Maszyny Turinga i problemy nierozstrzygalne

Maszyna Turinga jest automatem taśmowym, składa się z taśmy (tablicy symboli) potencjalnie nieskończonej w prawo, zakładamy,że w prawie wszystkich (tzn. wszystkich poza skończoną liczbą) klatkach taśmy jest domyślny symbol B. Formalnie maszyna jest zadana przez sekwencję:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

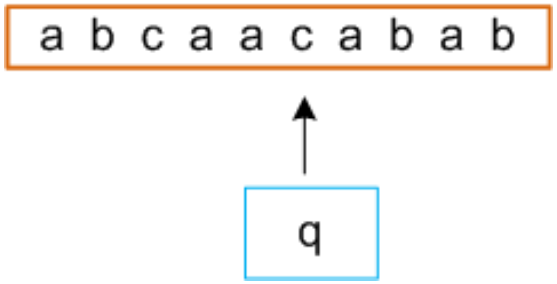
gdzie:

- Q, Σ, Γ to odpowiednio: zbiór stanów, symboli wejściowych i symboli taśmowych;
- $\delta : Q \times \Gamma \Rightarrow Q \times \Gamma - \{B\} \times \{-1, 0, +1\}$;
- $q_0 \in Q$, $F \subseteq Q$;
- B to specjalny symbol domyślny, $B \notin \Sigma$.

Konfiguracja maszyny składa się ze słowa na taśmie (pomijając symbole B), stanu oraz pozycji głowicy czytającej. Jeśli słowem na taśmie jest $a_1 a_2 \dots a_m$ a głowica czyta symbol k-ty, gdzie $k \leq m+1$, to zapisujemy konfigurację jako

$$a_1 a_2 a_3 \dots a_{k-1}\, q\, a_k a_{k+1} \dots a_m.$$

Zakładamy, ze symbolem $(m+1)$-szym jest automatycznie B.



Przykładowa konfiguracja maszyny.
Liniowy zapis konfiguracji to $abcaa\,q\,cabab$.

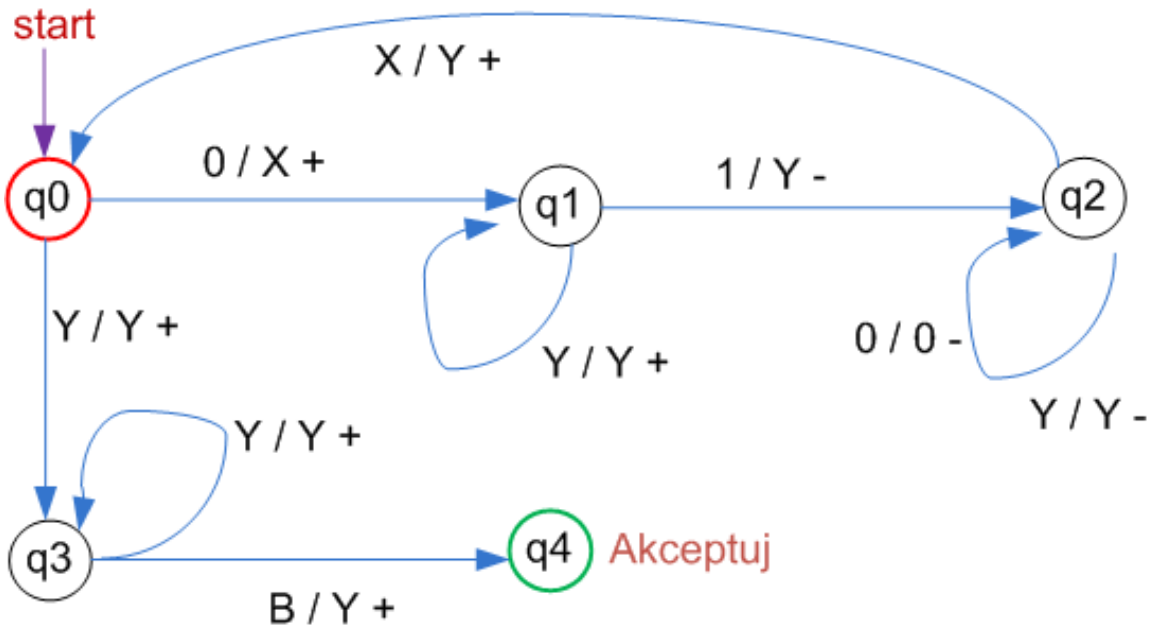
Następująca maszyna akceptuje język

$$L = \{ a^n b^n : n \geq 1 \}$$

przejście $p \xrightarrow{Z \,/\, Z' \, K} q$ oznacza, że

$$\delta(p, Z) = (q, Z', K),$$

gdzie K oznacza ruch głowicy maszyny.



