AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Maszyna Turinga – obliczenia

Teoria automatów i języków formalnych

Dr inż. Janusz Majewski
Katedra Informatyki

Przykład (1)

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$
$F = \{q_5\}$
$\Gamma = \{1, 2, b\}$
$\Sigma = \{1\}$

δ:	b	1	2
q_0	q_1, 2, L	q_0, 1, R	
q_1	q_2, b, R	q_1, 1, L	q_1, 2, L
q_2		q_3, b, R	q_4, b, R
q_3	q_4, 1, R	q_3, 1, R	q_3, 2, R
q_4	q_1, 1, L	q_5, 1, R	
q_5		q_5, 1, R	

Start: $(q_0, \uparrow 11)$, Stop: $(q_0, 1111\uparrow)$

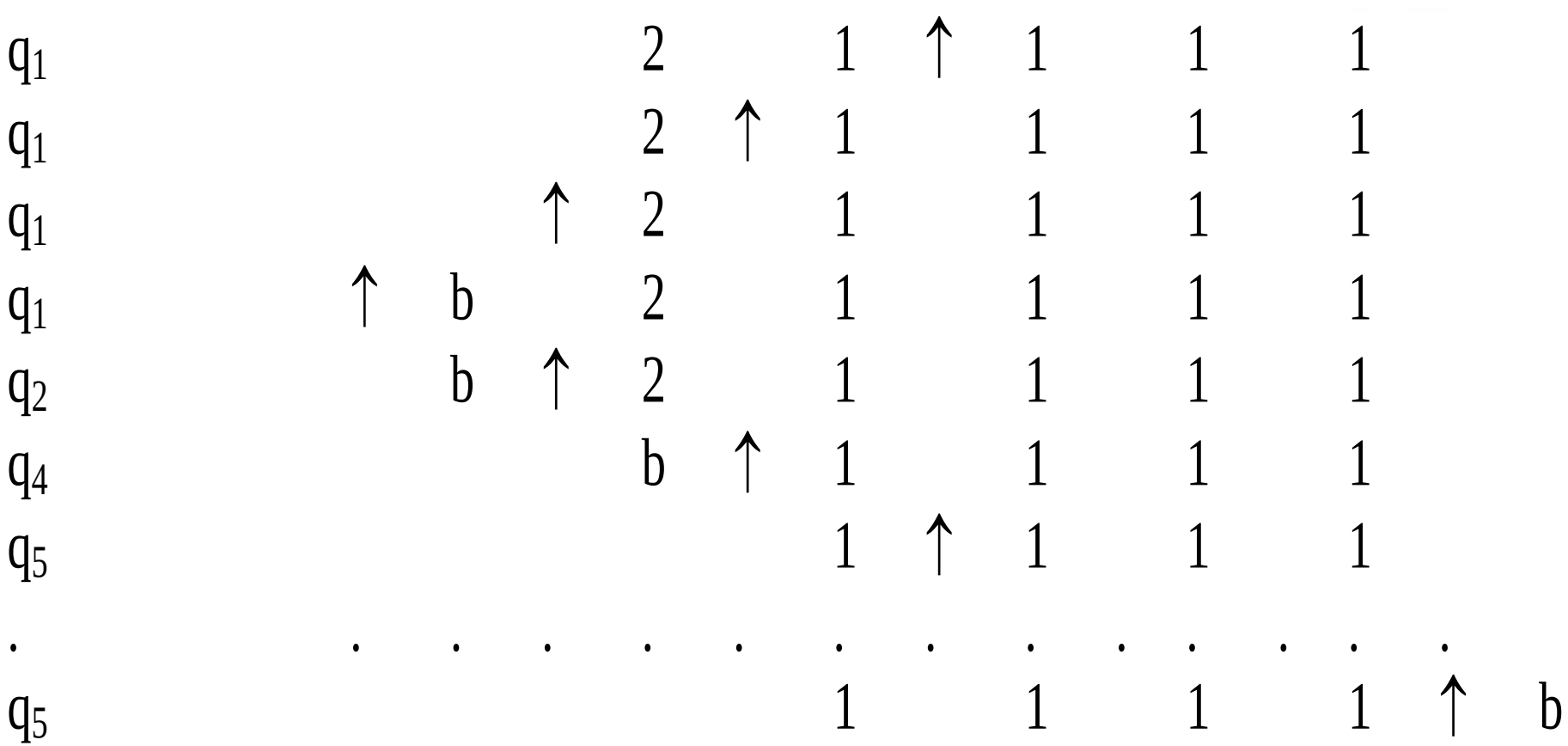
Przykład (2)

