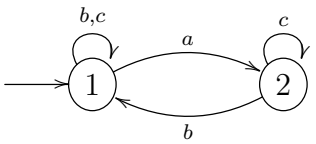
Języki, automaty i obliczenia
egzamin (zadania), przykładowe rozwiązania

Zad. 1. Niech L będzie językiem nad alfabetem $\{a, b, c\}$ zawierającym słowa spełniające poniższe warunki:

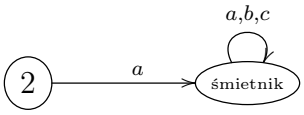
- pomiędzy każdymi dwoma literami a jest przynajmniej jedna litera b,
- pomiędzy każdymi dwoma literami b jest przynajmniej jedna litera c.

Czy minimalny automat deterministyczny dla języka L ma mniej niż 6 stanów?

Rozwiązanie. Tak. Oto automat $\mathcal{A}_1$, który sprawdza pierwszy warunek:



Obydwa stany są akceptujące. Można z niego łatwo uczynić automat deterministyczny, poprzez dodanie dodatkowego stanu nieakceptującego „śmietnik" i krawędzi:



Analogiczny automat $\mathcal{A}_2$ sprawdza drugi warunek.

Język L jest rozpoznawany przez automat produktowy $\mathcal{A}_1 \times \mathcal{A}_2$, który ma 4 stany; automat deterministyczny uzyskujemy przez dodatnie stanu śmietnik, podobnie jak powyżej. Zatem język L jest rozpoznawany przez automat deterministyczny o 5 stanach.

Zad. 2. Dla automatu deterministycznego $\mathcal{A}$ i słowa w, niech $\#_{\mathcal{A}}(w)$ oznacza liczbę odwiedzin stanów akceptujących w biegu automatu $\mathcal{A}$ na słowie w. Podaj algorytm wielomianowy dla następującego problemu:

> *Dane:* dwa automaty deterministyczne $\mathcal{A}$, $\mathcal{B}$ nad alfabetem A.
>
> *Pytanie*: czy istnieje słowo $w \in A^*$ takie, że $\#_{\mathcal{A}}(w) > \#_{\mathcal{B}}(w)$?

Rozwiązanie. Algorytm buduje automat ze stosem $\mathcal{D}$ rozpoznający język

$$L = \{w \in A^* : \#_{\mathcal{A}}(w) > \#_{\mathcal{B}}(w)\},$$

a następnie sprawdza, czy język tego automatu jest niepusty.

Najpierw opiszemy konstrukcję automatu $\mathcal{D}$. Rozważmy automat produktowy $\mathcal{C} = \mathcal{A} \times \mathcal{B}$. Przeróbmy go na automat ze stosem $\mathcal{D}$, który po przeczytaniu słowa w przechowuje na stosie liczbę

$$n(w) = \#_{\mathcal{A}}(w) > \#_{\mathcal{B}}(w).$$

W jaki sposób? Na dnie stosu jest cały czas symbol początkowy $\perp$, a ponadto:

- jeśli $n(w) \geq 0$, stos zawiera dodatkowo n symboli $+$;
- jeśli $n(w) \leq 0$, stos zawiera dodatkowo n symboli $-$.

Stany automatu $\mathcal{D}$ to stany automatu $\mathcal{C}$. Stan początkowy też. Dodatkowo, $\mathcal{D}$ ma stan akceptujący „ok".

Jeśli dokładnie jeden ze stanów początkowych automatów $\mathcal{A}$ i $\mathcal{B}$ jest akceptujący, automat $\mathcal{D}$ w pierwszym kroku, za pomocą pustego przejścia, wkłada na stos jeden symbol $+$ albo $-$. Pozostałe przejścia automatu $\mathcal{D}$ to dokładnie przejścia automatu $\mathcal{C}$, ale mające następujący efekt uboczny: jeśli po wykonaniu przejścia dokładnie jeden ze stanów automatów $\mathcal{A}$, $\mathcal{B}$ jest akceptujący, to dodajemy albo usuwamy jeden symbol $+$ albo $-$ ze stosu, w zależności od tego, jaki symbol jest na szczycie stosu. Ponadto, w każdym stanie różnym od „ok" automat $\mathcal{D}$, widząc symbol $+$ na szczycie stosu może przejść za pomocą pustego przejścia do stanu „ok". Z konstrukcji wynika, że $L(\mathcal{D}) = L$. Rozmiar $\mathcal{D}$ jest wielomianowy względem rozmiarów $\mathcal{A}$ i $\mathcal{B}$.

