7 Rozstrzygalność

Większość zagadnień informatyki związana jest z algorytmicznym rozwiązywaniem problemów. Istnieją jednak problemy, których nie da się rozwiązać niezależnie od ilości dostępnej pamięci oraz czasu dostępnego dla rozwiązującego je urządzenia. Przyczyną takiej sytuacji nie jest konieczność podania skomplikowanego wyniku lecz brak możliwości zautomatyzowania działań koniecznych do jego uzyskania. Dla ułatwienia analizy ograniczymy się więc do badania *problemów decyzyjnych*, które na podstawie danych wejściowych dają odpowiedź *TAK* lub *NIE*.

Intuicyjnie, rozstrzygalność problemu oznacza istnienie rozwiązującego go algorytmu. Jeśli algorytm o tej własności nie istnieje – problem jest nierozstrzygalny. Z formalnego punktu widzenia wygodnie jest reprezentować problemy obliczeniowe za pomocą języków nad ustalonym alfabetem Σ. Słowo $w \in \Sigma^*$ nazywamy instancją problemu. Rozstrzygnięcie problemu P polega wówczas na udzieleniu odpowiedzi (twierdzącej lub przeczącej) na pytanie o należenie do odpowiadającego mu języka L_P.

Formalnie:

Definicja 7.1

Język $L \subseteq \Sigma^*$ nazywamy rozstrzygalnym jeśli istnieje maszyna Turinga, która zatrzymuje się dla każdego słowa $w \in \Sigma^*$ odpowiednio w stanie akceptującym jeśli $w \in L$ oraz w stanie odrzucającym jeśli $w \notin L$.

Przykład 7.1

Problem P: „liczba $x \in \mathbb{N}$ jest parzysta" jest rozstrzygalny. Problem ten jest reprezentowany przez język:

$$L_P = \Big\{ w \in \{0,1\}^* : w \text{ jest binarnym zapisem parzystej liczby naturalnej} \Big\}.$$

Maszyna Turinga M_P rozstrzygająca język L_P sprawdza ostatnią (najmniej znaczącą) cyfrę zapisu binarnego w. Jeśli cyfra ta jest zerem M_P zwraca odpowiedź *TAK*, w przeciwnym przypadku zwraca odpowiedź *NIE*.

Zauważmy, że dla niektórych instancji $w \in \Sigma^*$ rozstrzygnięcie problemu „$w \in L_P$" może być bardzo proste. Nie oznacza to jednak automatycznie jego rozstrzygalności. Aby język był rozstrzygalny wymagamy istnienia maszyny Turinga dającej odpowiedź twierdzącą lub przeczącą dla dowolnej instancji $w \in \Sigma^*$. Nierozstrzygalność języka nie oznacza, że rozstrzygający go algorytm nie jest znany, lecz że taki algorytm nie istnieje i nie będzie możliwe jego stworzenie również w przyszłości.

Zaprezentujemy teraz przykłady problemów decyzyjnych, dla których nie istnieją algorytmy rozstrzygające.

Twierdzenie 7.1

Problem „ϕ_x jest totalna" jest nierozstrzygalny.

Dowód

Załóżmy, że problem „ϕ_x jest totalna" jest rozstrzygalny. Zatem obliczalna jest funkcja:

$$c(x) = \begin{cases} 1 & \phi_x \text{ jest totalna} \\ 0 & \phi_x \text{ nie jest totalna} \end{cases}$$

Wówczas obliczalna jest również funkcja symulująca działanie programów liczących funkcje totalne:

$$t(x,y) = \begin{cases} \Psi_U(x,y) & c(x) = 1 \\ 0 & c(x) = 0 \end{cases}$$

Rozważmy funkcję $g(x) = t(x,x) + 1$, która w oczywisty sposób jest totalna. Przypuśćmy, że i jest numerem programu obliczającego funkcję g, czyli $g(x) = t(i,x)$. Wówczas, z definicji t mamy $g(i) = t(i,i)$. Jednocześnie z określenia g mamy $g(i) = t(i,i) + 1$, co daje sprzeczność. ■

Twierdzenie 7.2

Problem „$x \in D_x$" jest nierozstrzygalny.

Dowód

Załóżmy, że problem „$x \in D_x$" jest rozstrzygalny. Wówczas funkcja

$$g(x) = \begin{cases} 1 & x \notin D_x \\ \infty & x \in D_x \end{cases}$$

jest obliczalna. Niech m będzie numerem dowolnego programu ją obliczającego ($g = \phi_m$). Wówczas:

$$m \in D_m \iff m \in D_g \iff m \notin D_m$$

co daje sprzeczność. ■

Zbiory rekurencyjne

Zbiory rekurencyjne są obiektami matematycznymi ściśle związanymi z pojęciem rozstrzygalności problemów obliczeniowych.