q_0			↑	1		1								
q_0				1	↑	1								
q_0				1		1	↑	b						
q_1				1	↑	1		2						
q_1			↑	1		1		2						
q_1	↑	b		1		1		2						
q_2		b	↑	1		1		2						
q_3				b	↑	1		2						
q_3						1	↑	2						
q_3						1		2	↑	b				
q_4						1		2		1	↑	b		
q_1						1		2	↑	1		1		

Przykład (3)

q_1						1	↑	2			1			1									
q_1				↑		1		2			1			1									
q_1		↑		b		1		2			1			1									
q_2				b	↑	1		2			1			1									
q_3						b	↑	2			1			1									
q_3								2	↑		1			1									
q_3								2			1	↑		1									
q_3								2			1			1	↑		b						
q_4								2			1			1			1	↑		b			
q_1								2			1			1	↑		1			1			

Przykład (4)


q1 2 1 ↑ 1 1 1
q1 2 ↑ 1 1 1 1
q1 ↑ 2 1 1 1 1
q1 ↑ b 2 1 1 1 1
q2 b ↑ 2 1 1 1 1
q4 b ↑ 1 1 1 1
q5 1 ↑ 1 1 1
. . . .
q5 1 1 1 1 ↑ b

Obliczalność w sensie Turinga

Obliczalność funkcji w sensie Turinga—definicja

$N = \{0,1,2,\ldots\}$ (zbiór liczb naturalnych z zerem)

Funkcję f

$f: (x_1,\ldots,x_k) \in N^k \mapsto N \ni f(x_1,\ldots,x_k), \quad k=1,2,\ldots$

nazywamy obliczalną w sensie Turinga jeżeli

$(\exists A \in \mathcal{A}_T)\ ((q_0, \uparrow 1^{x1} b 1^{x2} b \ldots b 1^{x2}) \succ^* (q, 1^{f(x1,\ldots,xk)} \uparrow))$

gdzie: $q \in F$, $\Sigma=\{1\}$, $\Gamma=\{1, b, \ldots\}$

Obliczalność w sensie Turinga

- W ogólnym przypadku funkcją obliczalną przez maszynę Turinga nazywamy odwzorowanie danych wejściowych w zbiór danych wyjściowych. Jeśli dla pewnych danych wejściowych obliczenie maszyny zatrzymuje się w stanie akceptującym to zakładamy, że wynik obliczenia zapisany jest na taśmie. W przeciwnym przypadku, gdy maszyna wykonuje nieskończone obliczenie albo kończy obliczenie w stanie nie będącym stanem akceptującym – zakładamy, że wartość funkcji jest nieokreślona.

