O zadaniu 48

Antoni Kościelski

1 Treść zadania

Mamy odpowiedzieć na pytanie, czy język $\mathcal{L} \subseteq \{0,1\}^*$ złożony ze słów w spełniających warunek

$$| w |_0 \leqslant | w |_1 \leqslant 2 \cdot | w |_0$$

jest bezkontekstowy.

Symbol $| w |_a$ oznacza tu liczbę wystąpień litery a w słowie w. Tak więc rozważany język składa się ze słów w zawierających przynajmniej tyle znaków 1, co znaków 0, ale nie więcej niż podwojoną liczbę znaków 0.

2 Automaty skończony ze stosem

Dla ustalenia uwagi przypomnimy teraz definicję automatu ze stosem.

Taki automat jest niedeterministyczny i wykonuje dwa rodzaje ruchów: zwykłe i tzw. ε-ruchy. Składa się z głowicy czytającej-piszącej, taśmy wejściowej i stosu. Może się znajdować w jednym ze skończenie wielu stanów.

Głowica automatu potrafi zapoznać się z literą zapisaną w obserwowanej komórce taśmy wejściowej oraz ustalić, czy stos jest pusty, a także przeczytać literę znajdującą się na szczycie stosu. Nie zmienia napisu na taśmie wejściowej, potrafi jednak wykonać na stosie serię czynności polegającą na usunięciu litery znajdującej się w wierzchołku stosu, a następnie dopisaniu do stosu wskazanego słowa.

Formalnie automat ze stosem definiuje się jako siódemkę postaci

$$\langle Q, \Sigma, q_0, \Gamma, Z, \delta, F \rangle,$$

gdzie Q, Σ, Γ są skończonymi zbiorami nazywanymi odpowiednio zbiorem stanów, alfabetem wejściowym i alfabetem stosowym, q_0 i Z są wyróżnionymi elementami odpowiednio ze zbiorów Q i Γ, zwanymi stanem początkowym i symbolem dna stosu, a δ jest tzw. relacją przejścia. Zbiór $F \subseteq Q$ nazywa się zbiorem stanów akceptujących i w części zastosowań nie jest potrzebny.

Relacja przejścia δ może być uważana za podzbiór iloczynu kartezjańskiego

$$Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$$

lub funkcję typu

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}.$$

Dobrze jest też ustalić, w jaki sposób będziemy mówić o stosie. Zwykle przyjmuje się, że zawartość stosu jest (niepustym) słowem nad alfabetem Γ. Pierwszą literę takiego słowa uważamy za znajdującą się w wierzchołku stosu, kolejne litery znajdują się coraz głębiej na stosie. Przyjmujemy też, że ostatnią literą stosu jest zawsze specjalna litera oznaczająca dno stosu. Zaobserwowanie tej litery w wierzchołku stosu oznacza, że stos jest pusty. Na stosie nie może być dwóch takich liter, a w razie konieczności usunięcia jej ze stosu powinniśmy ponownie ją umieścić w odpowiednim miejscu.

Automat skończony ze stosem można utożsamiać z programem komputerowym szczególnej, niżej podanej postaci ($W \in \Sigma^*$ jest danym słowem, W_i to i-ta litera W, litery są numerowane od 1):

```
stan = q0; stos = Z; i = 1;

dopóki w δ jest piątka jednej z postaci ⟨ stan, Wi, stos1, ... ⟩
lub ⟨ stan, ε, stos1, ... ⟩

   wybierz jedną taką piątkę ⟨stan, a, stos1, q, t⟩ ∈ δ;//a∈ {Wi, ε}

   stan = q; stos = t + stos2stos3...stos|stos|;

   jeżeli a∈ Σ, to i = i + 1;

jeżeli i > |W| oraz stos = Z, to akceptuj dane, a w przeciwnym
razie - odrzuć je.
```

W powyższym programie symbol a oznacza albo odpowiednią literę słowa W, albo symbol ε, który jest czymś nie należącym do Σ. Jeżeli $a \in \Sigma$, to automat wykonuje zwykły ruch, w przeciwnym razie – ε-ruch. Zauważmy także, że w definicji relacji przejścia symbol ε ma dwa znaczenia, na ostatniej pozycji w piątce należącej do δ oznacza słowo puste.

Niektóre własności automatów skończonych możemy dowodzić tak, jak własności podanego programu, na przykład stosując niezmienniki.