Aby sprawdzić, czy $L(\mathcal{D})$ jest niepusty, algorytm przekształca $\mathcal{D}$ w równoważną gramatykę bezkontekstową $\mathcal{G}$ (rozmiaru wielomianowego względem rozmiaru $\mathcal{D}$). Niech $\mathcal{N}$ oznacza zbiór nieterminali. Aby sprawdzić niepustość gramatyki, algorytm posługuje się podzbiorem $Z \subseteq \mathcal{N}$; początkowo, $Z = \emptyset$. Zbiór ten zawiera te nieterminale, które dotychczas zostały uznane przez algorytm za niepuste (tzn. nieterminale X takie, że gramatyka $\mathcal{G}$, w której za symbol startowy uznajemy X, generuje niepusty język). Algorytm powtarza następujący krok tak długo, jak to możliwe:

> dodaj do Z nieterminal X, o ile X ma produkcję taką, że wszystkie nieterminale występujące po prawej stronie należą już do Z.

Liczba kroków jest wielomianowa, a każdy z nich można zrealizować w czasie wielomianowym przeszukując wszystkie produkcje. Algorytm odpowiada „tak", jeśli po zakończeniu powyżej opisanej iteracji symbol startowy gramatyki $\mathcal{G}$ należy do Z.

Zad. 3. Ustalmy alfabet $\{a, b\}$ i porządek $a < b$. Niech $\texttt{sort}(w)$ oznacza wynik *posortowania* słowa w, np. $\texttt{sort}(abbaa) = aaabb$. Zdefiniujmy operację na językach:

$$\texttt{sort}(L) = \{\texttt{sort}(w) \ : \ w \in L\}.$$

Czy dla każdego języka $L \in \text{NSPACE}(n)$, język $\texttt{sort}(L) \in \text{NSPACE}(n)$?

Przypomnienie: klasa NSPACE(n) zawiera języki rozpoznawane przez niedeterministyczne maszyny Turinga w pamięci rozmiaru n, gdzie n to długość słowa wejściowego.

Rozwiązanie. Tak. Niech $L \in \mathrm{NSPACE}(n)$ i niech $\mathcal{M}$ będzie maszyną rozpoznającą L w pamięci $s(n) = n$. Skonstruujemy maszynę $\mathcal{M}'$ rozpoznającą język $\mathtt{sort}(L)$. Chcemy, aby zachodził warunek (*): maszyna $\mathcal{M}'$ akceptuje słowo w wtedy, i tylko wtedy gdy $w = \mathtt{sort}(v)$, dla pewnego $v \in L$. Warunek ten gwarantuje, że $L(\mathcal{M}') = \mathtt{sort}(L)$.

Obliczenie maszyny $\mathcal{M}'$ składa się z trzech faz. W pierwszej fazie obliczenia, maszyna $\mathcal{M}'$ sprawdza, czy słowo wejściowe w jest posortowane, tzn. czy wszystkie a leżą na lewo od wszystkich b. Następnie, druga faza przeznaczona jest na przetworzenie słowa w na niedeterministycznie wybraną permutację v słowa w, tzn. słowo v które ma tyle samo liter a i b co słowo w. Faza ta jest zrealizowana poprzez powtarzanie następującej czynności w nieskończoność:

> Niedeterministycznie wybierz jedną z dwóch możliwości: albo zakończ fazę drugą i przejdź do fazy trzeciej, albo wybierz niedeterministycznie dwie pozycje słowa i zamień litery na tych pozycjach (co maszyna realizuje przechodząc przez słowo tam i spowrotem).

Faza trzecia polega na uruchomieniu maszyny $\mathcal{M}$; akceptacja ma miejsce, gdy maszyna $\mathcal{M}$ jest w stanie akceptującym.

Zad. 4. Pokaż nierozstrzygalność następującego problemu decyzyjnego.

> *Dane:* język bezkontekstowy L nad alfabetem A.
>
> *Pytanie*: czy $A^* \setminus L$ jest skończony?

Rozwiązanie. Redukcja z problemu uniwersalności dla języków bezkontekstowych. Zdefiniujemy funkcję obliczalną f, która przerabia język bezkontekstowy L nad A na język bezkontekstowy $f(L)$ nad $A \cup \{\$\}$, i spełnia warunek:

$$L = A^* \quad \text{wtedy, i tylko wtedy gdy} \quad (A \cup \{\$\})^* \setminus f(L) \text{ skończony},$$

dla każdego języka bezkontekstowego L. Symbol \$ wybieramy dowolnie, ale tak, żeby $\$ \notin A$. Nie jest istotne, czy język L jest reprezentowany przez gramatykę czy przez automat ze stosem. Niech

$$f(L) = L \ \cup \ L\$(A \cup \{\$\})^*.$$

Jeśli $L = A^*$, to $f(L) = (A \cup \{\$\})^*$, czyli $(A \cup \{\$\})^* \setminus f(L)$ jest pusty, a więc skończony. Jeśli $L \neq A^*$, to dla każdego $w \in A^* \setminus L$ język $f(L)$ zawiera np. wszystkie słowa $w\$^n$, dla $n \in \mathbb{N}$, więc jest nieskończony.