Funkcje rekurencyjne

Funkcje rekurencyjne — definicja:

1. Funkcją rekurencyjną jest:
   a) $Z(x) = 0$ — zero
   b) $S(x) = x+1$ — następnik
   c) $I_{i,n}(x_1, \ldots, x_i, \ldots x_n) = x_i$ — projekcja (identyczność)
2. Jeśli $f_1, \ldots, f_n$ są funkcjami rekurencyjnymi m argumentów, g jest funkcją rekurencyjną n argumentów, to funkcją rekurencyjną jest

   $h(x_1, \ldots, x_m) = g(f_1(x_1, \ldots, x_m), \ldots, f_n(x_1, \ldots, x_m))$ — podstawienie

Funkcje rekurencyjne

3. Jeśli f jest funkcją rekurencyjną n argumentów, g jest funkcją rekurencyjną $n+2$ argumentów, to $h(y,x_1,\ldots,x_n)$ (funkcja $n+1$ argumentów) jest funkcją rekurencyjną określoną jako:
   $h(0,x_1,\ldots,x_n) = f(x_1,\ldots,x_n)$
   $h(y+1,x_1,\ldots,x_n) = g(y, h(y,x_1,\ldots,x_n),x_1,\ldots,x_n)$ — rekursja prosta
4. Jeśli f jest funkcją rekurencyjną $n+1$ zmiennych to funkcja $h(x_1,\ldots,x_n)$ będąca funkcją n zmiennych jest funkcją rekurencyjną określoną jako:
   $h(x_1,\ldots,x_n)= \mu_y(f(y,x_1,\ldots,x_n))$
   gdzie $\mu_y(f(y,x_1,\ldots,x_n))$ oznacza najmniejszą liczbę y spełniającą równanie:
   $f(y,x_1,\ldots,x_n)=0$ dla danych $x_1,\ldots,x_n$ — minimum efektywne

Funkcje rekurencyjne

Funkcje budowane przy pomocy operacji 1,2,3 i 4 noszą nazwę funkcji rekurencyjnych, zaś funkcje tworzone przy pomocy operacji 1, 2 i 3 nazywają się funkcjami pierwotnie rekurencyjnymi

$\mathcal{F}_{PR}$ — klasa funkcji pierwotnie rekurencyjnych

$\mathcal{F}_{R}$ — klasa funkcji rekurencyjnych

$\mathcal{F}_{PR} \subset \mathcal{F}_{R}$ $\quad$ $\mathcal{F}_{PR} \neq \mathcal{F}_{R}$

Funkcje rekurencyjne

Przy rozpatrywaniu obliczalności funkcji pierwotnie rekurencyjnych możemy oszacować liczbę taktów potrzebnych maszynie Turinga do obliczenia takiej funkcji, czyli określić złożoność czasową algorytmu realizowanego przez maszynę Turinga.

Dla funkcji rekurencyjnych tworzonych przy pomocy operacji 4 (minimum efektywne) nie da się w przypadku ogólnym przeprowadzić takiego oszacowania. Jednakże dowodzi się, że maszyna Turinga w skończonej liczbie kroków jest w stanie funkcje te obliczyć (pod warunkiem, że są one określone dla wszystkich argumentów swojej dziedziny).

Funkcje rekurencyjne

Przykłady:

a) $D(y,x)=y+x$ jest funkcją rekurencyjną, gdyż można ją otrzymać w drodze podstawienia i rekursji prostej funkcji podstawowych:
$D(0,x) = I_{1,1}(x) = x$
$D(y+1,x) = S(I_{2,3}(y,D(y,x),x)) = S(D(y,x)) = y+x+1$

b) $H(x)=2x$ jest funkcją rekurencyjną, gdyż można ją otrzymać w drodze podstawienia funkcji rekurencyjnych do funkcji $D(y,x)$, o której wiemy z punktu a), że jest rekurencyjna:
$H(x) = D(I_{1,1}(x), I_{1,1}(x)) = D(x,x) = x+x = 2x$

c) $M(y,x)=yx$ jest rekurencyjna, gdyż:
$M(0,x)=Z(x)=0$
$M(y+1,x) = I_{2,2}(y,D(M(y,x),x)) = D(M(y,x),x) = yx+x = (y+1)x$

d) $E(y,x)=x^y$ jest rekurencyjna, gdyż wykorzystując c) otrzymujemy:
$E(0,x) = S(Z(x)) = S(0) = 1 = x^0$
$E(y+1,x) = I_{2,2}(y,M(x,E(y,x)) = M(x,E(y,x)) = xx^y = x^{y+1}$

Obliczalność a akceptacja języka

Zbiór $B \subseteq N$ nazywamy przeliczalnie rekurencyjnym, gdy jego funkcja charakterystyczna $f(x)$:

$$f(x)=\begin{cases} 0, \text{ dla } x \notin B \\ 1, \text{ dla } x \in B \end{cases}$$

jest funkcją rekurencyjną.

