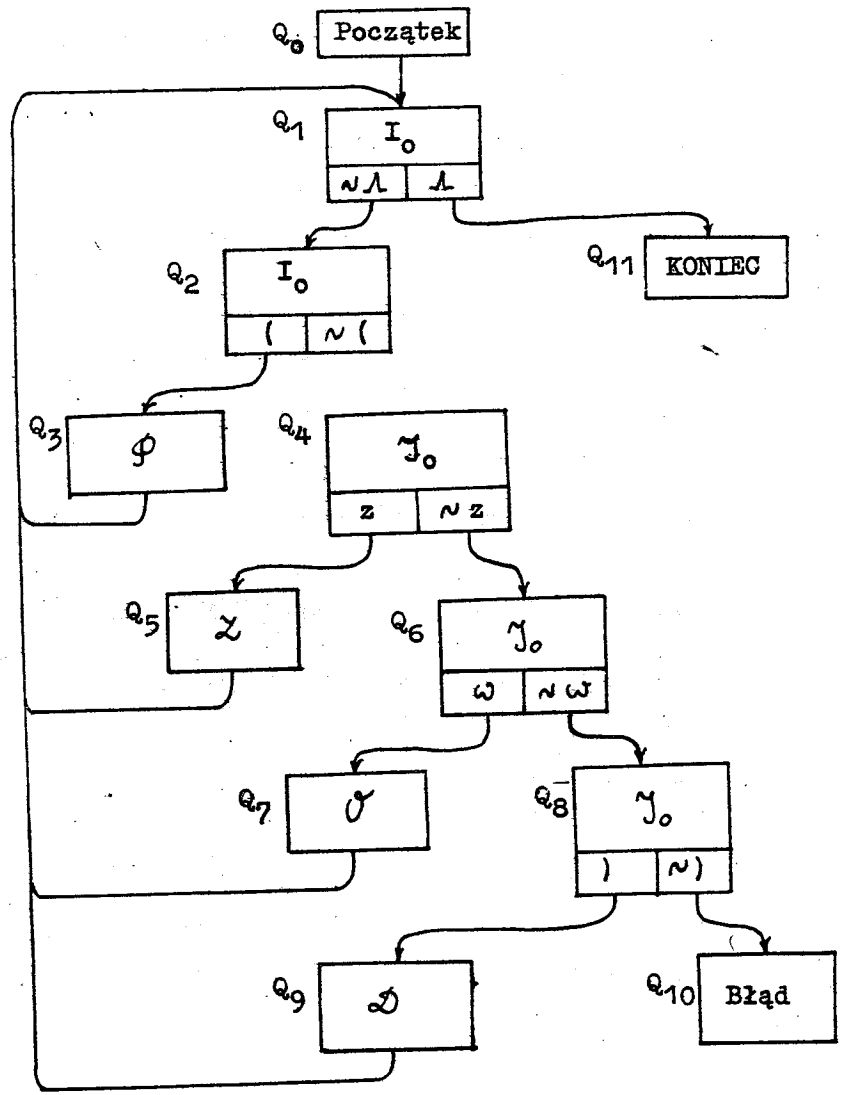
to zapiszemy funkcje $\mathcal{O}$ jako dwie funkcje

$$O_1(m) = m'$$

$$O_2(m) = \sigma$$

$$n_2 = \begin{cases} O_2(O_2(m_Z), m_D) & \text{, jeżeli } O_2(m_Z) \neq * \text{ ,} \\ O_2(m_S) & \text{, jeżeli } O_2(m_Z) = * \text{ .} \end{cases}$$

Możemy już teraz podać sterowanie tej maszyny.



Maszyna ta działa więc następująco : Obliczana formuła jest analizowana symbol po symbolu zaczynając od symbolu pierwszego.

Lewostronny nawias jest opuszczany i maszyna przechodzi do czytania następnego symbolu. Odczytanie zmiennej powoduje umieszczenie jej w skorowidzu zmiennych (na wierzchu stosu i przejście do czytania następnego symbolu formuły.

Odczytanie symbolu operacji powoduje zapisanie go do skorowidza operacji (na wierzchu stosu) i przejście do czytania następnego symbolu formuły. Odczytanie nawiasu prawostronnego powoduje wykonanie operacji znajdującej się na ostatnim miejscu w skorowidzu operacji. Argumenty tej operacji są wskazane przez dwie ostatnie pozycje skorowidza zmiennych. Jeżeli na pozycjach tych są zmienne, to argumenty są pobierane z miejsc wskazanych przez te zmienne z pamięci danych D. Jeżeli zaś w skorowidzu zmiennych na którymkolwiek z dwu ostatnich miejsc lub obu znajduje się * , to odpowiedni argument operacji jest pobierany z ostatniego miejsca pamięci wyników częściowych [1].

Wynik operacji jest wpisywany na ostatnie miejsca

---

1) Przez ostatnie miejsca w pamięci rewersyjnej rozumiemy miejsce wskazywane przez wskaźnik.

w pamięci wyników częściowych. Po zakończeniu operacji do skorowidza zmiennych wpisywany jest na ostatnie miejsce symbol * .

W przypadku odczytania symbolu pustego Λ maszyna kończy działanie.

