Języki, Automaty i Obliczenia

# Zadania

Część trzecia

Damian Niwiński i Wojciech Rytter

Styczeń 2002

# 1 Teoria obliczeń

## 1.1 Maszyny Turinga

1. Skonstruować maszynę Turinga obliczającą funkcję $2^n$ reprezentowaną *unarnie.* Bardziej dokładnie, zakładamy, że alfabet wejściowy składa się z jednego symbolu, powiedzmy $\{1\}$. Jeśli na wejściu dany jest ciąg $n$ jedynek, (tzn. konfiguracja wejściowa jest $q_0 1^n$), to po wykonaniu obliczenia konfiguracja powinna być $q_f 1^{2^n}$.

2. Skonstruować maszynę Turinga obliczającą funkcję $\lceil \log n \rceil$ reprezentowaną *unarnie.*

3. Przyjmijmy $\Sigma = \{0, 1\}$. Skonstruować maszyny Turinga rozpoznające następujące języki:

   (a) zbiór palindromów

   (b) $\{w\$w \,:\, w \in \Sigma^*\}$

   (c) $\{ww \,:\, w \in \Sigma^*\}$

   (d) zbiór ciągów reprezentujących binarnie liczby pierwsze.

4. Graf zorientowany o $n$ wierzchołkach ponumerowanych $0, 1, \ldots, n-1$, reprezentujemy ciągiem zer i jedynek długości $n^2$, takim, że $k$-ty bit jest 1 o ile istnieje krawędź z wierzchołka o numerze $i$ do wierzchołka o numerze $j$, gdzie $k = i \cdot n + j + 1$.

   (a) Skonstruować *niedeterministyczną* maszynę Turinga rozpoznającą zbiór słów zero-jedynkowych, które w powyższy sposób reprezentują te grafy, w których istnieje ścieżka z wierzchołka o numerze 0 do wierzchołka o numerze $n-1$.

   (b) Skonstruować maszynę *deterministyczną* realizującą to samo zadanie.

5. Przypomnijmy, że dwie maszyny Turinga o tym samym alfabecie wejściowym uważamy za *równoważne* o ile akceptują ten sam język. Udowodnić, że dla każdej maszyny Turinga istnieje równoważna jej maszyna posiadająca dokładnie jeden stan akceptujący

i taka, że w konfiguracji akceptującej głowica znajduje się nad pierwszą komórką taśmy (tzn. konfiguracja akceptująca jest postaci $q_f$ coś).

6. Dla danej *niedeterministycznej* maszyny Turinga skonstruować równoważną maszynę *deterministyczną*.

7. Dla danych deterministycznych maszyn Turinga $M_1$ i $M_2$ skonstruować deterministyczne maszyny rozpoznające języki
   - $L(M_1) \cup L(M_2)$
   - $L(M_1) \cap L(M_2)$
   - $L(M_1)L(M_2)$
   - $L(M_1)^*$.

8. Udowodnić, że dla każdej maszyny Turinga nad alfabetem wejściowym $\{0, 1\}$, istnieje równoważna jej maszyna, której alfabet *wszystkich* symboli roboczych obejmuje jedynie symbole 0,1,$B$.

9. Powiemy, że maszyna Turinga jest *write-once* jeśli może pisać tylko w pustych komórkach taśmy, a symbol raz napisany nie może być zastąpiony żadnym innym symbolem (w szczególności nie może być "zmazany" tj. zastąpiony przez "blank"). Dla dowolnej maszyny Turinga z jedną taśmą, skonstruować maszynę *write-once* z dwiema taśmami, która akceptuje ten sam język.

   (*) Dowieść, że maszyny typu *write-once* z *jedną* taśmą akceptują jedynie języki regularne.

10. Dla dowolnej maszyny Turinga (nad dowolnym alfabetem), skonstruować równoważną maszyne o *jednej taśmie* i *czterech* stanach. (Wolno powiększyć alfabet roboczy.)

11. Pojęcie automatu ze stosem można rozszerzyć do automatu z $k$ stosami, $k \geq 2$. Dowieść, że dla każdej maszyny Turinga istnieje równoważny jej deterministyczny automat z dwoma stosami. Wywnioskować stąd, że automat z $k$ stosami, $k > 2$ może być symulowany przez automat z 2 stosami (symulację tę można również opisać bezpośrednio).

12. Udowodnić, że jeśli automat skończony wyposażyć dodatkowo w kolejkę, to otrzymany model ma siłe obliczeniową maszyny Turinga, tzn. dla dowolnej maszyny istnieje automat z kolejką, który rozpoznaje ten sam język.