Definicja 7.2

Niech $A \subseteq \mathbb{N}^n$ będzie zbiorem, zaś $c_A : \mathbb{N}^n \longrightarrow \mathbb{N}$ jego funkcją charakterystyczną określoną:

$$c_A(\overline{x}) = \begin{cases} 1 & \overline{x} \in A \\ 0 & \overline{x} \notin A \end{cases}$$

Zbiór A nazywamy rekurencyjnym, jeśli jego funkcja charakterystyczna jest obliczalna ($c_A \in C_n$).

Przykład 7.2

1. Zbiór liczb parzystych (podzbiór $\mathbb{N}$) jest zbiorem rekurencyjnym.
2. Zbiór $\{x \in \mathbb{N} : \ x \in D_x\}$ nie jest zbiorem rekurencyjnym.

Zauważmy, że jeśli zbiór A jest rekurencyjny (jego funkcja charakterystyczna c_A jest obliczalna), problem „$\overline{x} \in A$" jest rozstrzygalny. Z drugiej strony, jeśli problem „$\overline{x} \in A$" jest nierozstrzygalny, funkcja c_A nie jest obliczalna, a

zatem zbiór A nie jest rekurencyjny. Procedurę obliczającą wartość funkcji c_A nazywa się często procedurą rozstrzygającą. Relację między rekurencyjnością zbiorów oraz rozstrzygalnością problemów możemy zapisać w skrócie:

$$A \text{ jest rekurencyjny} \iff c_A \in C_n \iff \text{„}\overline{x} \in A\text{” jest rozstrzygalny}$$

Podstawowe własności zbiorów rekurencyjnych opisuje poniższe twierdzenie.

Twierdzenie 7.3

Dla ustalonego $n \geqslant 1$ zbiór wszystkich podzbiorów rekurencyjnych $A \subseteq \mathbb{N}^n$ jest zamknięty na operacje teoriomnogościowe.

Dowód

W sposób oczywisty zbiory $\emptyset$ oraz $\mathbb{N}^n$ są rekurencyjne. Niech $A, B \subseteq \mathbb{N}^n$ będą rekurencyjne. Wówczas, obliczalne będą również funkcje charakterystyczne następujących zbiorów:

- dopełnienie zbioru A: $c_{\mathbb{N}^n \setminus A}(\overline{x}) = 1 - c_A(\overline{x})$,
- suma zbiorów A i B: $c_{A \cup B}(\overline{x}) = \max\Big(c_A(\overline{x}), c_B(\overline{x})\Big)$,
- przekrój zbiorów A i B: $c_{A \cap B}(\overline{x}) = c_A(\overline{x}) \cdot c_B(\overline{x})$.

■

Redukowalność problemów obliczeniowych

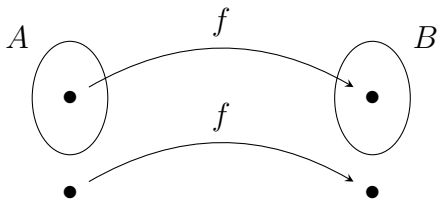
Redukcją nazywamy procedurę przekształcającą instancję jednego problemu obliczeniowego w instancję drugiego tak, by rozwiązanie drugiego problemu można było wykorzystać przy rozwiązaniu pierwszego. Jest to narzędzie bardzo często stosowane w dowodzeniu rozstrzygalności oraz nierozstrzygalności.

Definicja 7.3

Powiemy, że język (problem) A jest redukowalny do języka B jeśli istnieje totalna i obliczalna funkcja $f : \Sigma^* \longrightarrow \Sigma^*$ taka, że

$$\forall_{x \in \Sigma^*}\ x \in A \iff f(x) \in B.$$

Funkcję f nazywamy redukcją A do B.



Rysunek 7.1: Graficzna interpretacja redukcji problemów obliczeniowych

Przykład 7.3

Problem znalezienia drogi w nieznanym mieście można zredukować do problemu znalezienia mapy tego miasta.

Obserwacja

Zauważmy, że jeśli problem obliczeniowy A redukuje się do problemu B, to:

- Możliwe jest wykorzystanie rozwiązania problemu B w celu rozwiązania problemu A.
- Rozwiązanie problemu A nie może być trudniejsze niż rozwiązanie problemu B (ponieważ rozwiązanie B daje rozwiązanie A).