Oznaczmy przez $\vdash$ relację przejścia w jednym kroku od jednej konfiguracji do następnej. Konfiguracją początkową maszyny jest $K_0 = q_0 a_1 a_2 \ldots a_n$, gdzie $a_1 a_2 \ldots a_n$ jest słowem wejściowym w (nad alfabetem Σ).
Maszyna Turinga M akceptuje słowo gdy z konfiguracji początkowej możemy dojść do konfiguracji zawierającej stan akceptujący.
Językiem akceptowanym przez M jest

$$L(M) = \{ w : q_0 w \vdash^* \alpha q \beta, \text{ dla pewnych } \alpha, \beta \in \Gamma^*, q \in F \}$$

Dla naszej przykładowej maszyny:

$$q_0 0011 \vdash^* XXYYY q_4, \text{ oraz } q_4 \in F.$$

Zatem $0011 \in L(M)$.

Możliwe są różne wersje maszyny Turinga, równoważne co do mocy obliczeniowej (z dokładnością do funkcji wielomianowej)

- Maszyna Turinga z taśma nieskończona w obie strony;
- Maszyna Turinga wielotaśmowa;
- Maszyna z dwuwymiarową taśmą;
- Maszyna z jedną kolejką zamiast taśmy (automat kolejkowy);
- Maszyna z dwoma stosami zamiast taśmy;
- Maszyna z dwoma licznikami zamiast taśmy.

Twierdzenie (Teza Churcha-Turinga)

Dany problem decyzyjny, reprezentowany przez język L, jest częściowo rozstrzygalny (posiada algorytm, być może bez własności stopu) wtw. gdy istnieje maszyna Turinga akceptująca L.

Języki (problemy decyzyjne) częściowo rozstrzygalne to te dla których istnieje akceptująca maszyna Turinga, nazywamy je również przeliczalnie rekurencyjnymi.
Częściowa rozstrzygalność oznacza, że istnieje algorytm który poprawnie odpowiada "TAK", ale może sie zdarzyć, że odpowiedzią powinno być "NIE", natomiast algorytm tej odpowiedzi nigdy nie udzieli (nie zatrzyma się).

Języki dla których istnieje algorytm decyzyjny (maszyna Turinga) z własnościa stopu nazywamy rekurencyjnymi (rozstrzygalnymi).

Twierdzenie

Jeśli L jest rekurencyjny to $\bar{L}$ (dopełnienie L) też jest.

Jeśli L i $\bar{L}$ są przeliczalnie rekurencyjne to L i $\bar{L}$ są rekurencyjne.

Każdą maszynęTuringa możemy **konstruktywnie** zakodować jako ciąg zer i jedynek i przyporządkować maszynie Turinga numer, M_i jest i-tą maszyną Turinga (której kodem jest zapis binarny liczby i). Mogłoby się zdarzyć, że w normalnym kodowaniu nie ma maszyny odpiadającej kodowaniu i, wtedy przyjmujemy że M_i jest jednostanową maszyną która nic nie akceptuje. Słowa wejściowe też można zakodować efektywnie (np. leksykograficznie).
Niech w_i i-tym słowem wejściowym. Wszystko redukujemy do alfabetu binarnego.
Definiujemy język przekątniowy (d od ang. *diagonal*):

$$L_d = \{ w_i : w_i \notin L(M_i) \}$$

Twierdzenie

(1) *Język L_d nie jest przeliczalnie reukurencyjny.*
Nie istnieje takie j, że $L_d = L(M_j)$.
(2) *Dopełnienie języka L_d jest przeliczalnie rekurencyjne, ale jest nierozstrzygalne (nie rekurencyjne).*

Dowód

(1) Gdyby $L_d = L(M_j)$ dla pewnego j to

$$w_j \in L(M_j) \Leftrightarrow w_j \notin L(M_j).$$

Otrzymujemy sprzeczność, zatem takiego j nie ma. Tak więc nie istnieje maszyna Turinga akceptująca L_d.