F	D	Z	σ	S
(				
(				
(				
x	2	x		
+		x	+	
y	4	xy	+	
)		*		6
•		*	•	6
z	1	* z	•	6
)		*		6
/		*	/	6
(		*		6
x	2	* x	/	6
-		* x	/ -	6
z	1	*xz	/ -	6
)		**	/	6,1
)		*		6

b) Maszyna z pamięcią roboczą.

Podamy teraz drugi sposób obliczania wartości formuły nawiasowej. W maszynie tej będą następujące pamięci :

D - pamięć danych

F - pamięć formuły

R - pamięć robocza

σ - skorowidz operacji

Pamięci D, F, σ są identyczne jak w maszynie poprzedniej. Pamięć R jest pamięcią rewersyjną. Adresami i alfabetem w niej są liczby naturalne. Warunki w tej maszynie będą takie same jak w maszynie poprzedniej.

Meta-instrukcje tej maszyny będą następujące :

$I = \{\mathcal{I}_o, \mathcal{P}, \sigma, \mathcal{D}, \mathcal{R}\}$. $\mathcal{I}_o, \mathcal{P}, \sigma$ są identyczne jak w poprzedniej maszynie, zaś dwie pozostałe zdefiniujemy następująco :

$$\mathcal{D}(m) = m' = \langle m'_D , m'_F , m'_R , m'_\sigma \rangle$$

$$m'_F = \langle c_F , l'_F \rangle \text{ , gdzie}$$

$$l'(f) = l(f) + 1,$$

$$m'_o = \langle c_o , l'_o \rangle \text{ , gdzie}$$

$$l'(0) = l(0) - 1 ,$$

$$m'_R = I_r\left[\omega(n_1,n_2), m_R\right] \text{ , gdzie}$$

$$\omega = 0_2(m_o) ,$$

$$n_1 = 0_2(0_1(m_R))$$

$$n_2 = 0_2(m_R) .$$

Instrukcja wykonania działania $\mathcal{D}$ powoduje więc

wykonanie działania znajdującego się na ostatnim miejscu w skorowidzu operacji na dwu ostanich liczbach w pamięci roboczej i umieszczenie wyniku na miejscu przedostatniej liczby w R.

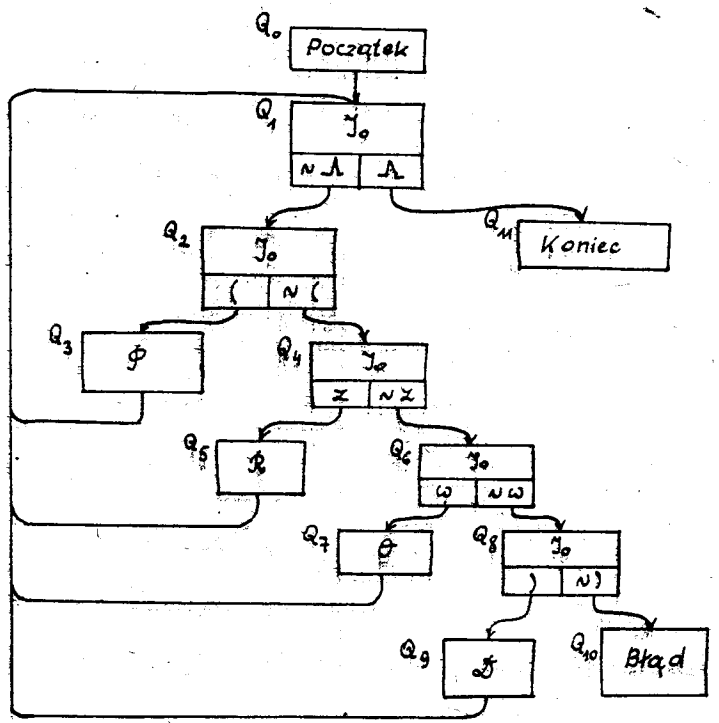
$$\mathcal{R}(m) = m' = \langle m'_D , m'_F , m'_R , m_\sigma \rangle$$

m'_F - jak wyżej

$$m'_R = I_r\{ O_2 [(O_2(m_F)), m_D], m_R \} .$$

Instrukcja ta powoduje więc wpisanie wartości zmiennej odczytanej w pamięci F do pamięci roboczej R.

Sterowanie będzie teraz miało postać :



Przykład :

F	D		R
(			
(			
(			
x	2		2
+		+	2
y	4	+	2,4
)			6
•		•	
z	1	•	6,1
)			6
/		/	6
(		/	6
x	2	/	6,2
-		/-	6,2
z	1	/-	6,2,1
)		/	6.1
)			6

Maszyna ta działa więc w ten sposób : w pamięci F czytana jest symbol po symbolu formuła. Nawias lewostronny jest pomijany. Odczytanie zmiennej powoduje wpisanie jej wartości do pamięci roboczej R. Odczytanie symbolu operacji powoduje wpisanie go do skorowidza operacji . Odczytanie nawiasu powoduje wykonanie ostatniego działa-

nia ze skorowidza operacji $\star$ na dwu ostatnich argumentach z pamięci roboczej i wpisanie wyniku na miejsce w pamięci R. Odczytanie symbolu pustego kończy działanie maszyny.

Spis rzeczy