Maszyny Turinga

Jerzy Pogonowski

Zakład Logiki Stosowanej UAM
www.logic.amu.edu.pl
pogon@amu.edu.pl

Funkcje rekurencyjne

Plan na dziś:

- Definicja maszyny Turinga;
- Przykłady obliczeń.

Wreszcie podamy pierwszą z matematycznych reprezentacji pojęcia obliczalności. Ograniczymy się przy tym do podstawowych definicji oraz kilku przykładów obliczeń.

Należy pamiętać, że ten kurs ma charakter wyłącznie propedeutyczny — tak był dotąd traktowany, a wedle nieoficjalnych jeszcze projektów nowego programu studiów Językoznawstwa i Nauk o Informacji, ma być całkiem zniesiony. Wykładu na temat maszyn Turinga szukać trzeba wtedy będzie w innych miejscach. Może istotnie studentom JiNoI zbyteczna jest jakakolwiek wiedza o matematycznych podstawach informatyki?

Będziemy korzystać z definicji oraz przykładów zamieszczonych w:

- I.A. Ławrow, L.L. Maksimowa *Zadania z teorii mnogości, logiki matematycznej i teorii algorytmów*. Wydawnictwo Naukowe PWN, Warszawa 2004. Z języka rosyjskiego przełożył Jerzy Pogonowski.

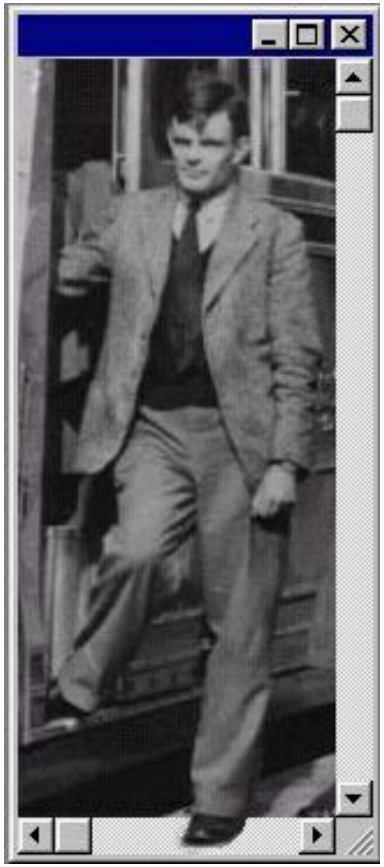
Dalsza, przykładowa, literatura:

- Hopcroft, J.E., Ullman, J.D. 1994. *Wprowadzenie do teorii automatów, języków i obliczeń.* Wydawnictwo Naukowe PWN, Warszawa.
- Kościelski, A. 1997. *Teoria obliczeń. Wykłady z matematycznych podstaw informatyki.* Wydawnictwo Uniwersytetu Wrocławskiego, Wrocław.
- Moczurad, M. 2002. *Wybrane zagadnienia z teorii rekursji.* Wydawnictwo Uniwersytetu Jagiellońskiego, Kraków.
- Murawski, R. 2000[3]. *Funkcje rekurencyjne i elementy metamatematyki. Problemy zupełności, rozstrzygalności, twierdzenia Gödla.* Wydawnictwo Naukowe UAM, Poznań.

Alan Turing

Pomnik w Manchester


140



Uwaga. Definicje, które zostaną za chwilę podane, „przystosowane" są do maszyn Turinga, które liczą wartości pewnych funkcji. Przy tym, korzystamy z faktu, że liczby naturalne (a także ich ciągi) mogą być kodowane poprzez ciągi binarne.

Można także patrzeć na maszyny Turinga jako na konstrukcje, które pozwalają rozpoznawać języki formalne, czyli ciągi słów utworzone ze znaków pewnego alfabetu.