(2) Uruchamiamy i-tą maszynę sprawdzającą czy $w_i \in L(M_i)$. Kodowanie maszyn jest konstruktywne, zatem mamy algorytm, który dla $w_i \in L(M_i)$ da odpowiedź "TAK" i się zatrzyma. Dopełnienie L_d nie jest rekurencyjne bo L_d nie jest przeliczalnie rekurencyjne.

Twierdzenie

Następujące problemy są nierozstrzygalne (nierekurencyjne), natomiast są częściowo rozstrzygalne (przeliczalnie rekurencyjne)

- *Problem stopu maszyny Turinga*
- *Problem przynależności $w \in L(M)$ (dla danego słowa w i maszyny Turinga M).*

Dowód

Gdybyśmy umieli sprawdzić pełnym algorytmem (z własnością stopu) któryś z powyższych problemów to mielibyśmy algorytm dla dopełnienia L_d, tzn. umielibyśmy sprawdzić czy $w_i \in L(M_i)$ za pomocą algorytmu, który się zawsze zatrzymuje. No ale ten ostatni problem jest nierozstrzygalny (chociaż jest przeliczalnie rekurencyjny).

Niech

$$Valid(M, w) \;=\; K_0 \# K_1^R \# K_2 \# K_3^R \# K_4 \# K_5^R \# \ldots K_m$$

będzie zbiorem składajacym się z jednego słowa odpowiadającego zmodyfikowanej historii działania maszyny M na słowie w, tzn. $K_0 = q_0\, w$, $K_i \vdash K_{i+1}$, oraz K_m zawiera stan akceptujący.
Jeśli takiego słowa nie ma, czyli $w \notin L(M)$, to definiujemy $Valid(M) = \emptyset$.
Niech $NonValid(M, w)$ będzie dopełnieniem $Valid(M, w)$.

Lemat

Dla danej maszyny Turinga i słowa w można skonstruować gramatyki G_1, G_2, G takie, ze

$$Valid(M) \;=\; L(G_1) \cap L(G_2), \quad L(G) \;=\; NonValid(M, w)$$

.

Twierdzenie

Następujące problemy są nierozstrzygalne dla gramatyk bezkontekstowych G_1, G_2, G, G'

1. $L(G_1) \cap L(G_2) \neq \emptyset$
2. $L(G) = \Sigma^*$
3. $L(G')$ *jest językiem regularnym*

Pierwsze dwa punkty są oczywistym wnioskiem z poprzedniego lematu. Dla dowodu (3) weźmy dowolny nieregularny język bezkontekstowy L_0. Wtedy bezkontekstowy język

$$L = \Sigma^* \# L_0 \cup L(G') \# \Sigma^*$$

jest regularny wtw. gdy $L(G') = \Sigma^*$.
Tak więc problem z punktu (2) sprowadziliśmy do problemu z punktu (3). Nierozstrzygalność (2) implikuje nierozstrzygalność (3).

Twierdzenie

Nie istnieje algorytm konstrukcji automatu skończonego dla języka $L(G)$, gdzie G jest gramatyką bezkontekstową generującą język regularny (wiemy że taki automat istnieje).

Uzasadnienie.

Gramatyka G z punktu (2) poprzedniego twierdzenia generuje język regularny. Gdybyśmy umieli skonstruować automat skończony A dla tego języka to moglibysmy sprawdzic łatwo czy $L(G) = \Sigma^*$ korzystając z algorytmu na równoważność deterministycznych automatów skończonych. Jednym z tych automatów jest A a drugim automat akceptujący Σ^*.

Problem $L(G) = \Sigma^*$ jest nierozstrzygalny, zatem nie ma algorytmu konstruującego automat A (pomimo tego, że wiemy że A istnieje).