	1	2	3	4	Σ
xmarci10					

Úloha 19.12.2019

(Termín odovzdania 19.12.2019)

Úloha 1

Uvažujte jazyk $L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$, kde $\#_x(w)$ značí počet výskytov symbolu x v reťazci w. Dokážte, že jazyk L je bezkontextový. Postupujte nasledovne:

(a) Najprv navrhnite gramatiku G, ktorá bude mať za cieľ jazyk L generovať.

(b) Potom pomocou indukcie k dĺžke slova $w \in L$ dokážte, že $L = L(G)$.

(15 bodov)

Riešenie

(a) $G = \{N, \Sigma, P, S\} = \{\{S\}, \{\varepsilon, a, b\}, \{S \to aSb, S \to bSa, S \to SS, S \to \varepsilon\}, S\}$

(b) Aby sme dokázali, že $L = L(G)$, musíme dokázať, že $\boldsymbol{L(G) \subseteq L \wedge L \subseteq L(G)}$. Na dokázanie vyššie uvedeného použijeme dôkaz indukciou vzhľadom k dĺžke slova $\boldsymbol{w}$ kde $|w| = \boldsymbol{i}$.

(1.) Dokazujeme $\boldsymbol{L(G) \subseteq L}$:

- **Bázový prípad:** $i = 0$, slovo dĺžky 0 možno vygenerovať z gramatiky:
  I. priamo použitím pravidla $S \to \varepsilon$: $S \underset{G}{\Rightarrow} \varepsilon$
  II. alebo nepriamo: $S \underset{G}{\Rightarrow} SS \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$
  Keďže $\#_a(\varepsilon) - \#_b(\varepsilon) = 0$ slovo $\varepsilon \in L$, a teda pre $i = 0$ **platí**, že $L(G) \subseteq L$
- **Indukčný predpoklad:** Predpokladajme, že $L(G) \subseteq L$ platí pre všetky slová dĺžky i, takže platí:
  $$\forall w' \Big(S \underset{G}{\overset{*}{\Rightarrow}} w' \wedge |w'| = i \implies w' \in L \Big)$$
- **Indukčný krok:** Ukážeme, že implikácia platí aj pre slová dĺžky $i + 2$ (daný jazyk obsahuje iba slová s párnym počtom znakov), tj. že platí:
  $$\forall w \Big(S \underset{G}{\overset{*}{\Rightarrow}} w \wedge |w| = i + 2 \implies w \in L \Big)$$
  Pri generovaní slova w môže nastať niekoľko variánt, vzhľadom k voľbe pravidla:

  I. $S \underset{G}{\Rightarrow} aSb \underset{G}{\overset{*}{\Rightarrow}} aw'b = w \qquad |w'| = i \quad |w| = i + 2$

  Podľa indukčného predpokladu platí, že $w' \in L$, čiže $\#_a(w') - \#_b(w') = 0$. Potom je zrejmé, že pridaním jedného a a jedného b ($S \to aSb$) k slovu w' ostane zachovaný rovnaký počet symbolov a a b aj v slove w a teda $\boldsymbol{w \in L}$.

II. $S \underset{G}{\Rightarrow} bSa \overset{*}{\underset{G}{\Rightarrow}} bw'a = w \qquad |w'| = i \quad |w| = i + 2$

Znovu podľa indukčného predpokladu platí, že $w' \in L$. Tentokrát, však na deriváciu použijeme pravidlo $S \to bSa$, ktoré ale taktiež neporuší podmienku príslušnosti reťazca w do jazyka L. A teda $\boldsymbol{w \in L}$.

III. $S \underset{G}{\Rightarrow} SS \overset{*}{\underset{G}{\Rightarrow}} w_1w_2 = w \quad w_1 \in \{\varepsilon, aw_1'b, bw_1'a\} \quad |w_1'| = i \quad |w_1| = i + 2$

$\qquad\qquad\qquad\qquad\qquad w_2 \in \{\varepsilon, aw_2'b, bw_2'a\} \quad |w_2'| = i \quad |w_2| = i + 2$

V prípade, že pri derivácii použijeme pravidlo $S \to SS$, môžme výsledný reťazec w rozdeliť na dve časti: $w = w_1w_2$, kde každá z týchto častí vznikla práve z jedného neterminálu S. To znamená, že reťazce w_1 a w_2 vznikli jednou z derivácií popísaných v predchádzajúcich bodoch. Teda je zrejmé, že $w_1, w_2 \in L$ a platí, že $\#_a(w_1) = \#_b(w_1) \wedge \#_a(w_2) = \#_b(w_2)$. A keďže reťazec w vznikne konkatenáciou reťazcov w_1 a w_2 neporuší tým podmienku rovnakého počtu symbolov a a b a tým pádom $\boldsymbol{w \in L}$.