Rozważane będą deterministyczne maszyny Turinga.
Definicje maszyn Turinga, spotykane w literaturze, mogą różnić się pewnymi drobnymi szczegółami (np. liczbą stanów końcowych).

Maszyna Turinga T określona jest całkowicie przez:

- (a) *alfabet zewnętrzny* $\mathcal{A} = \{a_0, a_1, \ldots, a_n\}$ (gdzie $a_0 = 0,\ a_1 = 1$);
- (b) *alfabet stanów wewnętrznych* $Q = \{q_0, q_1, \ldots, q_m\}$;
- (c) *program*, tj. zbiór wyrażeń $T(i,j)$ $(i = 1, \ldots, m;\ j = 0, \ldots, n)$, z których każde może mieć jedną z następujących postaci:
  - $q_i a_j \to q_k a_l$,
  - $q_i a_j \to q_k a_l R$
  - $q_i a_j \to q_k a_l L$

  gdzie $0 \leqslant k \leqslant m,\ 0 \leqslant l \leqslant n$.

Stan q_0 jest wyróżniony — jest stanem końcowym.
Stan q_1 też jest wyróżniony — jest stanem początkowym.
Symbole a_0 i a_1 także są wyróżnione — a_0 jest symbolem tzw. pustej klatki taśmy. Symbole a_0 i a_1 wystarczają do zapisania każdej informacji.
Wyrażenia $T(i,j)$ nazywają się *rozkazami*.

Słowem maszynowym lub *konfiguracją* nazywamy słowo postaci:

$$Aq_k a_l B$$

gdzie $0 \leqslant k \leqslant m$, $0 \leqslant l \leqslant n$, A oraz B — są słowami (być może pustymi) w alfabecie $\mathcal{A}$.

Piszemy a_i^x jako skrót wyrażenia $\underbrace{a_i a_i \ldots a_i}_{x \text{ razy}}$.

Proszę zauważyć, że maszyna Turinga jest pewnym obiektem matematycznym. Wszelkie określenia w rodzaju: „praca maszyny", „maszyna zatrzymuje się", itd. także mają ściśle zdefiniowany sens matematyczny.

Niech dane będą maszyna T oraz słowo maszynowe $M = Aq_i a_j B$, gdzie $0 \leqslant i \leqslant m$. Przez M'_T oznaczymy słowo otrzymane z M według następujących reguł:

- (1) dla $i = 0$ niech $M'_T = M$;
- (2) dla $i > 0$:
  - (a) jeśli $T(i,j)$ jest postaci $q_i a_j \to q_k a_l$, to $M'_T = Aq_k a_l B$;
  - (b) jeśli $T(i,j)$ jest postaci $q_i a_j \to q_k a_l R$, to:
    - (B_1) jeśli B nie jest słowem pustym, to $M'_T = Aa_l q_k B$,
    - (B_2) jeśli B jest słowem pustym, to $M'_T = Aa_l q_k a_0$;
  - (c) jeśli $T(i,j)$ jest postaci $q_i a_j \to q_k a_l L$, to:
    - (C_1) jeśli $A = A_1 a_s$ dla pewnych A_1 oraz a_s, to $M'_T = A_1 q_k a_s a_l B$,
    - (C_2) jeśli A jest słowem pustym, to $M'_T = q_k a_0 a_l B$.

Przyjmijmy $M_T^{(0)} = M,\ M_T^{(n+1)} = (M_T^{(n)})'$.

- Mówimy, że maszyna T *przetwarza* słowo maszynowe M w słowo M_1, jeżeli dla pewnego n: $M_T^{(n)} = M_1$. Piszemy wtedy $M \underset{T}{\Rightarrow} M_1$.
- Piszemy $M \underset{T}{\Rrightarrow} M_1$, jeśli maszyna T przetwarza M w M_1 i nie jest przy tym wykorzystywany warunek (C_2) powyższej definicji.
- Piszemy natomiast $M \underset{T}{\mapsto} M_1$, jeśli maszyna T przetwarza M w M_1, a przy tym nie są wykorzystywane warunki (B_1) oraz (C_2) powyższej definicji.

