Konstrukcja kompilatorów

Lista zadań nr 1

Na zajęcia 14 października 2020

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytłuszczoną** czcionką.

Zadanie 1. Skonstruuj **automat skończony** rozpoznający komentarz w języku C, czyli ciąg znaków rozpoczynający się znakami *//* i zakończony znakiem *\n* lub rozpoczynający się znakami */**, kończący się znakami **/* i nie zawierający wewnątrz podciągu **/*.

Zadanie 2. Napisz **wyrażenie regularne** reprezentujące komentarz w języku C.

Zadanie 3. Na slajdzie numer 58 (w module *wyrażenia regularne*) jest definicja wyrażenia opisującego stałe liczbowe w Pascalu. Narysuj automat odpowiadający temu wyrażeniu. Zadbaj o to, żeby był on **deterministyczny**.

Zadanie 4. Rozważmy rozszerzenie języka C o zagnieżdżone komentarze. Chcielibyśmy, żeby napis

/ zagnieżdżony /* komentarz */ */*

był poprawnym komentarzem, i żeby można było zagnieżdżać dowolnie wiele razy.
Uzasadnij, że nie istnieje deterministyczny automat skończony rozpoznający takie komentarze.
Wskazówka: Automat deterministyczny o n stanach nie odróżni od siebie pewnych dwóch prefiksów słowa $(/*)^{n+1}$ i można go oszukać — dopisać taki sufiks, który automat zaakceptuje, mimo że całe słowo nie jest poprawnym komentarzem.
Wskazówka 2: To zadanie jest łatwe i wcale nie trzeba uczyć się na pamięć dowodu lematu o pompowaniu.

Na wykładzie powiedzieliśmy, że dla każdego niedeterministycznego automatu skończonego istnieje równoważny (tzn. rozpoznający ten sam język) automat deterministyczny. Determinizacja automatu $\langle \Sigma, S, s_0, F, \Delta \rangle$, gdzie Σ jest alfabetem, S zbiorem stanów, $s_0 \in S$ stanem początkowym, $F \subseteq S$ zbiorem stanów akceptujących, a $\Delta \subseteq S \times \Sigma \times S$ *relacją* przejścia, polega na

- skonstruowaniu automatu $\langle \Sigma, \mathcal{P}(S), \{s_0\}, \{Q \subseteq S \mid Q \cap F \neq \emptyset\}, \delta \rangle$, gdzie $\delta : \mathcal{P}(S) \times \Sigma \to \mathcal{P}(S)$ jest *funkcją* przejścia zdefiniowaną wzorem $\delta(Q, a) = \{s \in S \mid \exists q \in Q.\ \langle q, a, s \rangle \in \Delta\}$,
- i ewentualnym usunięciu z niego stanów nieosiągalnych (tzn. takich, których nie można osiągnąć po przeczytaniu żadnego słowa).

Zamiast usuwania stanów nieosiągalnych można je także po prostu pominąć podczas konstrukcji.

Zadanie 5. Narysuj automat niedeterministyczny o nie więcej niż trzech stanach **rozpoznający** język $(0+1)^*0(0+1)$, a następnie go zdeterminizuj.

Zadanie 6. Uzasadnij, że nie istnieje automat deterministyczny o mniej niż czterech stanach rozpoznający język z poprzedniego zadania.
Wskazówka: Automat o mniej niż czterech stanach nie odróżni pewnych dwóch słów w zbiorze $\{00, 01, 10, 11\}$ i można go oszukać.

Zadanie 7. Skonstruuj automat niedeterministyczny o $n+1$ stanach rozpoznający język $(0+1)^*0(0+1)^{n-1}$. Uzasadnij, że nie istnieje automat deterministyczny o mniej niż 2^n stanach rozpoznający ten język.

Zadanie 8. Determinizacja automatu nie zawsze musi być kosztowna. Naturalny automat niedeterministyczny rozpoznający język opisany wyrażeniem regularnym *if + else + endif* ma 12 stanów. Istnieje jednak równoważny automat deterministyczny o 8 stanach[1]. Narysuj oba automaty.

[1] Można go nawet znaleźć automatycznie uruchamiając po determinizacji dodatkowy algorytm minimalizacji automatu. Algorytm ten jednak w ogólnym przypadku jest dość skomplikowany i nie będziemy się nim tutaj zajmować.