Ukázali sme, že pre všetky varianty kedy z našej gramatiky vygenerujeme slovo w dĺžky $i + 2$ (kde i je dĺžka slovo patriaceho do jazyka L), slovo w bude opäť patriť do jazyka L, čím sme dokázali, že indukčný krok **platí**.

(2.) Dokazujeme $\boldsymbol{L \subseteq L(G)}$:

- **Bázový prípad:** i = 0, platí $\#_a(\varepsilon) = \#_b(\varepsilon)$ a teda $\varepsilon \in L$. Vďaka existencii pravidla $S \to \varepsilon$ platí tiež $\varepsilon \in L(G)$ a teda pre i=0 **platí**, že $L \subseteq L(G)$.
- **Indukčný predpoklad:** Predpokladajme, že $L \subseteq L(G)$ platí pre všetky slová dĺžky i, takže platí:

$$\forall w' \Big(|w'| \leq i - 2 \wedge w' \in L \implies w' \in L(G) \Big)$$

- **Indukčný krok:** Pre w také, že $|w| = i \wedge w \in L$ ukážeme, že $w \in L(G)$. Slovo w patriace do jazyka L môže mať nasledujúce tvary:

I. $w = \boldsymbol{aw'b} \quad |w'| = i \quad |w| = i + 2$

Je zrejmé, že podľa indukčného predpokladu existuje derivácia $S \overset{*}{\underset{G}{\Rightarrow}} w'$. Slovo w potom získame použitím pravidla $S \to aSb$ nasledovne:

$$S \underset{G}{\Rightarrow} aSb \overset{*}{\underset{G}{\Rightarrow}} aw'b = w$$

Tým sme ukázali, že slovo w možno generovať gramatikou G a teda $\boldsymbol{w \in L(G)}$.

II. $w = \boldsymbol{bw'a} \quad |w'| = i \quad |w| = i + 2$

Analogicky k I. podľa indukčného predpokladu existuje derivácia $S \overset{*}{\underset{G}{\Rightarrow}} w'$. Tentokrát však slovo w získame pomocou pravidla $S \to bSa$:

$$S \underset{G}{\Rightarrow} bSa \overset{*}{\underset{G}{\Rightarrow}} bw'a = w$$

Opäť slovo w možno generovať gramatikou G a teda $\boldsymbol{w \in L(G)}$.

III. $w = \boldsymbol{axa} \quad x \in \{a,b\}^* \wedge \#_a(x) = \#_b(x) - 2$.

<u>Veta 1.2.1:</u>

Slovo w môžme vždy rozdeliť na dve časti tak, že každá z týchto častí bude mať rovnaký počet a aj b:

$$w = w'w'' \quad |w'|,|w''| < i \quad |w| = i \quad w', w'' \in L$$

<u>Dôkaz 1.2.1</u>:

Zavedieme funkciu: $\alpha_n = \#_a w_1 \dots w_n - \#_b w_1 \dots w_n$. Je zrejmé, že pre túto funkciu bude platiť, že $\alpha_0 = \alpha_i = 0$. Ďalej je zrejmé, že $\alpha_1 = 1$ a $\alpha_{i-1} = -1$. Z toho vidíme, že medzi α_1 a α_{i-1} musí existovať nejaké $\alpha_j = 0$. Potom môžeme reťazec w rozdeliť na $w = w'w''$ kde $w' = w_1 ... w_j$ a $w'' = w_{j+1} ... w_i$. Nový reťazec $w' \in L$. A keďže reťazec $w \in L$ a musí mať rovnaký počet a a b tak z toho plynie, že aj reťazec $w'' \in L$.