13. Automat z $k$ licznikami $c_1, \ldots, c_k$, określamy podobnie jak automat z $k$ stosami, z tym, że liczniki zawierają liczby naturalne, a dostępne operacje na licznikach są postaci $c_i := c_i + 1$, $c_i := c_i \dot{-} 1$ (gdzie $0 \dot{-} 1 = 0$), oraz test $c_i \stackrel{?}{=} 0$. Tzn. przejścia takiego automatu są postaci $q \to p$, $q, c_i \stackrel{?}{=} 0 \to p$, $q, c_i \stackrel{?}{\neq} 0 \to p$, $q \to p, c_i := c_i + 1$, $q \to p, c_i := c_i \dot{-} 1$. Nie ma taśmy, lecz zakładamy, że w chwili początkowej dana wejściowa jest wartością licznika $c_1$, a pozostałe liczniki mają wartość 0. Dowieść najpierw, że dla każdej maszyny Turinga nad alfabetem $\{1\}$ istnieje równoważny

jej automat z 4 licznikami. Następnie wykazać, że liczba liczników może być dalej zredukowana do trzech. (Ograniczenie alfabetu maszyny Turinga nie jest istotne, bo słowa nad alfabetem $n$–literowym można również łatwo "przerobić" na liczby naturalne w systemie unarnym.)

## 1.2 Obliczalność i nierozstrzygalność

1. Udowodnić równoważność następujących warunków

   - język $L$ jest częściowo obliczalny,
   - język $L$ jest dziedziną pewnej funkcji częściowo obliczalnej,
   - język $L$ jest zbiorem wartości pewnej funkcji częściowo obliczalnej,
   - język $L$ jest zbiorem wartości pewnej funkcji obliczalnej.

2. Udowodnić, że język $L$ jest obliczalny, wtedy i tylko wtedy, gdy jest skończony lub jest zbiorem wartości pewnej funkcji obliczalnej rosnącej.

3. Udowodnić następujące twierdzenie Turinga–Posta : jeśli zarówno język jak i jego uzupełnienie są częściowo obliczalne, to są również obliczalne.

4. Udowodnić, że istnieje zbiór częściowo obliczalny, którego uzupełnienie nie zawiera żadnego nieskończonego zbioru częściowo obliczalnego.

5. Udowodnić, że istnieje automat z dwoma (*sic !*) licznikami $A$ taki, że nie jest rozstrzygalne, czy $A$ zatrzymuje się dla danej wejściowej $n$. Wskazówka : wykorzystać zadanie 13.

6. Udowodnić nierozstrzygalność następującego *problemu Posta.*

   Dane są dwie listy słów : $u_1, \ldots, u_n \in \Sigma^*$ i $w_1, \ldots, w_n \in \Sigma^*$. Pytanie, czy istnieje ciąg indeksów $i_1, \ldots, i_m \in \{1, \ldots, n\}$, taki, że $u_{i_1} \ldots u_{i_n} = w_{i_1} \ldots w_{i_n}$ ? Jeśli taki ciąg istnieje, to słowo $u_{i_1} \ldots u_{i_n}$ $(= w_{i_1} \ldots w_{i_n})$ nazywamy *rozwiązaniem* danej instancji problemu Posta.

   Wskazówka : Rozważyć zmodyfikowany problem Posta, w którym dodatkowo wymaga się, by pierwszymi słowami w rozwiązaniu były $u_1$ i $w_1$ (tzn. $i_1 = 1$). Redukcja zmodyfikowanego problemu do oryginalnego problemu Posta nie jest trudna. Z kolei przedstawić algoerytm, który dla dowolnej maszyny Turinga $M$ i jej wejścia $w$ konstruuje instancję $P$ zmodyfikowanego problemu Posta w ten sposób, że $P$ ma rozwiązanie wtedy i tylko wtedy, gdy $M$ akceptuje $w$. Rozwiązaniem $P$ (o ile istnieje) będzie właśnie akceptujące obliczenie $M$ na $w$.

7. Dowieść, że następujące problemy są nierozstrzygalne :

   - Dana gramatyka bezkontekstowa $G$ nad alfabetem $\Sigma$. Pytanie : czy $L(G) = \Sigma^*$ ? (Tzw. problem *uniwersalności.*) Wskazówka : Dla maszyny Turinga $M$, *udane obliczenie* jest ciągiem $c_0\#c_1\#\ldots\#c_f$, gdzie $c_i$ są konfiguracjami $M$, $c_0$ jest konfiguracją początkową, $c_f$ akceptującą, oraz $c_i \vdash_M c_{i+1}$. Należy przedstawić algorytm, który dla dowolnej maszyny $M$ konstruuje gramatykę $G_M$ generującą wszystkie ciągi, które nie są udanymi obliczeniami. A zatem, gdybyśmy potrafili rozstrzygać problem uniwersalności, moglibyśmy również rozstrzygać problem niepustości ($L(M) \neq \emptyset$ ?) dla maszyn Turinga, co jest niemożliwe (na mocy twierdzenia Rice'a).

- Dane dwie gramatyki bezkontekstowe $G_1, G_2$. Pytanie : czy $L(G_1) \cap L(G_2) \neq \emptyset$ ? Wskazówka : Podobnie jak poprzednie zadanie, ale należy wykorzystać operację lustrzanego odbicia.
- Dana gramatyka bezkontekstowa $G$. Pytanie : czy $G$ jest jednoznaczna (tzn. dla każedego słowa w $L(G)$ istnieje dokładnie jedno drzewo wyprowadzenia).