Zobaczymy za chwilę działanie bardzo prostej maszyny Turinga, na przykładzie programu zawartego w książce:

Wychodzimy z tej prezentacji i oglądamy działanie programu.

Mówimy, że maszyna T *oblicza* n-argumentową częściową funkcję liczbową f, gdzie $\delta_f \subseteq \mathcal{N}^n$, $\rho_f \subseteq \mathcal{N}$, jeśli spełnione są następujące warunki:

- (a) jeśli $\langle x_1, x_2, \ldots, x_n \rangle \in \delta_f$, to maszyna T *zatrzymuje się*, tj. przetwarza słowo $q_1 01^{x_1} 01^{x_2} 0 \ldots 01^{x_n} 0$ w pewne słowo Aq_0B, a przy tym słowo Aq_0B zawiera $f(x_1, x_2, \ldots, x_n)$ wystąpień symbolu 1;
- (b) jeśli $\langle x_1, x_2, \ldots, x_n \rangle \notin \delta_f$, to maszyna rozpoczynając działanie od słowa $M = q_1 01^{x_1} 01^{x_2} 0 \ldots 01^{x_n} 0$ *pracuje w nieskończoność*, tj. q_0 nie występuje w $M_T^{(n)}$ dla żadnego n.

δ_f oraz ρ_f oznaczają dziedzinę oraz przeciwdziedzinę f, odpowiednio.

Mówimy, że maszyna T *prawidłowo oblicza* funkcję f, jeśli spełnione są warunki:

- (a) jeśli $\langle x_1, x_2, \ldots, x_n \rangle \in \delta_f$, to

$$q_1 01^{x_1} 01^{x_2} 0 \ldots 01^{x_n} 0 \underset{T}{\Rightarrow} q_0 01^{f(x_1,x_2,\ldots,x_n)} 00 \ldots 0;$$

- (b) jeśli $\langle x_1, x_2, \ldots, x_n \rangle \notin \delta_f$, to maszyna, rozpoczynając działanie od słowa $q_1 01^{x_1} 01^{x_2} 0 \ldots 01^{x_n} 0$ pracuje w nieskończoność.

Funkcję f nazywamy *obliczalną* (*prawidłowo obliczalną*), jeśli istnieje maszyna, która oblicza (prawidłowo oblicza) funkcję f.

Dwa przykłady operacji na maszynach Turinga.

Niech T_1, T_2, T_3 będą maszynami Turinga z tym samym alfabetem zewnętrznym $\mathcal{A} = \{a_0, a_1, \ldots, a_n\}$, z alfabetami stanów wewnętrznych:

$$Q_1 = \{q_0, q_1, \ldots, q_r\}, \quad Q_2 = \{q_0, q_1, \ldots, q_s\}, \quad Q_3 = \{q_0, q_1, \ldots, q_t\}$$

oraz programami Π_1, Π_2, Π_3, odpowiednio.

Złożeniem $T_1 \cdot T_2$ maszyn T_1 i T_2 nazywamy maszynę T, której program jest sumą zbiorów:

$$S^{q_0}_{q_{r+1}}\Pi_1 \cup S^{q_1 \ldots q_s}_{q_{r+1} \ldots q_{r+s}}\Pi_2,$$

gdzie $S^{q_j}_{q_i}\Pi$ oznacza zbiór rozkazów otrzymanych z Π poprzez zamianę wszystkich wystąpień q_j na q_i.