Máme $w = w'w''$ kde $w' \in L \wedge w'' \in L$. Podľa indukčného predpokladu existujú derivácie $S \underset{G}{\overset{*}{\Rightarrow}} w'$ a $S \underset{G}{\overset{*}{\Rightarrow}} w''$. Na odvodenie w potom použijeme deriváciu $S \underset{G}{\Rightarrow} SS \underset{G}{\overset{*}{\Rightarrow}} w'S \underset{G}{\overset{*}{\Rightarrow}} w'w'' = w$. Takže $\boldsymbol{w \in L(G)}$.

IV. $w = \boldsymbol{bxb}$ analogicky k $w = axa$

Ukázali sme, že $\forall w \in L : |w| = i$ platí $w \in L(G)$. Indukčný krok **platí**.

Pomocou indukcie k dĺžke slova $w \in L$ sme dokázali, že $\boldsymbol{L = L(G)}$.

Úloha 2

Uvažujte *doprava čítaný jazyk* TS M, značený ako $L^P(M)$, ktorý je definovaný ako množina reťazcov, ktoré M príjme v behu, pri ktorom nikdy nepohne hlavou *doľava* a nikdy neprepíše žiadny symbol na páske za iný. Dokážte, či je problém prázdnosti doprava čítaného jazyka TS M, tj. či $L^P(M) = \emptyset$, je rozhodnuteľný:

- ak *áno*, napíšte algoritmus v pseudokóde, ktorý daný problém bude rozhodovať;
- ak *nie*, dokážte nerozhodnuteľnosť redukciou z jazyka *HP*.

(15 bodov)

Riešenie

(pozn.: Hlavná myšlienka zhrnutá za algoritmom.)

Algoritmus 1 Rozhodne či $L^P(M) = \emptyset$

Vstup: TS $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$
Výstup: $L^P(M) = \emptyset \implies$ ***True***, $L^P(M) \neq \emptyset \implies$ ***False***

/* Funkcia, ktorá z pôvodného TS odstráni prechody doľava a taktiež všetky prechody ktoré prepisujú symbol na iný symbol */
function GETRIGHTMOVINGTS$(Q, \Sigma, \Gamma, \delta', q_0, q_F)$
  $\delta' = \delta$
  for $\forall q, q' \in Q$ **do**
    for $\forall a, b \in \Gamma$ **do**
      if $\delta(q, a) = (q', L) \vee \delta(q, a) = (q', b)$ **then**
        $\delta' = \delta' \setminus (q, a)$
      end if
    end for
  end for
  return $(Q, \Sigma, \Gamma, \delta', q_0, q_F)$
end function

