Języki, Automaty i Obliczenia

Michał Balcerzak
Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski, Polska

April 17, 2018

Treść zadania

Powiemy, że automat niedeterministyczny $\mathcal{A}$ jest *k-niedeterministyczny*, gdzie $k \in \mathbb{N}$, jeżeli dla każdego słowa wejściowego w, automat $\mathcal{A}$ ma co najwyżej k biegów po słowie w zaczynających się w stanie początkowym.

Pokazać, że istnieje algorytm, który otrzymawszy 10-niedeterministyczny automat $\mathcal{A}$, w czasie wielomianowym produkuje równoważny mu automat deterministyczny

Dokładniej, czas działania algorytmu dla danego 10-niedeterministycznego automatu $\mathcal{A}$ ma być ograniczony przez $poly(|Q| + |A| + |\delta|)$, gdzie $poly : \mathbb{N} \to \mathbb{N}$ jest pewnym wielomianem, oraz Q, A, δ to, odpowiednio: zbiór stanów, alfabet i zbiór tranzycji automatu $\mathcal{A}$. Nie jest sprecyzowane jak algorytm ma działać, gdy wejściowy automat $\mathcal{A}$ nie jest 10-niedeterministyczny (może wtedy np. działać w czasie wykładniczym, produkować nierównoważny automat, lub zwracać błąd).

Z rozwiązania powinno jasno wynikać istnienie takiego algorytmu. W szczególności, należy dokładnie uzasadnić poprawność algorytmu i jego czas działania. Algorytm można opisać pseudokodem lub słowami, lub jego istnienie może być wnioskiem z postaci przeprowadzonej konstrukcji matematycznej.

Rozwiązanie

Bieg b_w automatu 10-niedeterministycznego $\mathcal{A}$ po pewnym słowie $w = a_1a_2a_3 \dots a_n$ to ciąg tranzycji zawartych w automacie $\mathcal{A}$:

$$b_{w,0} \xrightarrow{a_1} b_{w,1} \xrightarrow{a_2} b_{w,2} \xrightarrow{a_3} \dots \xrightarrow{a_n} b_{w,n}$$

gdzie $b_{w,0}, b_{w,1}, \dots, b_{w,n} \in Q$.

Powiemy, że dwa biegi automatu $\mathcal{A}$ po pewnym słowie w – b_w oraz c_w – są *różne* wtedy i tylko wtedy gdy $\exists_{i \in \mathbb{N}_{|w+1|}} b_{w,i} \neq c_{w,i}$ (gdzie $\mathbb{N}_i = \{0, 1, \dots, i-1\}$).

Z kolei przez to, że $\mathcal{A}$ ma k biegów po słowie w będziemy rozumieć to, że każde dwa biegi spośród biegów $\mathcal{A}$ po słowie w są różne.

Stanem końcowym biegu b_w nazwiemy $b_{w,|w|}$.

Wniosek 1: Po wczytaniu pewnego słowa $w \in A^*$ do $\mathcal{A}$ możemy zakończyć bieg w co najwyżej 10 różnych stanach.

Dowód: W przeciwnym przypadku musiałoby istnieć przynajmniej 11 biegów $\mathcal{A}$ po słowie w, spośród których każdy musiałby mieć inny stan końcowy. To jednak nie jest możliwe, gdyż wtedy każdy z tych biegów byłby różny od dowolnego z pozostałych, więc $\mathcal{A}$ musiałby mieć co najmniej 11 biegów po słowie w, co byłoby sprzeczne z jego własnością 10-niedeterministyczności. □

Przyjrzyjmy się procesowi determinizacji automatów niezdeterminizowanych przytoczonemu na wykładzie. Dla przypomnienia – automat deterministyczny $(A, Q', q_0, F', \delta')$ otrzymamy z automatu niedetermnistycznego $(A, Q, I_\mathcal{A}, F_\mathcal{A}, \delta)$ biorąc:

- $Q' = P(Q)$ (zbiór potęgowy Q)
- $q_0 = I_\mathcal{A}$
- $F' = \{S \mid S \subseteq Q' \wedge S \cap F_\mathcal{A} \neq \emptyset\}$
- $\delta' = f \in Q'^{(Q' \times A)}$ takie, że $f(S, a) = \{q \in Q \mid \exists_{p \in S}(p, a, q) \in \delta\}$

Fakcik 1: Skrypt wykładowy mówi, że dla dowolnego $w \in A^*$ stan $q_0 \cdot w$ jest równy zbiorowi tych stanów, które można osiągnąć w automacie $\mathcal{A}$ zaczynając w którymś ze stanów początkowych i wczytując słowo w.

Niech automat $\mathcal{A}' = (A, Q', q_0, F', \delta')$ stanowi bezpośrednią determinizację automatu $\mathcal{A} = (A, Q, I_\mathcal{A}, F_\mathcal{A}, \delta)$.

Lemat 1: Dla każdego osiągalnego stanu $q_{\mathcal{A}'}$ automatu $\mathcal{A}'$ zachodzi $|q_{\mathcal{A}'}| \leq 10$.

Dowód: Dowolny stan $q_{\mathcal{A}'} \in Q'$ jest osiągalny wtedy i tylko wtedy, gdy istnieje $w \in A^*$ że $q_0 \cdot w = q_{\mathcal{A}'}$. Na podstawie fakciku 1 każdy osiągalny stan $q_{\mathcal{A}'}$ jest więc tożsamy z pewnym zbiorem T stanów osiągalnych poprzez wczytanie pewnego w do $\mathcal{A}$. Z kolei na podstawie wniosku 1 możemy stwierdzić, że zbiór T musi mieć moc niewiększą niż 10 - a więc moc tegoż dowolnie wybranego, osiągalnego stanu $q_{\mathcal{A}'}$ również ma takąż własność, co kończy dowód. □

Automaty są równoważne gdy akceptują ten sam język. Na mocy wykładu $\mathcal{A}'$ jest równoważny automatowi $\mathcal{A}$. Stwórzmy na podstawie $\mathcal{A}'$ automat $\mathcal{B}$ będący, opisując intuicyjnie, tożsamy z tą częścią $\mathcal{A}'$, której stany mają moce niewiększe niż 10. Automat $\mathcal{B} = (A, \breve{Q}, q_0, \breve{F}, \breve{\delta})$ zdefiniujemy następująco:

- $\breve{Q} = \{S \mid S \in Q' \wedge |S| \leq 10\} \cup \{\perp\}$
- $\breve{F} = F' \cap \breve{Q}$
- $\breve{\delta} = f \in \breve{Q}^{(\breve{Q} \times A)}$ takie, że

$$f(S,a) = \begin{cases} \delta'(S,a), & \text{gdy } S \neq \perp \wedge |\delta'(S,a)| \leq 10 \\ \perp, & \text{w przeciwnym przypadku} \end{cases}$$

Niech napis $\mathcal{C}(w)$ oznacza dla automatu $\mathcal{C}$ i słowa $w \in A^*$ stan osiągany po wczytaniu w przez automat $\mathcal{C}$.

Lemat 2: Automat $\mathcal{B}$ jest równoważny automatowi $\mathcal{A}$, czyli

$$\bigvee_{w \in A^*} \mathcal{B}(w) \in \breve{F} \Leftrightarrow \mathcal{A}(w) \in F_{\mathcal{A}}$$

Dowód: Weźmy dowolne $a_1a_2a_3 \dots a_n = w \in A^*$. Wiemy że $\mathcal{A}(w) \in F_{\mathcal{A}} \Leftrightarrow \mathcal{A}'(w) \in F'$. Wówczas:

- $\mathcal{B}(\varepsilon) = \mathcal{A}'(\varepsilon) = q_0$
- Jeśli dla pewnego v – i-literowego prefiksu w gdzie $i \in \{0, 1, \dots, n-1\}$ – zachodzi $\mathcal{A}'(v) = \mathcal{B}(v) = q$, to $\breve{\delta}(q, a_{i+1}) = \delta'(q, a_{i+1})$, gdyż
  - $q \neq \perp$, bo $q = \mathcal{A}'(v) \notin Q'$
  - $\delta'(q, a_{i+1})$ jest osiągalny w $\mathcal{A}'$, bo q jest osiągalny, a więc na mocy lematu 1 $|\delta'(q, a_{i+1})| \leq 10$

  Stąd, na mocy indukcji wynika $\mathcal{B}(w) = \mathcal{A}'(w)$. Ponadto stan $\mathcal{A}'(w)$ jest osiągalny, więc z własności $\mathcal{B}$ należy również do $\breve{Q}$, więc

$$\mathcal{A}(w) \in F_{\mathcal{A}} \Leftrightarrow \mathcal{A}'(w) \in F' \Leftrightarrow \mathcal{B}(w) \in \breve{Q} \wedge \mathcal{B}(w) \in F' \Leftrightarrow \mathcal{B}(w) \in \breve{F}$$

□

Fakcik 2: Spróbujmy oszacować $|\breve{Q}|$. Dla $m = |Q|$ zachodzi:

- $|Q'| = \sum_{i=0}^{m} \binom{m}{i}$
- $$|\breve{Q}| = 1 + \sum_{i=0}^{10} \binom{m}{i} = 1 + \sum_{i=0}^{10} \frac{m!}{i!(m-i)!} = 1 + \sum_{i=0}^{10} \frac{\prod_{k=0}^{i-1} m-k}{i!} \leq$$
  $$\leq 1 + \sum_{i=0}^{10} \prod_{k=0}^{i-1} m = 1 + \sum_{i=0}^{10} m^i \leq 1 + \sum_{i=0}^{10} m^{10} = 11m^{10} + 1 = O(m^{10})$$
- $|\breve{Q}| \leq O(m^{10}) = O(|Q|^{10})$

Algorytm 1: Automat $\mathcal{B}$ można wygenerować w czasie wielomianowym.

Przyjmijmy że mamy bezpośredni dostęp do zbiorów Q, A, δ przytoczonych w treści zadania. Przyjmijmy również, że struktura zbioru udostępnia

- konstruktor pustego zbioru o stałej złożoności czasowej
- procedury przeszukiwania zawartości oraz wstawiania elementów, obydwie o czasie liniowym względem rozmiaru przetrzymywanych danych

Takie złożoności czasowe są możliwe do osiągnięcia np. przy implementacji zbioru za pomocą pojedynczo linkowanej listy (liczymy na to, że nie dotyczy nas ekstremalny, o ile w ogóle możliwy przypadek, w którym ewentualna konwersja udostępnionej nam struktury do takiej struktury zbioru nie byłaby możliwa do wykonania w wielomianowym czasie).

1. Pierwszym krokiem algorytmu bedzie wygenerowanie zbioru $\breve{Q}$. Jego reprezentantem w naszym kodzie będzie `QB`. Z definicji ma on być zbiorem wszystkich podzbiorów Q o mocy niewiększej niż 10 (plus śmietnik).

```
1  QB := {⊥, {}}
2  elems := {{}}  # wszystkie (i-1)-elementowe podzbiory Q
3  temp := {}  # tu na podst. elems generujemy i-elem. podzb. Q
4
5  for (i := 1; i <= 10; i++):
6      for e in elems:
7          for q in Q:
8              e.insert(q)
9              temp.insert(e)
10     for t in temp:
11         QB.insert(t)
12     swap(temp, elems)
13     temp := {}
```