3 Języki bezkontekstowe

Jest kilka sposobów definiowania języków bezkontekstowych. Często definiuje się je jako języki generowane przez gramatyki bezkontekstowe.

Mogą być też definiowane jako języki akceptowane przez automaty ze stosem. Automaty te są definiowane na wiele sposobów, definicje zwykle różnią się szczegółami. Co najważniejsze, automaty ze stosem akceptują dane albo przez przejście do stanu akceptującego, albo przez opróżnienie stosu.

Wszystkie wyżej wspomniane sposoby definiowania automatów ze stosem i akceptowania przez nie danych powinny prowadzić do równoważnych definicji bezkontekstowości.

Wyżej podany program zawiera raczej restrykcyjną definicję akceptowania przez opróżnienie stosu. Zgodnie z tą definicją automat ze stosem akceptuje dane słowo, jeżeli po przeczytaniu tego słowa i po poprawnym wykonaniu wszystkich możliwych do wykonania rozkazów okazuje się, że stos automatu jest pusty, czyli zawiera tylko symbol dna stosu.

Język akceptowany przez automat skończony składa się dokładnie z tych słów, które są akceptowane przez ten automat, albo dokładniej, dla których tak potrafimy dobierać wykonywane rozkazy, aby spowodować opróżnienie stosu.

Język jest bezkontekstowy, jeżeli jest w podanym sensie akceptowany przez pewien automat skończony.

4 Idea pierwszego rozwiązania

Wiadomo, że języki takie, jak interesujący nas język $\mathcal{L}$, bada się analizując wartości wyrażenia

$$2 \cdot \mid w \mid_0 - \mid w \mid_1.$$

Najzwyklejszy algorytm obliczający wartość tego wyrażenia może zostać zrealizowany przez automat ze stosem. Bardzo łatwo sprawdza się, czy dane słowo ma dwa razy więcej znaków 1 niż 0. Właściwie ten sam automat może sprawdzać, czy w danym słowie jest tyle samo 0, co 1: wystarczy widząc znak 1 odejmować od obliczanej wartości 2 zamiast 1. Jeżeli dla danego słowa w będziemy – widząc 1 – zmniejszać niedeterministycznie wartość $2 \cdot \mid w \mid_0 - \mid w \mid_1$ o 1 lub 2, to sprawdzimy, czy liczba $\mid w \mid_1$ znajduje się między $\mid w \mid_0$ i $2 \cdot \mid w \mid_0$.

Będziemy więc konstruować pewien automat. Automat ten na stosie będzie pamiętać pewną liczbę całkowitą. Liczbę naturalną n będzie pamiętać jako $+^n Z$, zaś ujemną liczbę $-n$ – jako $-^n Z$. Widząc w danym słowie znak 0 automat ten zwiększy o 2 liczbę pamiętaną na stosie. Nie jest to czynność banalna: jeżeli na stosie przechowywana jest liczba ujemna, to konieczne jest dwukrotne badanie wierzchołka stosu, aby zdecydować, czy można usunąć dwa znaki $-$, czy też trzeba usunąć $-$ i dodać znak $+$. Jeżeli w słowie wejściowym zostanie napotkany znak 1, to automat zmniejszy wartość pamiętaną na stosie o 1, a następnie niedeterministycznie podejmie decyzję albo o dalszym zmniejszeniu wartości stosu o 1, albo o pozostawieniu jej bez zmiany.

5 Definicja potrzebnego automatu

Automat będzie miał trzy stany: stan początkowy q_0 oraz q_1 i q_{-1}. Alfabetem wejściowym będzie oczywiście $\Sigma = \{0, 1\}$. Alfabet stosowy będzie składał się z trzech znaków: symbolu dna stosu Z oraz znaków $+$ i $-$. Nie musimy definiować zbioru stanów akceptujących (nie będą nam potrzebne). Relację przejścia δ zdefiniujemy wymieniając wszystkie jej elementy. Oto one:

$$
\begin{array}{lll}
\langle q_0, 0, Z, \ q_0, ++Z\rangle, & \langle q_0, 0, +, \ q_0, +++\rangle, & \langle q_0, 0, -, \ q_1, \varepsilon\rangle, \\
\langle q_1, \varepsilon, -, \ q_0, \varepsilon\rangle, & \langle q_1, \varepsilon, Z, \ q_0, +Z\rangle, & \\
\langle q_0, 1, Z, \ q_{-1}, -Z\rangle, & \langle q_0, 1, +, \ q_{-1}, \varepsilon\rangle, & \langle q_0, 1, -, \ q_{-1}, --\rangle, \\
\langle q_{-1}, \varepsilon, Z, \ q_0, Z\rangle, & \langle q_{-1}, \varepsilon, +, \ q_0, +\rangle, & \langle q_{-1}, \varepsilon, -, \ q_0, -\rangle, \\
\langle q_{-1}, \varepsilon, Z, \ q_0, -Z\rangle, & \langle q_{-1}, \varepsilon, +, \ q_0, \varepsilon\rangle, & \langle q_{-1}, \varepsilon, -, \ q_0, --\rangle.
\end{array}
$$

6 Własności słów akceptowanych przez automat

Najpierw zauważmy, że stwierdzenie

$$\exists n \in N \ \texttt{stos} = +^n Z \vee \texttt{stos} = -^n Z$$

jest niezmiennikiem wyżej podanego programu. Stąd wynika, że zawartość stosu możemy interpretować jako liczbę całkowitą. Liczbę tę będziemy oznaczać symbolem $v(\texttt{stos})$. Oczywiście, mamy $v(+^n Z) = n$ oraz $v(-^n Z) = -n$.

Drugim, godnym uwagi spostrzeżeniem jest to, że nasz automat zatrzymuje się dopiero po przeczytaniu całego, danego słowa i kończy pracę w stanie q_0. Nietrudno przekonać się oglądając relację przejścia, że jeżeli automat znajduje się w stanie q_0 i na taśmie wejściowej widzi jakąś literę, to ma pewien ruch do wykonania. Podobnie jest w sytuacji, gdy automat jest w stanie q_1 lub q_{-1}, ale wtedy działanie automatu nie zależy od tego, co widać na taśmie wejściowej (w tych stanach automat zawsze wykonuje ε-ruchy).

Indeksy stanów zostały tak dobrane, że pamiętają odłożoną w czasie zmianę stosu: w stanie q_1 zwiększamy liczbę pamiętaną na stosie o 1, w stanie q_{-1} – zmniejszamy ją o 1. Stany odpowiadają więc pewnym liczbom. Przyjmijmy, że $v(q_i) = i$.

Dzięki wprowadzonym oznaczeniom może zapisać ważny niezmiennik naszego automatu. Jest nim stwierdzenie

$$2 \cdot | w |_0 - 2 \cdot | w |_1 \leqslant v(\texttt{stos}) + v(\texttt{stan}) \leqslant 2 \cdot | w |_0 - | w |_1,$$

gdzie w oznacza przeczytaną część danego słowa W, a więc $w = W_1 W_2 \ldots W_{i-1}$. Wydaje się, że podana nierówność jest intuicyjnie jasna. Natomiast jest oczywiste, że podlega formalnemu sprawdzeniu tak, jak każdy inny precyzyjnie sformułowany niezmiennik pętli.

Fakt, że podana nierówność jest niezmiennikiem działania automatu implikuje, że każde słowo, które zostaje przez nasz automat zaakceptowane, należy do języka $\mathcal{L}$. Podany niezmiennik jest prawdziwy bezpośrednio po uruchomieniu automatu. Zachodzi więc także po zakończeniu pracy automatu. Wtedy jednak również mamy $\texttt{stos} = Z$, $w = W_1 W_2 \ldots W_{|W|} = W$ oraz $\texttt{stan} = q_0$. Podstawiając te ustalenia w niezmienniku otrzymujemy

$$2 \cdot | W |_0 - 2 \cdot | W |_1 \leqslant 0 \leqslant 2 \cdot | W |_0 - | W |_1.$$

Z pierwszej nierówności wynika, że $| W |_0 \leqslant | W |_1$, a z drugiej – $| W |_1 \leqslant 2 \cdot | W |_0$. Wobec tego $W \in \mathcal{L}$.

7 Akceptowanie słów języka $\mathcal{L}$

Program opisujący działanie automatu ze stosem możemy uzupełnić o różne dodatkowe polecenia służące na przykład do śledzenia pracy automatu (możemy ewentualnie sami robić notatki podczas obserwacji pracy automatu).