Powyższe rozważania sformułujemy w postaci twierdzenia.

Twierdzenie 7.4

Niech A i B będą językami, zaś funkcja f redukcją A do B. Jeśli język B jest rozstrzygalny, również język A jest rozstrzygalny.

Dowód

Niech M_B będzie maszyną Turinga rozstrzygającą język B. Maszyna M_A rozstrzygająca problem „$x \in A$” działa według następującego schematu:

- Oblicz wartość $f(x)$.
- Uruchom maszynę M_B na wejściu $f(x)$.
- Jako wynik zwróć wynik działania maszyny M_B.

■

Wniosek

Jeśli nierozstrzygalny problem A jest redukowalny do problemu B, to problem B również jest nierozstrzygalny.

Powyższy wniosek wykorzystywany jest bardzo często w dowodach nierozstrzygalności przebiegających według następującego schematu. Zakładamy rozstrzygalność problemu B i redukujemy do niego problem A, o którym wiemy, że jest nierozstrzygalny. Wówczas z rozstrzygalności problemu B musiałaby wynikać rozstrzygalność problemu A co prowadzi do sprzeczności.

Uwaga

Istnienie redukcji problemu A do problemu B nic nie mówi na temat rozstrzygalności żadnego z nich. Daje ona jedynie możliwość rozwiązania problemu A korzystając z rozwiązania problemu B oraz uzależnia rozstrzygalność A od rozstrzygalności B.

W czasie jednego z wcześniejszych wykładów pokazaliśmy dowód nierozstrzygalności problemu akceptowania słowa wejściowego przez maszynę Turinga. Pokażemy teraz za pomocą redukcji, że określenie czy maszyna Turinga zatrzyma się dla danych wejściowych (tzw. *problem stopu*) jest problemem nierozstrzygalnym.

Twierdzenie 7.5 (Problem stopu)

Problem stopu

$$\mathcal{P}_{STOP} = \Big\{(M, x) : \text{ maszyna } M \text{ zatrzymuje się na danych } x\Big\}$$

jest nierozstrzygalny.

Dowód

Załóżmy, że problem $\mathcal{P}_{STOP}$ jest rozstrzygalny przez maszynę Turniga M_S. Skonstruujemy maszynę M_A rozstrzygającą problem $\mathcal{P}_{ACC}$ akceptowania słowa wejściowego przez maszynę Turinga.

Maszyna M_A działa według schematu:

- Uruchom maszynę M_S na danych wejściowych (M, x)
- Jeśli M_S odrzuci dane wejściowe (M nie zatrzymuje się na wejściu x), to odrzuć.
- Jeśli M_S zaakceptuje dane wejściowe (M zatrzymuje się na wejściu x), symuluj działanie maszyny M do jej zatrzymania.

- Zwróć wynik (akceptacja/odrzucenie) zwrócony przez maszynę M.

Jeśli maszyna M_S rozstrzygałaby problem stopu, maszyna M_A rozstrzygałaby problem akceptowania wejścia. Wiemy jednak, że problem ten jest nierozstrzygalny. Uzyskana sprzeczność dowodzi, że również problem stopu jest nierozstrzygalny. ■

Za pomocą redukcji można również udowodnić nierozstrzygalność następujących problemów:

- Problem pustości języka akceptowanego przez maszynę Turinga.
- Problem, czy język rozpoznawany przez maszynę Turinga jest regularny (bezkontekstowy, skończony, pusty, rozstrzygalny).
- Problem równości języków akceptowanych przed dwie różne maszyny Turinga.

Ćwiczenie 7.1

Sformułuj formalne dowody (redukcje) dla powyższych problemów.

Twierdzenie Rice'a

Kolejnym ważnym narzędziem pozwalającym badać rozstrzygalność problemów obliczeniowych jest twierdzenie Rice'a. Mówi ono, że sprawdzenie dowolnej nietrywialnej własności funkcji obliczalnych jest nierozstrzygalne. Własność nazwiemy nietrywialną, jeśli istnieje co najmniej jedna funkcja obliczalna posiadająca tę własność oraz co najmniej jedna funkcja obliczalna nie posiadająca tej własności.

Twierdzenie 7.6 (Rice – wersja 1 – funkcje)

Niech $\mathcal{B}$ będzie właściwym i niepustym podzbiorem zbioru wszystkich funkcji obliczalnych. Wówczas problem „$\phi_x \in \mathcal{B}$" jest nierozstrzygalny. Równoważnie, zbiór $B = \{x \in I\!N : \ \phi_x \in \mathcal{B}\}$ nie jest rekurencyjny.