Za pomocą trywialnej indukcji można pokazać, że komentarze z linii 2-3 są niezmiennikami pętli z linii 5 w momencie gdy program dochodzi do linii 10. Wobec tego `temp`, `elems` $\subseteq \breve{Q}$, więc $|\texttt{temp}| + |\texttt{elems}| \leq O(|Q|^{10})$. Ponadto, gdy przez $\texttt{temp}_j$ oznaczymy zawartość struktury `temp` w momencie

gdy program jest w linii 10 a wartość `i` wynosi j, wówczas dzięki liniom 10-11 zachodzi

$$\texttt{QB} = (\sum_{j=1}^{10} \texttt{temp}_j) \cup \{\bot, \varnothing\} = \sum_{j=0}^{10} \{S \mid S \in P(Q) \wedge |S| = j\} \cup \{\bot\} =$$
$$= \{S \mid S \in Q' \wedge |S| \leq 10\} \cup \{\bot\} = \breve{Q}$$

(a więc zgodnie z definicją).

Powyższy kod składa się z trzech czasowo liniowych, zagnieżdżonych przejść po zbiorach, kolejno $\mathbb{N}_{10}$, `elems` i Q, gdzie $|\texttt{elems}| \leq |\breve{Q}| \cdot |Q| \leq O(|Q|^{11})$ i $|\mathbb{N}_{10}| = O(1)$. Z kolei czas każdej z podprocedur `insert` jest ograniczony przez rozmiar struktur, na których jest wykonywana, a więc co najwyżej $O(|Q|^{10})$. Sumaryczny czas wykonania tego programu T_1 będzie więc rzędu co najwyżej $O(|Q|^{11} \cdot 1 \cdot |Q| \cdot |Q|^{10}) \leq O(|Q|^{30})$

2. Wyliczenie $\texttt{FB} = \breve{F}$ jest trywialne.

```
1 FB := QB.filter(q => {
2     for e in q:
3         for f in F_A:
4             if f == e: return true
5     return false
6 })
```

Procedura `filter` jest możliwa do wyrażenia za pomocą fora i jednego bufora, kod zapisałem w ten sposób tak, by zachować czytelność. Złożoność czasowa tej procedury jest identyczna ze złożonością przejścia `for`em po strukturze `QB`.

`FB` po przejściu algorytmu będzie zawierał te elementy $\breve{Q}$, które mają część wspólną z $F_{\mathcal{A}}$ (bo istnieje element zawarty w obydwu zbiorach), więc $\texttt{FB} = \{S \mid S \subseteq \texttt{QB} \wedge S \cap F_{\mathcal{A}} \neq \emptyset\} = \{S \mid S \subseteq \breve{Q} \wedge S \cap F_{\mathcal{A}} \neq \emptyset\} = \{S \mid S \subseteq Q' \wedge S \cap F_{\mathcal{A}} \neq \emptyset\} \cap \breve{Q} = F' \cap \breve{Q} = \breve{F}$.

Czas wykonania się powyższego kodu oczywiście jest wielomianowy z racji rozmiarów przeglądanych struktur oraz metod na nich wykonywanych - możemy zastosować rozumowanie podobne do przeprowadzonego wyżej. Powinno nam wyjść $T_2 \leq O(|\texttt{QB}| \cdot 10 \cdot |F_{\mathcal{A}}|) \leq O(|Q|^{10} \cdot |Q|) \leq O(|Q|^{11})$ (zauważamy że $\texttt{QB} = \breve{Q}$ oraz $F_{\mathcal{A}} \subseteq Q$).

3. W końcu postaramy się wyliczyć funkcję przejścia $\breve{\delta}$. Jest to ostatnia rzecz potrzebna nam do tego, żeby automat zadziałał - pozostałe rzeczy, czyli alfabet A oraz stan początkowy $q_0 = I_{\mathcal{A}}$ można łatwo wywnioskować z danego nam na początku automatu $\mathcal{A}$.