/* Funkcia, spočíta "Δ" uzáver pre stav q_i. To znamená množinu všetkých stavov kde sa dostanem z q_i pomocou prechodu Δ/Δ */
function ΔCLOSURE(δ, q_i)
  $(x_1, x_2) = \delta(q_i, \Delta)$
  $closure = \{q_i\}$
  while $x_2 == \Delta$ **do**
    $closure.add(x_1)$
    $(x_1, x_2) = \delta(x_1, x_2)$
  end while
  return $closure$
end function

```
/* Funkcia, spočíta "$\Delta_R$" uzáver pre stav $q_i$. To jest množinu všetkých
stavov kde sa dostanem z $q_i$ pomocou prechodu $\Delta/\Delta$ alebo $\Delta/R$ */
function $\Delta_R$Closure$(\delta, q_i)$
    $(x_1, x_2) = \delta(q_i, \Delta)$
    $closure = \{q_i\}$
    while $x_2 == \Delta \lor x_2 == R$ do
        $closure.add(x_1)$
        $(x_1, x_2) = \delta(x_1, x_2)$
    end while
    return $closure$
end function

/* Funkcia, prevedie daný Right Moving DTS na RKA */
function TMtoRKA$(Q, \Sigma, \Gamma, \delta, q_0, q_F)$
    $RKA\ K = (Q, \Sigma \cup \{\varepsilon\}, \delta_A, q_0, \{g_F\})$
    for each $q_1, q_2 \in Q$ do
        for each $a \in \Sigma$ do
/* pohyb doprava prečítaním symbolu a */
            if $\delta(q_1, a) = (q_2, R)$ then
                $\delta_A(q_1, a) = \delta_A(q_1, a) \cup q_2$
/* nahradenie prechodu $a/a$, $\varepsilon$ prechodom + ošetrenie HALT */
            else if $(\delta(q_1, a) = (q_2, a)) \land (\delta(q_2, a) \neq \emptyset \lor q_2 = q_F)$ then
                $\delta_A(q_1, \varepsilon) = \delta_A(q_1, \varepsilon) \cup q_2$
/* ošetrenie situácie prechodov $\Delta/\Delta$ na začiatku TS */
            else if $(\delta(q_1, \Delta) = (q_2, \Delta)) \land (q_2 \in \Delta$Closure$(\delta, q_0))$ then
                $\delta_A(q_1, \varepsilon) = \delta_A(q_1, \varepsilon) \cup q_2$
/* ošetrenie situácie prechodov $\Delta/R$ na začiatku TS */
            else if $(\delta(q_1, \Delta) = (q_2, R)) \land (q_1 \in \Delta$Closure$(\delta, q_0))$ then
                $\delta_A(q_1, \varepsilon) = \delta_A(q_1, \varepsilon) \cup q_2$
/* ošetrenie situácie prechodov $\Delta/\Delta|R$ na konci TS */
            else if $(\delta(q_1, \Delta) = (q_2, R)) \land (q_F \in \Delta_R$Closure$(\delta, q_1))$ then
                $\delta_A(q_1, \varepsilon) = \delta_A(q_1, \varepsilon) \cup q_F$
            else if $(\delta(q_1, \Delta) = (q_2, \Delta)) \land (q_F \in \Delta_R$Closure$(\delta, q_1))$ then
                $\delta_A(q_1, \varepsilon) = \delta_A(q_1, \varepsilon) \cup q_F$
            end if
        end for
    end for
    return K
end function

procedure Main$(M = (Q, \Sigma, \Gamma, \delta, q_0, q_F))$
    $TM\ M^* =$ getRightMovingTS$(M)$
    $RKA\ K =$ TMtoRKA$(M^*)$
    $DKA\ D =$ RKAtoDKA$(K)$                    ▷ TIN skriptá Algoritmus 3.6
    if $\exists q \in Q_D : (q \in F_D \land q$ je dostupný z $q_0^D)$ then
        return False
    else
        return True
    end if
end procedure
```

Myšlienka

Obecne je problém (ne)prázdnosti u TS nerozhodnuteľný. Avšak o *read-only right moving TS* (náš prípad) vieme povedať, že jazyk prijímaný takýmto TS patrí do triedy regulárnych jazykov. Tým pádom hlavná myšlienka algoritmu spočíva v prevode TS M na deterministický KA, kde už vieme triviálne povedať, či je jeho jazyk (ne)prázdny (overíme dostupnosť koncového stavu z toho počiatočného).

Algoritmus začína vo funkcii Main, ktorá prijíma na svojom vstupe deterministický TS M. Následne prvým krokom je prevod TS M na TS M^*, ktorý už neobsahuje prechody, v ktorých by hýbal hlavou doľava alebo by prepisoval nejaký symbol na vstupnej páske za iný symbol. Následne na takto upravený TS zavoláme funkciu, ktorá tento TS prevedie na RKA. Posledným krokom je potom prevod RKA na DKA a následné rozhodnutie o (ne)prázdnosti.

Úloha 3

Uvažujte jazyk $L_{42} = \{\langle M \rangle \mid \text{TS } M$ zastaví na niektorom vstupe tak, že páska bude obsahovať práve 42 neblankových symbolov$\}$. Dokážte pomocou redukcie, že L_{42} je nerozhodnuteľný. Uveďte ideu dôkazu čiastočnej rozhodnuteľnosti L_{42}.

(10 bodov)

Riešenie

Dôkaz urobíme technikou redukcie z problému $P_1 : P_1 \leq P_2$, kde problém P_1 odpovedá problému HP a P_2 je zadaný problém značený ako L_{42}. Dostávame teda zápis redukcie:

$$HP \leq L_{42}$$

Problém zastavenia (HP) je nerozhodnuteľný — ak teda bude zvolená redukcia urobená správne, tak v dôsledku je aj problém L_{42} nerozhodnuteľný.