Rozwidleniem maszyn T_1, T_2, T_3 względem (q_i, q_j),
(symbolicznie $T_1 \left\{ \begin{array}{l} q_i = T_2 \\ q_j = T_3 \end{array} \right.$), gdzie $q_i, q_j \in Q_1$,

nazywamy maszynę T, której program otrzymujemy w sposób następujący:

- z Π_1 usuwamy rozkazy $T_1(i,k)$ oraz $T_1(j,k)$ dla $k = 0, 1, \ldots, n$, otrzymany w ten sposób zbiór oznaczamy przez Π_1';
- wtedy

$$\Pi = \Pi_1' \cup S^{q_1 q_2 \ldots q_s}_{q_i q_{r+1} \ldots q_{r+s+1}} \Pi_2 \cup S^{q_1 q_2 \ldots q_t}_{q_j q_{r+s} \ldots q_{r+s+t-2}} \Pi_3.$$

Złożenia i rozwidlenia maszyn Turinga to zatem pewne operacje na tych maszynach. Przy obliczaniu różnych funkcji czasem wygodnie jest pracować z maszynami Turinga będącymi wynikami operacji na innych, prostszych maszynach Turinga. Dla przykładu, funkcje definiowane przez schemat rekursji prostej z funkcji prawidłowo obliczalnych w sensie Turinga określać można przez złożenie i rozwidlenie stosownych maszyn Turinga.

Pokażemy teraz, że maszyny Turinga mogą być kodowane przez liczby naturalne. Niech p_n oznacza n-tą liczbę pierwszą.

Niech $A = a_{s_0} \dots a_{s_k}$ będzie słowem w alfabecie $\{a_0, a_1, a_2 \dots\}$.
Przyjmijmy:

$$k_l(A) = \prod_{t=0}^{k} p_t^{s_{k-t}}, \; k_r(A) = \prod_{t=0}^{k} p_t^{s_t}.$$

Jeśli $M = Aq_i a_j B$ jest słowem maszynowym, to przyjmijmy:

$$\nu(M) = 2^{k_l(A)} \cdot 3^i \cdot 5^j \cdot 7^{k_r(B)}.$$

Numerem rozkazu $T(i,j)$ nazwiemy liczbę:

$$\mu(T(i,j)) = p_{c(i,j)}^{p_0^k \cdot p_1^l \cdot p_2^s},$$

gdzie $c(x,y) = \frac{(x+y)^2+3x+y}{2}$ jest *funkcją numerującą Cantora* oraz:

$$\begin{array}{lll} s=0, & \text{jeśli } T(i,j) \text{ jest postaci} & q_i a_j \longrightarrow q_k a_l, \\ s=1, & \text{jeśli } T(i,j) \text{ jest postaci} & q_i a_j \longrightarrow q_k a_l L, \\ s=2, & \text{jeśli } T(i,j) \text{ jest postaci} & q_i a_j \longrightarrow q_k a_l R. \end{array}$$

Numerem $\lambda(T)$ *maszyny* T nazwiemy iloczyn wszystkich numerów rozkazów $T(i,j)$ maszyny T.

Tak więc, maszyny oraz ich programy (a także ich dane) możemy traktować tak, jak liczby naturalne! Zobaczymy później, jak ważne ma to konsekwencje.

1. Jaką funkcję $f(x)$ oblicza maszyna T o następującym programie:

- $q_1 0 \longrightarrow q_2 0 R$,
- $q_1 1 \longrightarrow q_0 1$,
- $q_2 0 \longrightarrow q_0 1$,
- $q_2 1 \longrightarrow q_2 1 R$?

Odpowiedź. $f(x) = x + 1$.
Aby tę odpowiedź uzasadnić, należy udowodnić, że maszyna Turinga T przetwarza słowo $q_1 01^x 0$ w słowo $Aq_0 B$ takie, że w $Aq_0 B$ występuje $x + 1$ razy symbol 1.

Rozważmy przykładowe obliczenie. Weźmy słowo 1^3.
Do tablicy idzie ochotnik. Śmiało.