Maszyna Turinga jest wtedy w stanie w skończonej liczbie kroków stwierdzić, czy $x \in B$, czy też $x \notin B$, czyli potrafi obliczyć funkcję charakterystyczną dla tego x.

Zbiór $B \subseteq N$ nazywamy przeliczalnie rekurencyjnym, jeżeli $B=\varnothing$ (B jest pusty) lub istnieje taka funkcja rekurencyjna $f(x,y)$, taka że:

$$(\forall x \in B)\ (\exists y \in N)\ (f(x,y)=0)$$

Klasa zbiorów rekurencyjnych Z_R jest podklasą właściwą klasy zbiorów rekurencyjnie przeliczalnych Z_{RP}

$$Z_R \subset Z_{RP} \text{ ale } Z_R \neq Z_{RP}$$

Obliczalność a akceptacja języka

Jeżeli $B \subset N$ jest zbiorem rekurencyjnie przeliczalnym, to maszyna Turinga jest w stanie w skończonej liczbie kroków określić, czy $x \in B$ tylko wtedy, gdy x rzeczywiście należy do B.

Gdy natomiast $x \notin B$ to $\neg(\exists y \in N)\ (f(x,y)=0)$, ale aby to sprawdzić trzeba przebadać wszystkie liczby naturalne, a tych jest nieskończenie wiele, więc badania nie da się przeprowadzić w skończonej liczbie kroków.

Pojęcia zbiorów rekurencyjnych i rekurencyjnie przeliczalnych odnosiły się do zbiorów liczb naturalnych. Można je wszakże przenieść na grunt języków.

Obliczalność a akceptacja języka

Numeracja Gödla: Można ponumerować słowa języka:

1. numerujemy elementy alfabetu
   $T=\{a_1,a_2,\ldots,a_n\}$
2. niech $p_1p_2p_3p_4\ldots$ będzie ciągiem rosnącym liczb pierwszych, np. $2,3,5,7,11,13,\ldots$
3. określamy funkcję $num(x)$ dla $x \in \Sigma^*$

$$num(\varepsilon) = 0$$

$$num(a_{i_1}a_{i_2}\ldots a_{i_k}) = \prod_{j=1}^{k} p_j^{\,i_j}$$

Można pokazać, że odwzorowanie

$$num: \Sigma^* \mapsto N$$

jest wzajemnie jednoznaczne (funkcja $num(x)$ jest różnowartościowa).

Obliczalność a akceptacja języka

Przykład:

1. $\Sigma = \{a,b\}$, numerujemy litery => $\Sigma = \{a_1,a_2\}$
2. określamy rosnący ciąg liczb pierwszych: $p_1, p_2, p_3, \ldots$ jako $2, 3, 5, 7, \ldots$
3. analizujemy słowo $x = abaa \in \Sigma^*$, $x = a_1a_2a_1a_1$, $num(x) = p_1^1\, p_2^2\, p_3^1\, p_4^1 = 2^1*3^2*5^1*7^1 = 2*9*5*7 = 630$

Obliczalność a akceptacja języka

Niech $L \subseteq \Sigma^*$ będzie językiem.

Zbiór *num(L)* określony jako

$$num(L) = \{n \in N \mid n = num(x) \wedge x \in L\}$$

jest zbiorem numerów słów tego języka.

Język L nazywamy rekurencyjnym, gdy jego zbiór *num(L)* jest zbiorem rekurencyjnym.