Możemy również sformułować równoważne twierdzenie mówiące, że wszystkie nietrywialne własności języków akceptowanych przez maszyny Turinga są nierozstrzygalne. Własność nazwiemy nietrywialną, jeśli istnieje co najmniej jeden język mający tę własność oraz co najmniej jeden język nie mający tej własności.

Twierdzenie 7.7 (Rice – wersja 2 – języki)

Niech $\mathcal{B}$ będzie właściwym i niepustym podzbiorem zbioru wszystkich języków rozpoznawalnych przez maszyny Turinga. Wówczas problem „$L(M) \in \mathcal{B}$" jest nierozstrzygalny (zbiór $B = \{M : \ L(M) \in \mathcal{B}\}$ nie jest rekurencyjny).

Dowód

Bez straty ogólności możemy założyć, że język pusty $\emptyset \notin \mathcal{B}$. W przeciwnym przypadku wykorzystamy obserwację, że zbiór jest rekurencyjny wtedy i tylko wtedy, gdy jego dopełnienie jest rekurencyjne, i przeprowadzimy dowód dla dopełnienia zbioru $\mathcal{B}$.

Ponieważ zbiór $\mathcal{B}$ jest niepusty, możemy również założyć, że istnieje język $L \in \mathcal{B}$ rozpoznawany przez maszynę Turinga M_L.

Dla maszyny Turinga M oraz słowa x konstruujemy maszynę M_x, dla której $L(M_x) = L$ lub $L(M_x) = \emptyset$. Maszyna M_x na słowie wejściowym y działa według następującego schematu:

- Symuluje działanie maszyny M na wejściu x.
- Jeśli maszyna M odrzuci x, M_x zatrzymuje się w stanie odrzucającym.
- Jeśli maszyna M zaakceptuje x, M_x rozpoczyna symulację maszyny M_L na wejściu y.
- Jeśli maszyna M_L zatrzyma się, M_x zwraca wynik jej działania.

Jeśli maszyna M zaakceptuje x, maszyna M_x, symulując maszynę M_L, akceptuje y lub nie zatrzymuje się. Zatem językiem rozpoznawanym przez maszynę M_x jest $L \in \mathcal{B}$.

Jeśli maszyna M nie zaakceptuje słowa x (odrzuci lub nie zatrzyma się), maszyna M_x nie nie zaakceptuje słowa y. Zatem językiem rozpoznawanym przez maszynę M_x będzie język pusty $\emptyset \notin \mathcal{B}$.

Otrzymaliśmy równoważność: $L(M_x) \in \mathcal{B}$ wtedy i tylko wtedy, gdy maszyna M akceptuje słowo x. Ponieważ problem akceptowania wejścia przez maszynę Turinga jest nierozstrzygalny, również problem „$L(M_x) \in \mathcal{B}$" jest nierozstrzygalny. ■

Ponieważ zbiory $\emptyset$ oraz $\mathbb{N}^n$ są rekurencyjne, twierdzenie Rice'a możemy zapisać równoważnie jako:

Twierdzenie 7.8 (Rice – wersja 3 – zbiory)

Zbiór $B = \{x \in \mathbb{N} : \ \phi_x \in \mathcal{B}\}$ jest rekurencyjny (problem „$\phi_x \in \mathcal{B}$" jest rozstrzygalny) wtedy i tylko wtedy, gdy $B = \emptyset$ lub $B = \mathbb{N}^n$.

Przykład 7.4

Problem wejścia: „$y \in D_x$" jest nierozstrzygalny.

Rozważmy zbiór $\mathcal{B} = \{f \in C_1 : \ y \in D_f\}$. Na mocy twierdzenia Rice'a wystarczy pokazać, że zbiór $\mathcal{B}$ jest właściwym i niepustym podzbiorem zbioru wszystkich funkcji obliczalnych. Istotnie, funkcja g_0 tożsamościowo równa 0 należy do zbioru $\mathcal{B}$, natomiast funkcja pusta $f_\emptyset$ do niego nie należy.

Przykład 7.5

Problem wyjścia: „$y \in \mathrm{Im}_x$" jest nierozstrzygalny.

Rozważmy zbiór $\mathcal{B} = \{f \in C_1 : \ y \in f(D_f)\}$. Zbiór $\mathcal{B}$ jest niepusty, ponieważ należy do niego funkcja g_y tożsamościowo równa y. Ponadto $\mathcal{B}$ jest właściwym podzbiorem C_1, ponieważ nie zawiera funkcji pustej $f_\emptyset$. Zatem na mocy twierdzenia Rice'a problem wyjścia jest nierozstrzygalny.