2. Skonstruować maszynę Turinga T, która prawidłowo oblicza funkcję $o(x) = 0$.

Odpowiedź. Na przykład:

$$\begin{aligned} &q_1 0 \longrightarrow q_2 0 R, \\ &q_2 0 \longrightarrow q_3 0 L, \quad && q_2 1 \longrightarrow q_2 1 R, \\ &q_3 0 \longrightarrow q_0 0, && q_3 1 \longrightarrow q_3 0 L. \end{aligned}$$

Aby tę odpowiedź uzasadnić, należy udowodnić, że dla dowolnego słowa A mamy:

$$q_1 0 A 0 \underset{T}{\Rightarrow} q_0 000 \ldots 0.$$

Rozważmy przykładowe obliczenie. Weźmy słowo 111.
Do tablicy idzie ochotnik.

3. Skonstruować maszynę Turinga T prawidłowo obliczającą funkcję $f(x, y) = x + y$.

Odpowiedź. Na przykład:

$$\begin{array}{ll} q_1 0 \to q_2 0 R, & \\ q_2 0 \to q_3 1 R, & q_2 1 \to q_2 1 R, \\ q_3 0 \to q_4 0 L, & q_3 1 \to q_3 1 R, \\ & q_4 1 \to q_5 0 L, \\ q_5 0 \to q_0 0, & q_5 1 \to q_5 1 L. \end{array}$$

Aby tę odpowiedź uzasadnić, należy udowodnić, że dla wszystkich x oraz y:

$$q_1 0 1^x 0 1^y 0 \underset{T}{\Rightarrow} q_0 0 1^{x+y} 0 0 \ldots 0.$$

Rozważmy przykładowe obliczenie. Obliczmy $2 + 3$.
Do tablicy idzie ochotnik.

4. Skonstruować maszynę Turinga T obliczającą funkcję $f(x) = x \dot{-} 1$, gdzie $x \dot{-} 1 = x - 1$ dla $x > 0$ oraz $0 \dot{-} 1 = 0$.

Odpowiedź. Na przykład:

$$\begin{array}{ll} q_1 0 \to q_2 0 R, & \\ q_2 0 \to q_0 0 L, & q_2 1 \to q_3 1 R, \\ q_3 0 \to q_4 0 L, & q_3 1 \to q_3 1 R, \\ & q_4 1 \to q_5 0 L, \\ q_5 0 \to q_0 0, & q_5 1 \to q_5 1 L. \end{array}$$

Aby tę odpowiedź uzasadnić, należy udowodnić, że dla każdego $x > 0$:

$$q_1 0 1^x 0 \underset{T}{\Rightarrow} A q_0 B$$

dla pewnych A, B takich, że symbol 1 występuje $x - 1$ razy w $A q_0 B$ oraz że $q_1 000 \underset{T}{\Rightarrow} C q_0 D$, dla pewnych C, D nie zawierających symbolu 1.

Rozważmy przykładowe obliczenie. Obliczmy $f(2)$.

Do tablicy idzie ochotnik.

5. Skonstruować maszynę Turinga T prawidłowo obliczającą funkcję $sg(x)$; gdzie $sg(x) = 1$ dla $x > 0$ oraz $sg(0) = 0$.

Odpowiedź. Na przykład:

$$\begin{aligned} &q_1 0 \to q_2 0 R, \\ &q_2 0 \to q_0 0 L, \quad q_2 1 \to q_3 1 R, \\ &q_3 0 \to q_4 0 L, \quad q_3 1 \to q_3 0 R, \\ &q_4 0 \to q_4 0 L, \quad q_4 1 \to q_0 0 L. \end{aligned}$$

Ćwiczenie. Co należy udowodnić, aby uzasadnić tę odpowiedź? Wykonaj przykładowe obliczenie.

Jak „mocne” są maszyny Turinga?
Języki przez nie rozpoznawane to języki rekurencyjnie przeliczalne.

Jest jasne, że nie każdy język może być rozpoznany przez maszynę Turinga: wszystkich języków nad ustalonym skończonym alfabetem jest kontinuum, a maszyn Turinga z tym alfabetem jest tylko przeliczalnie wiele.