  Wskazówka : wykorzystać problem Posta.

## 1.3 Hierarchia Chomsky'ego

1. Dowieść, że każdą gramatykę *monotoniczną*, czyli o regułach $\alpha \to \beta$, gdzie $|\beta| \geq |\alpha|$, można sprowadzić do postaci, w której każda reguła ma formę

   $$\alpha X \beta \to \alpha \gamma \beta$$

   gdzie $X \in V$, $\alpha, \beta, \gamma \in (\Sigma \cup V)^*$, $\gamma \neq \varepsilon$ ($V$ – zbiór symboli nieterminalnych, $\Sigma$ – zbiór symboli terminalnych).

   *Uwaga.* Przy powyższym sprowadzeniu trzeba na ogół powiększyć zbiór symboli nieterminalnych.

2. Dowieść, że języki kontekstowe, to dokładnie języki rozpoznawane przez niedeterministyczne *automaty liniowo ograniczone*.

   *Uwaga.* Automaty liniowo ograniczone są określone podobnie jak maszyny Turinga, z tym, że jedyną dostępną pamięcią są komórki taśmy zajęte na początku przez słowo wejściowe (przy czym koniec słowa jest zaznaczony specjalnym markerem).

3. Dowieść, że iloraz języka kontekstowego przez język regularny może nie być językiem obliczalnym.

   *Wskazówka.* Wykorzystać język obliczeń maszyny Turinga, która akceptuje język nieobliczalny.

   Wywnioskować dalej, że ani języki kontekstowe, ani języki obliczalne nie są zamknięte na ilorazy przez języki regularne.

4. Dowieść, że języki częściowo obliczalne **są** zamknięte na ilorazy przez języki częściowo obliczalne.

5. Które z czterech klas hierarchii Chomsky'ego są zamknięte na operacje $\cup, \cap, -$ ?

## 1.4 Złożoność obliczeniowa

1. *Konstruowalność pamięciowa.* Skonstruować maszynę Turinga *off-line*, która dla wejścia $w$ o długości $n$ zużywa dokładnie $f(n)$ komórek pamięci roboczej, gdzie $f(n) = 2n, n^2, n^k, 2^n, \log n, \ldots$.

2. *Konstruowalność czasowa.* Skonstruować maszynę Turinga, która dla wejścia $w$ o długości $n$ wykonuje dokładnie $f(n)$ kroków, gdzie $f(n) = 2n, 3n, n^2, n^k, 2^n, 2^{2^n}, \ldots$.

3. Dowieść, że klasy $P$, $NP$ i $PSPACE$ są zamknięte na operację $*$.

4. *Problem domina.* Instancja problemu domina obejmuje liczbę naturalną $M$ daną unarnie oraz skończony zbiór *wzorców kostek domina*, tzn. wektorów postaci (*up, down, left, right* ), gdzie *up, down, left* i *right* są binarnie zadanymi liczbami, które w tym kontekście nazywamy *kolorami*. (Liczba kolorów nie jest ograniczona, a więc zależy od danej instancji problemu.) Pytanie brzmi: czy kwadrat $M \times M$ można pokryć kostkami domina w ten sposób, że sąsiednie kostki stykają się bokami o tym samym kolorze, a na bokach kwadratu jest kolor 0 ("biały"). Zakładamy przy tym, że każdy wzorzec można wykorzystać dowolnie wiele razy, ale nie można go "obracać" (tzn. *up* jest zawsze górną krawędzią itd.).

   Należy udowodnić, że problem domina jest $NP$-zupełny. Wskazówka : zastosować redukcję generyczną, tzn. wychodząc od dowolnej niedeterministycznej maszyny Turinga działającej w czasie wielomianowym.

5. Udowodnić, że następujący problem *dokładnego pokrycia* (ang. *exact cover*) jest $NP$-zupełny. Dana rodzina podzbiorów $\{1, 2, \ldots, n\}$ (liczby dane binarnie). Pytanie: czy istnieje podrodzina podzbiorów rozłącznych, które w sumie dają $\{1, 2, \ldots, n\}$.

   Wskazówka : wykorzystać zadanie 4.

6. Udowodnić, że następujący problem *plecakowy* (ang. *knapsack problem*) jest $NP$-zupełny. Dane: liczby $n, m_1, \ldots, m_k$ (binarnie). Pytanie: czy istnieje podzbiór zbioru $\{m_1, \ldots, m_k\}$ taki, że $m_1 + \ldots + m_k = n$.

   Wskazówka : wykorzystać zadanie 5.

7. Dowieść, że złożenie dwóch funkcji obliczalnych w pamięci (przestrzeni) logarytmicznej jest funkcją obliczalną w pamięci logarytmicznej.

8. Dowieść, że każdy problem z klasy $NP$ redukuje się do problemu *3-CNF SAT* w pamięci logarytmicznej.