Język L nazywamy rekurencyjnie przeliczalnym, gdy jego zbiór *num(L)* jest zbiorem rekurencyjnie przeliczalnym.

Obliczalność a akceptacja języka

Akceptowalność języka L przez maszynę Turinga

$A = <Q, q_0, F, \Gamma, \Sigma, \delta> \in \mathcal{A}_T$

Q —zbiór stanów (q_0–stan początkowy, F–zbiór stanów końcowych)

Γ— alfabet taśmy

$\Sigma \subset \Gamma$—alfabet wejściowy

$b \in \Sigma - \Gamma$—symbol pusty (blank)

δ — funkcja przejścia

Maszyna Turinga A akceptuje język

$$L(A) = \{x \in \Sigma^* \mid (\exists q \in F)\ (\exists y \in \Gamma^*)\ ((q_0, \uparrow x) \succ^*_A (q, y \uparrow))\}$$

gdzie: $(q, y \uparrow)$ – konfiguracja stopująca

Obliczalność a akceptacja języka

Stwierdzenia dotyczące zbiorów rekurencyjnych i rekurencyjnie przeliczalnych przenoszą się na akceptowalność języków rekurencyjnych i rekurencyjnie przeliczalnych przez maszynę Turinga.

$\mathcal{L}_R$ — klasa języków rekurencyjnych

$\mathcal{L}_{RP}$ — klasa języków rekurencyjnie przeliczalnych

$\mathcal{L}_{TUR}$ — klasa języków akceptowanych przez maszynę Turinga

Obliczalność a akceptacja języka

Jeżeli $L \in \mathcal{L}_R$ to maszyna Turinga potrafi stwierdzić czy $x \in L$, czy też $x \notin L$ w skończonej liczbie kroków.

Jeżeli $L \in \mathcal{L}_{RP}$ to maszyna Turinga potrafi stwierdzić, że $x \in L$ tylko wtedy, gdy x rzeczywiście należy do L, w przeciwnym razie w przypadku ogólnym nie zatrzyma się po wykonaniu skończonej liczby kroków.

$$\mathcal{L}_R \subset \mathcal{L}_{TUR} \quad \text{ale} \quad \mathcal{L}_R \neq \mathcal{L}_{TUR}$$

$$\mathcal{L}_{RP} = \mathcal{L}_{TUR} = \mathcal{L}_{KOMB}$$

gdzie: $\mathcal{L}_{KOMB}$—klasa języków kombinatorycznych (klasa 0 (zero) w klasyfikacji Chomsky'ego)

Obliczalność a akceptacja języka

Tw: Klasa zbiorów rekurencyjnych $\mathcal{Z}_R$ jest zamknięta ze względu na operacje sumy, przecięcia (iloczynu mnogościowego) oraz uzupełnienia do N.
Jeżeli $A \in \mathcal{Z}_R$ to $(N - A) \in \mathcal{Z}_R$

Tw. Klasa zbiorów rekurencyjnie przeliczalnych $\mathcal{Z}_{RP}$ jest zamknięta ze względu na operacje sumy, przecięcia (iloczynu mnogościowego), nie jest natomiast zamknięta ze względu na uzupełnienie do N.

Obliczalność a akceptacja języka

Jeżeli $B \in \mathcal{Z}_{RP}$ to o zbiorze $(N - B)$ nic nie można powiedzieć, w szczególności nie można powiedzieć, że $(N - B)$ jest rekurencyjnie przeliczalny.

Gdyby $(N - B) \in \mathcal{Z}_{RP}$ to korzystając z faktu, że jeżeli $x \notin B$ to $x \in (N - B)$ maszyna Turinga potrafiłaby w skończonej liczbie kroków stwierdzić, że $x \in (N - B)$, a zatem mogłaby efektywnie określać, że $x \notin B$. To niestety nie ma miejsca.