JAiO lato 2024

rozwiązanie zadań z serii IV

Konrad Kaczmarczyk

11 June 2024

§1 Zadanie

Zadanie 1.1. Słowa postaci ww nazywamy *kwadratami*. Dla języka $L \subseteq \Sigma^*$ zdefiniujmy operację *ukwadratowienia*:

$$\square L := L \cap \{ww : w \in \Sigma^*\}$$

W problemie *niepustosci wyrażeń regularnych* pytamy, czy język definiowany przez dane wyrażenie regularne jest niepusty. Pokaż NP-zupełność problemu niepustości dla wyrażeń zbudowanych z liter $a \in \Sigma$, słowa pustego ε, operacji konkatenacji, sumy $+$ i ukwadratowienia $\square$. Zatem w wyrażeniach tych nie wolno używać *, a zamiast tego dysponujemy operacją ukwadratowienia. Przykładową instancją problemu jest wyrażenie

$$(\square\,(a\,(b + aaa)) + \varepsilon)\,a$$

definiujące język $\{aaaaa, a\}$.

Wskazówka: Na rozgrzewkę, pomocne może być rozważenie wyrażeń zbudowanych z liter $a \in \Sigma$, słowa pustego ε, operacji konkatenacji, sumy $+$ i przecięcia $\cap$. W tym wariancie, przykładową instancją jest wyrażenie

$$((ab \cap a(b + ac)) + \varepsilon)\,a$$

§2 Rozwiązanie

Aby udowodnić, że problem *niepustosci ukwadratowianych wyrażeń pseudo-regularnych* (jak to będe dalej nazywał), wykażemy jego przynależnosć do klasy NP, jak i jego NP-trudnosć

Przynależnosć do klasy NP

Niech certyfikatem będzie słowo w i jesli L (czyli ukwadratowione wyrażenie pseudo-regularne) jest niepuste, to możemy w czasie wielomianowym sprawdzić czy $w \in L$, mianowicie:

(a) Tworzymy język L' w którym opuszczamy wszystkie *kwadraty*:

$$(\square\,(a(b + aaa)) + \varepsilon)\,a \to ((a(b + aaa)) + \varepsilon)\,a$$

(b) Możemy wielomianowo sprawdzić czy $w \in L'$, i jesli tak jest to znamy przebieg słowa po tym wyrażeniu (które możemy uproscić do automatu niedeterministycznego, gdzie możemy zgadywać jego przebieg).

(c) znając ten przykładowy przebieg sprawdamy czy słowa wygenerowane w *kwadratach* są w postaci $ww : w \in \Sigma^*$ (jesli są zagnieżdżone to sprawdamy rekurencyjnie)

(d) Jesli, wszystkie wyrazenia są w odpowiedniej formie to wiemy że $w \in L$, w przeciwnym przypadku sprawdzamy kolejne przejscie.

Algorytm ten dla jednego słowa wykonuje się w czasie $P + Q \cdot |w|$ gdzie P, Q są wielomianami.

NP-trudnosć

Aby wykazać NP-trudność, możemy dokonać redukcji z innego znanego problemu NP-zupełnego, na przykład problemu spełnialności formuł logicznych (SAT).

Redukcja z problemu SAT

- **Problem 3SAT:** Dany jest zbiór zmiennych $x_1, x_2, \ldots, x_n$ oraz formuła logiczna w postaci koniunkcji klauzul (3CNF), gdzie każda klauzula jest alternatywą 3 literałów.
- **Konstrukcja wyrażeń regularnych:**
  - Zamieniamy każdą zmienną x_i oraz jej negację $\neg x_i$ na litery alfabetu Σ. Przykładowo, $x_i \to a_i$ oraz $\neg x_i \to b_i$.
  - Dla każdej klauzuli $(y_k \vee y_l \vee y_m)$, gdzie y_i jest literą odpowiadającą literałowi w klauzuli, tworzymy wyrażenie regularne

$$y_k \prod_{i \neq k}(a_i + b_i) + y_l \prod_{i \neq l}(a_i + b_i) + y_m \prod_{i \neq m}(a_i + b_i)$$

  - Potraktujmy na chwile powstałe wyrażenia jako języki $L_1, \ldots L_p$, dopełnijmy je wyrażeniami $K = (a_1 + b_1 + \varepsilon)(a_2 + b_2 + \varepsilon) \ldots (a_n + b_n + \varepsilon)$ tak aby powstała ich liczba była potęgą dwójki, i możemy założyć że $p = 2^c$
  - Wprowadzamy nowy symbol x, i zauważmy że operacja $\Box(L_i x L_{2^{c-1}+i} x)$ tworzy języki L'_i które po dekompozycji $LxLx = L'_i$, spełniają formuły $L \in L_i, L \in L_{2^{c-1}+i}$. W ten sposób możemy zredukować liczbę wyrażeń dzieląc ich ilosc przez 2,
  - Powtarzamy proces do momentu gdy zostanie jeden język i nazwijmy go M

  Jeśli $M \neq \varnothing$, istnieje słowo, które na przecięciu wszystkich L_i, co oznacza że spełnia ono wszystkie formuły, więc oryginalna formuła 3SAT jest spełnialna.

Wniosek

Problem *niepustosci ukwadratowianych wyrażeń pseudo-regularnych* jest NP-zupełny. Wykazaliśmy, że problem jest w klasie NP i przeprowadziliśmy redukcję z problemu SAT, co pokazuje, że problem jest NP-trudny.