Rozwiązując nasze zadanie możemy automat wyposażyć w trzy dodatkowe liczniki $\texttt{z}_0$, $\texttt{z}_1$ i $\texttt{d}$, zerowane po uruchomieniu automatu, których wartości są aktualizowane tuż po wybraniu z relacji δ stosownej piątki, zgodnie z następującymi instrukcjami

jeżeli a = 0, to $\texttt{z}_0$ = $\texttt{z}_0$ + 1;

jeżeli a = 1, to $\texttt{z}_1$ = $\texttt{z}_1$ + 1;

jeżeli stan = $\texttt{q}_{-1}$ i $\texttt{stos}_1 \neq t$, to d = d + 1;

Tak więc podczas pracy automatu dodatkowo zliczamy przeczytane 0 i 1, a także sytuacje, w których automat znajduje się w stanie q_{-1} i zmienia zawartość stosu, a faktycznie dodatkowo zmniejsza pamiętaną na stosie wartość o 1.

Nietrudno spostrzec, że podczas pracy automatu, gdy znajduje się on w stanie q_0, stale jest prawdziwa następująca zależność:

$$v(\texttt{stos}) = 2 \cdot \texttt{z}_0 - \texttt{z}_1 - \texttt{d}.$$

Stąd wynika, że jeżeli w danym słowie W oznaczymy jakoś $\texttt{d}$ jedynek i spowodujemy, że automat po przeczytaniu oznaczonych jedynek będzie wybierał i wykonywał rozkaz powodujący dodatkowe zmniejszenie liczby pamiętanej na stosie, to tak działający automat, po zatrzymaniu się będzie na stosie pamiętał liczbę

$$2 \cdot | W |_0 - | W |_1 - \texttt{d}.$$

Załóżmy więc, że $W \in \mathcal{L}$. Wtedy

$$| W |_0 \leqslant | W |_1 \leqslant 2 \cdot | W |_0.$$

Z tych nierówności wynikają dwie inne:

$$0 \leqslant 2 \cdot \mid W \mid_0 - \mid W \mid_1 \leqslant \mid W \mid_1.$$

Tak więc liczba $\mathtt{d} = 2 \cdot \mid W \mid_0 - \mid W \mid_1$ jest nieujemna i nie przekracza liczby 1 w słowie W. W tym słowie możemy więc oznaczyć $\mathtt{d}$ jedynek. Dla takiego słowa, nasz automat działający w wyżej opisany sposób zakończy działanie ze stosem pamiętającym liczbę 0, a więc z pustym stosem. Jesteśmy więc w stanie skłonić nasz automat do zaakceptowania każdego słowa z języka $\mathcal{L}$.

8 Gramatyka generująca $\mathcal{L}$

Dwa ostatnie rozdziały świadczą o tym, że język $\mathcal{L}$ jest akceptowany przy pustym stosie przez zdefiniowany automat. Jest więc to język bezkontekstowy. Możemy się o tym również przekonać podając bezkontekstową gramatykę generującą $\mathcal{L}$.

Mówiąc o gramatykach, będziemy używać symbolu $\rightarrow$ zapisując produkcje, a symbol $\Rightarrow$ będzie oznaczać relację wyprowadzenia zarówno w jednym, jak i w wielu krokach. Symboli tych nie będziemy uzależniać od konkretnej gramatyki, gdyż będziemy się zajmować zaledwie kilkoma.

Dobrze znana jest gramatyka $G_=$ generująca język

$$\mathcal{L}_= = \{w \in \{0,1\}^* : \mid w \mid_1 = 2 \cdot \mid w \mid_0\}.$$

Ma ona jeden symbol nieterminalny S i następujące produkcje:

$$\begin{array}{lll} S \rightarrow SS, & S \rightarrow \varepsilon, & \\ S \rightarrow 0S1S1, & S \rightarrow 1S1S0, & S \rightarrow 1S0S1. \end{array}$$

Bez trudu dowodzi się, że jeżeli słowo $w \in \{0,1,S\}^*$ daje się wyprowadzić z S w tej gramatyce, to $\mid w \mid_1 = 2 \cdot \mid w \mid_0$. Trudniej dowieść

Lemat 8.1 *Jeżeli $w \in \mathcal{L}_=$, to $S \Rightarrow w$ (w daje się wyprowadzić z S w gramatyce $G_=$).*