Postup redukcie:

1. Jazyky charakterizujúce dané problémy:
   - $HP = \{\langle M \rangle \# \langle w \rangle \mid M$je TS taký, že na w zastaví$\}$
   - $L_{42} = \{\langle M \rangle \mid$ TS M zastaví na niektorom vstupe tak, že páska bude obsahovať práve 42 neblankových symbolov$\}$
2. Zostavíme redukciu:

   $$\sigma : \{0, 1, \#\}^* \to \{0, 1\}^* \text{ z jazyka } HP \text{ na } L_{42}$$

   σ priradí každému vstupu $x \in \{0, 1, \#\}^*$ reťazec $\langle M_x \rangle$,kde M_x je TS, ktorý na vstupe $y \in \{0, 1\}^*$ pracuje nasledovne:
   1. M_x zmaže svoj vstup y.
   2. Na vstupnú pásku zapíše reťazec x.
   3. M_x posúdi, či x má štruktúru $x_1 \# x_2$, kde x_1 je kód TS a x_2 je kód vstupu. Ak nie, **vymaže vstupnú pásku** a potom odmietne.
   4. M_x odsimuluje na reťazci s kódom x_2 beh TS s kódom x_1:
      - Ak x_1 na x_2 zastaví: M_x vymaže vstupnú pásku, **zapíše na ňu 42 neblankových symbolov** a príjme.
      - Inak cyklí.
3. Implementácia M_σ:
   σ možno jednoducho implementovať úplným TS M_σ, ktorý pre vstup x vyprodukuje kód TS M_x, ktorý sa skladá zo štyroch komponent, ktoré odpovedajú vyššie uvedeným krokom:
   1. Komponenta, ktorá zmaže obsah vstupnej pásky.
   2. M_σ vypíše kód TS, ktorý zapíše na vstup reťazec $x = a_1, a_2, \ldots, a_n$. To je možné ľahko realizovať pomocou TS $Ra_1Ra_2Ra_3$.
   3. M_σ vypíše kód TS, ktorý na vstupe overí, či sa jedná o platnú inštanciu HP a ak nie odmietne.
   4. M_σ vypíše kód TS, ktorý spustí UTS na TS s kódom x_1 a vstupe s kódom x_2.

4. Možné jazyky TS M_x:
   - $L(M_x) = \emptyset \iff x$ nieje správne sformovaná inštancia HP alebo TS s kódom x_1 cyklí na vstupe s kódom x_2.
   - $L(M_x) = \Sigma^* \iff x$ je správne sformovaná inštancia HP, a TS s kódom x_1 zastavil vstupe s kódom x_2.
5. Na záver ukážeme že σ zachováva členstvo:
   $\forall x \in \{0, 1, \#\}^* : \ \sigma(x) = \langle M_x \rangle \in L_{42} \iff L(M_x) = \Sigma^* \iff x = x_1\#x_2,$ kde x_1 je kód TS, ktorý príjme vstup s kódom $x_2 \iff x \in HP$

Čiastočná rozhodnuteľnosť (idea)

K čiastočnému rozhodnutiu uvedeného problému L_{42} môžeme zostrojiť TS M', ktorý na svojej páske simuluje beh vstupného TS M pre jednotlivé možné vstupné reťazce.

M' nemôže iba systematicky vygenerovať jednotlivé vstupné reťazce v lexikografickom usporiadaní a na každom spustiť neobmedzenú simuláciu M. Pri zacyklení M by sa celý výpočet zacyklil, bez garancie nájdenia reťazca, ktorý M príjme (ak existuje).

Namiesto toho M' na svojej páske postupne rozbieha viac a viac simulácií TS M pre jednotlivé možné vstupné reťazce. V každej z týchto simulácií si potom pamätá naviac stav riadenia M pri zpracovaní daného vstupu. Jednotlivé rozbehnuté simulácie má vhodným spôsobom oddelené (a taktiež im môže zväčšovať potrebný priestor).

Simulácia prebieha tak, že M' vždy vykoná jeden krok na každej rozbehnutej simulácii. Ak jedna z nich vedie k prijatiu, M' príjme. Inak rozbehne ďalšiu simuláciu pre ďalší vstupný reťazec a tento postup opakuje.

Je zrejmé, že M' príjme, ak $L(M) \neq \emptyset$. Inak neskončí.