Problem stopu. Problem, czy dowolna maszyna Turinga dla dowolnych danych zakończy obliczenie, jest problemem nierozstrzygalnym.

Maszyny Turinga są dość trudnym narzędziem w praktycznych obliczeniach.
Są jednak przydatne w rozważaniach metateoretycznych.
A bez takich rozważań, jak wiadomo, byłabyś dzieckiem we mgle.
Bo skąd możesz wiedzieć, że coś robisz poprawnie, jeśli nie wiesz, co właściwie robisz?

Busy Beaver. Jest skończenie wiele zatrzymujących się maszyn Turinga o n stanach nad alfabetem binarnym. Funkcja przyporządkowująca liczbie n największą liczbę symboli 1, która pozostaje na taśmie po zakończeniu pracy maszyny Turinga o n stanach i zaczynającej pracę z taśmą, na której są tylko symbole 0, jest dobrze określona. **Nie** jest ona jednak obliczalna przez żadną maszynę Turinga.

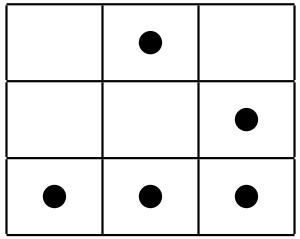
Gdybyśmy mieli jakiś systematyczny sposób na otrzymywanie wartości tej funkcji, to można byłoby rozstrzygać prawie każdy problem matematyczny.

Busy Beaver dla $n = 6$ wyprodukował 10^{865} symboli 1, zatrzymując się po 10^{1730} krokach.

Teza Churcha-Turinga. Każda funkcja, która jest (w intuicyjnym sensie) obliczalna, jest obliczalna przez pewną maszynę Turinga. To oczywiście **nie** jest twierdzenie matematyczne. Także twierdzenie do niego odwrotne, głoszące, iż każda funkcja obliczalna przez pewną maszynę Turinga jest (w intuicyjnym sensie) obliczalna, **nie** jest twierdzeniem matematycznym. To, że te dwie klasy funkcji są identyczne (a także pokrywają się z klasami wyznaczonymi przez inne matematyczne reprezentacje obliczalności) nazywa się Tezą Churcha-Turinga.

Test Turinga. To procedura zaproponowana przez Turinga. Masz dwóch (niewidocznych) interlokutorów: człowieka oraz komputer. Jeśli nie potrafisz wskazać, który z nich jest komputerem, to należy uznać, iż maszyna przeszła test na bycie podmiotem inteligentnym.

Gra Conwaya. Popularna „gra w życie" ma taką samą moc obliczeniową, jak uniwersalna maszyna Turinga. A oto glider, jeden z bohaterów tej gry (a także symbol subkultury hakerów):



Chinese Room. To problem zaproponowany przez Johna Searle'a. Nie znasz chińskiego. Masz do dyspozycji kompletne reguły przekładu z chińskiego na twój język oraz z twojego języka na chiński. Zamknięto cię w pokoju, do którego dostarczane są pytania po chińsku. Posługując się tylko wspomnianymi regułami, udzielasz odpowiedzi. Czy oznacza to, że rozumiesz język chiński?
Ten problem wiąże się zatem z pytaniem o to, jaka część naszej wiedzy ma charakter algorytmiczny.

Koniec

Od następnego wykładu będziemy pracować z funkcjami rekurencyjnymi.

Na jednym z dalszych wykładów wspomnimy (czysto informacyjnie) o innych jeszcze matematycznych reprezentacjach pojęcia obliczalności:

- algorytmach Markowa;
- obliczalności według Herbranda – Gödla;
- rachunku lambda;
- numeracjach Kleene'go i Posta.

Koniec

I tym wesołym akcentem możemy dzisiejsze zajęcia zakończyć.