W dowodzie powyższego lematu zwykle rozważa się funkcję $nj : \{0,1\}^* \rightarrow N$ (ewentualnie $nj : \{0,1,S\}^* \rightarrow N$), którą będziemy nazywać niedoborem jedynek, określoną wzorem

$$nj(v) = 2 \cdot \mid v \mid_0 - \mid v \mid_1$$

(czyli liczba jedynek, które trzeba dopisać do słowa $v \in \mathcal{L}$, aby uzyskać maksymalną liczbę jedynek pozwalającą na należenie do $\mathcal{L}$). Analizując zmiany wartości tej funkcji na prefiksach słowa w, pokazuje się, że słowa $w \in \mathcal{L}_= \setminus \mathcal{L}_=^2$ mają jedną z trzech postaci: $0w'1w''1$, $1w'0w''1$ lub $1w'1w''0$ dla pewnych słów $w', w'' \in \mathcal{L}_=$.

Uzasadnienie poprzedniego lematu można rozszerzyć do dowodu trochę mocniejszego, następującego:

Lemat 8.2 *Każde słowo $w \in \{0,1,S\}^*$ spełniające warunek $\mid w \mid_1 = 2 \cdot \mid w \mid_0$ daje się wyprowadzić w gramatyce $G_=$ z symbolu S.*

Zauważmy jeszcze następującą własność niedoboru jedynek:

Lemat 8.3 *Jeżeli słowo w powstaje ze słowa v przez dopisanie gdzieś w nim dwóch znaków: 0 i 1, to*

$$nj(w) = nj(v) + 1.$$

Dowód. Ponieważ $| w |_0 = | v |_0 + 1$ oraz $| w |_1 = | v |_1 + 1$, więc

$$nj(w) = 2 \cdot | w |_0 - | w |_1 = 2 \cdot | v |_0 + 2 - | v |_1 - 1 = nj(v) + 1. \ \square$$

Tak więc dopisując do słowa znaki 0 i 1 zwiększamy niedobór jedynek o 1, a usuwając 0 i 1 – zmniejszamy niedobór o 1.

Teraz już łatwo podać gramatykę G genarującą język $\mathcal{L}$: wystarczy do gramatyki $G_=$ dopisać dwie produkcje:

$$S \to 0S1 \ \text{ oraz } \ S \to 1S0.$$

Jest oczywiste, że wszelkie słowa wyprowadzane w tej gramatyce z symbolu S spełniają warunek z definicji $\mathcal{L}$. Również łatwo z lematu 2 wyprowadza się

Lemat 8.4 *Każde słowo $w \in \{0, 1, S\}^*$ spełniające nierówności*

$$| w |_0 \leqslant | w |_1 \leqslant 2 \cdot | w |_0$$

daje się wyprowadzić z w gramatyce G z symbolu S.

Dowód. Niedobór jedynek słów, o których mowa w lemacie, jest liczbą naturalną. Pozwala to przeprowadzić dowód lematu przez indukcję ze względu na niedobór jedynek.

Jeżeli niedobór jedynek słowa w jest równy 0, to na mocy lematu 2 daje się wyprowadzić z S w gramatyce $G_=$. Tym bardziej daje się wyprowadzić w gramatyce G.

Przypuśćmy, że słowo w ma dodatni niedobór jedynek. Wtedy spełnia ono nierówności

$$| w |_0 \leqslant | w |_1 < 2 \cdot | w |_0,$$

występuje w nim przynajmniej jeden znak 0, a także przynajmniej jeden znak 1. Gdzieś w tym słowie występują więc znaki 0 i 1 obok siebie, rozdzielone najwyżej znakami S, w tej lub w odwrotnej kolejności. Przypuśćmy więc, że

$$w = w'0S^k1w''.$$

Nietrudno zauważyć, że słowo $w'Sw''$ spełnia założenia lematu i w porównaniu z w ma o 1 mniejszy niedobór jedynek. Na mocy założenia indukcyjnego daje się ono wyprowadzić z S. Mając to wyprowadzenie łatwo wyprowadzić z S słowo w

$$S \Rightarrow w'Sw'' \Rightarrow w' \, 0S1 \, w'' \Rightarrow w' \, 0S^k1 \, w'' = w. \ \square$$

Z dowiedzionego lematu i poprzedzającej go uwagi otrzymujemy, że gramatyka G generuje język $\mathcal{L}$.