Do spamiętania wartości funkcji $\breve{\delta}$ możemy zastosować np. listową implementację słownika `dict(p ∈ QB -> dict(α ∈ A -> q ∈ QB))`. Sprawdzanie wartości tak zapisanej funkcji, a także ustawienie nowej wartości dla pewnej pary argumentów da się zrealizować w czasie liniowym względem rozmiaru danych, czyli w czasie $O(|\texttt{QB}| \cdot |A|) = O(|A| \cdot |Q|^{10})$.

```
1  DB := dict()
2
3  for s in QB:
4      DB[s] := dict()
5      for a in A:
6          temp := {}  # wszystkie stany osiagalne
7                      # z podstanow s po wczytaniu a
8          for q in s:
9              for el in δ:
10                 if el[0] == q && el[1] == a:
11                     temp.insert(el[2])
12         if s == ⊥ || temp.size > 10:
13             temp := ⊥
14         DB[s][a] := temp
```

Zachowane są niezmienniki pętli:

- 9 - po zakończeniu pętli `temp` zawiera wszystkie stany osiągalne przez wczytanie `a` do $\mathcal{A}$ w stanie `q`
- 8 - po wykonaniu spełniony jest komentarz z linii 6-7
- 5 - `DB[s][a]` zawiera $\breve{\delta}(\texttt{s}, \texttt{a})$
- 3 - $\forall_{s \in \breve{Q}} \forall_{a \in A} \texttt{DB}[s][a] = \breve{\delta}(s, a)$

Trzy zagnieżdżone pętle przeglądają struktury poszczególnych zbiorów w łącznym czasie $O(|\texttt{QB}| \cdot |A| \cdot 10 \cdot |\delta|) = O(|Q|^{10} \cdot |A| \cdot |\delta|)$. Z kolei rozmiar `temp`, a więc i czas wykonywania się najbardziej kosztownej procedury (linia 11) jest na pewno ograniczony przez $O(3+|\breve{Q}|)$, więc $T_3 \leq O(|Q|^{30} \cdot |A| \cdot |\delta|)$.

Koniec końców czas działania naszego algorytmu $T = \sum_{i\in\{1,2,3\}} T_i \leq O(|Q|^{30}) + O(|Q|^{11}) + O(|Q|^{30} \cdot |A| \cdot |\delta|) = O(|Q|^{20}(|Q|^{10} + |A| \cdot |\delta|) \leq O(h^{20}(h^{10} + h \cdot h) = O(h^{30})$ gdzie $h = |Q| + |A| + |\delta|$. Istnieje więc algorytm generujący automat deterministyczny z automatu 10-niedeterministycznego, którego czas jest ograniczony przez wielomian zależny od czynników h, co było do pokazania w tym zadaniu.

Oczywiście wygenerowanie tego automatu jest możliwe przy dużo mniejszej złożoności czasowej, powyższy algorytm można dużo lepiej zoptymalizować, jednak nie było to celem zadania. Starałem się dobierać jak najprostsze struktury, a moje oszacowania złożoności czasowej są bardzo zgrubne.

W niniejszym opisie algorytmu mogłem czasem przywiązywać mniejszą uwagę do problemu złożoności czasowej głębokiego kopiowania struktur (m.in. nie skupiałem się na tym, kiedy płytka kopia mogłaby przyspieszyć program - patrz akapit wyżej). Poszczególne z tych procedur są jednak wykonywane na strukturach o rozmiarze, który da się ograniczyć przez wielomianową kombinację którychś z parametrów $|Q|, |A|, |\delta|$, a głebokie kopiowanie tych struktur da sie wykonać w czasie wielomianowym ze względu na rozmiar przetrzymywanych w nich danych. Suma oraz iloczyn dwóch wielomianów daje wielomian, dzięki czemu to zagadnienie również nie powinno mieć znaczenia dla faktu realizowalności tego zadania w czasie wielomianowym.