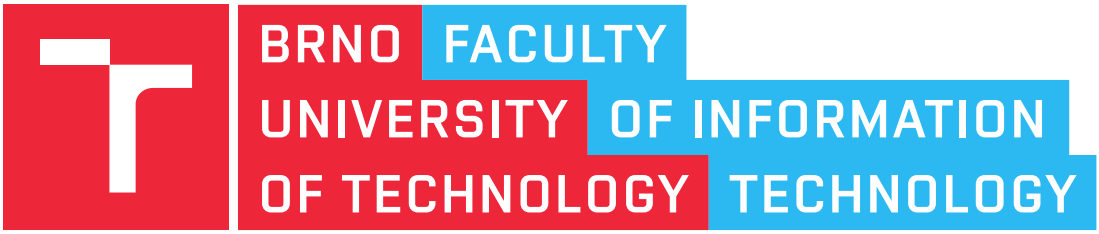

Teoretická informatika

Úkol 2

18. prosince 2019 Vladimír Dušek, xdusek27

Teoretická informatika (TIN) – 2019/2020
Úkol 2
(max. zisk 5 bodů – 10 bodů níže odpovídá 1 bodu v hodnocení předmětu)

1. Uvažujte jazyk $L = \{w \in \{a,b\}^* \mid \#_a(w) = \#_b(w)\}$, kde $\#_x(w)$ značí počet výskytů symbolu x v řetězci w. Dokažte, že jazyk L je bezkontextový. Postupujte následovně:
   (a) Nejdřív navrhněte gramatiku G, která bude mít za cíl jazyk L generovat.
   (b) Poté pomocí indukce k délce slova $w \in L$ dokažte, že $L = L(G)$.

   15 bodů

2. Uvažujte *doprava čtený jazyk* TS M, značený jako $L^P(M)$, který je definován jako množina řetězců, které M přijme v běhu, při kterém nikdy nepohne hlavou *doleva* a nikdy nepřepíše žádný symbol na pásce za jiný. Dokažte, zda je problém prázdnosti doprava čteného jazyka TS M, tj. zda $L^P(M) = \emptyset$, rozhodnutelný:
   - pokud *ano*, napište algoritmus v pseudokódu, který daný problém bude rozhodovat;
   - pokud *ne*, dokažte nerozhodnutelnost redukcí z jazyka HP.

   10 bodů

3. Uvažujte jazyk $L_{42} = \{\langle M \rangle \mid$ TS M zastaví na některém vstupu tak, že páska bude obsahovat právě 42 neblankových symbolů $\}$. Dokažte pomocí redukce, že L_{42} je nerozhodnutelný. Uveďte ideu důkazu částečné rozhodnutelnosti L_{42}.

   10 bodů

4. Uvažujte programovací jazyk **`Karel@TIN`** s následující gramatikou:

$$\begin{aligned}
\langle \textit{stmt} \rangle &::= \texttt{drop-screw} \mid \texttt{lift-screw} \mid \texttt{step} \mid \texttt{turn-left} \mid \textbf{\texttt{return}}\ b \mid \\
&\qquad \textbf{\texttt{if}}\ \texttt{empty:}\ \textbf{\texttt{goto}}\ n \mid \textbf{\texttt{if}}\ \texttt{not empty:}\ \textbf{\texttt{goto}}\ n \\
\langle \textit{stmt-list} \rangle &::= \langle \textit{stmt} \rangle \texttt{;} \langle \textit{stmt-list} \rangle \mid \langle \textit{stmt} \rangle \\
\langle \textit{program} \rangle &::= \langle \textit{stmt-list} \rangle \texttt{;}\ \textbf{\texttt{return}}\ b\texttt{;}
\end{aligned}$$

   kde $n \in \mathbb{N}$, $b \in \{0,1\}$ a počáteční neterminál je $\langle \textit{program} \rangle$ (uvažujeme, že $0 \in \mathbb{N}$). Sémantika je následující:
   - Program v **`Karel@TIN`** je interpretován robotem Karlem na dvojrozměrné mřížce ve všech směrech nekonečné. Políčka mřížky jsou indexována celými čísly, tj. množina indexů políček mřížky je $\mathbb{Z} \times \mathbb{Z}$.
   - Robot Karel se při provádění programu pohybuje po políčkách mřížky. Na zádech má batoh, ve kterém má nekonečně mnoho šroubků, které může pokládat na políčka mřížky (maximální počet šroubků na jednom políčku není omezen), případně je z políček zvedat a dávat do batohu.
   - *Konfigurace prostředí* je trojice $C = (pos, dir, grid)$ kde $pos \in \mathbb{Z} \times \mathbb{Z}$ je pozice Karla v mřížce, $dir \in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ označuje, kterým směrem je Karel natočen a $grid : \mathbb{Z} \times \mathbb{Z} \hookrightarrow \mathbb{N}$ je konečná parciální funkce, která některým políčkám mřížky přiřazuje hodnotu toho, kolik je na nich položených šroubků (0 a nedefinovaná hodnota znamenají, že na políčku s daným indexem není žádný šroubek).
   - Příkazem `drop-screw` položí robot Karel na políčko, na kterém se zrovna nachází, jeden šroubek z batohu.
   - Příkazem `lift-screw` zvedne robot Karel z políčka, na kterém se zrovna nachází, jeden šroubek, pokud tam nějaký je, a dá si ho do batohu; pokud na políčku žádný šroubek není, robot abnormálně terminuje (exploduje mu hlava).
   - Příkazem `step` udělá robot Karel jeden krok ve směru, kterým je otočen. Formálně, `step` změní konfiguraci prostředí z $C = (pos, dir, grid)$ na $C' = (pos + \vec{v}, dir, grid)$, kde $\vec{v} = (1,0)$, pokud $dir = \rightarrow$; $\vec{v} = (-1,0)$, pokud $dir = \leftarrow$; $\vec{v} = (0,1)$, pokud $dir = \uparrow$; a $\vec{v} = (0,-1)$, pokud $dir = \downarrow$.
   - Příkazem `turn-left` se robot otočí doleva, tedy změní svou orientaci z $\rightarrow$ na $\uparrow$ apod.
   - Příkazy **`if`** `empty:` **`goto`** n a **`if`** `not empty:` **`goto`** n provedou podmíněný skok na n-tý příkaz (příkazy jsou číslovány od 0 a odděleny znakem středníku) pokud na políčku, na kterém právě stojí robot, není žádný šroubek (`empty`), resp. je alespoň jeden šroubek (`not empty`). Pokud je n mimo rozsah programu, Karel abnormálně terminuje (upadnou mu nohy).
   - Příkaz **`return`** b ukončí program s návratovou hodnotou b.

Příklad 1: Program, pomocí kterého bude robot vytvářet rovnou cestu ze šroubků (na každém políčku cesty přesně jeden) ve směru počátečního otočení.

```
0 if empty: goto 3;
1 lift-screw;
2 if not empty: goto 1;
3 put-screw;
4 step;
5 if not empty: goto 0;
6 if empty: goto 0;
7 return 0;
```

Příklad 2: Program, pomocí kterého bude robot procházet po cestě sestavené ze šroubků, dokud to jde (pak vrací 1). Pokud robot pod sebou nemá na začátku žádný šroubek, okamžitě vrací 0.

```
0 if empty: goto 22;
1 step;
2 if not empty: goto 1;
3 turn-left;
4 step;
5 turn-left;
6 step;
7 turn-left;
8 if empty: goto 12
9 turn-left;
10 turn-left;
11 if not empty: goto 1;
12 step;
13 step;
14 if not empty: goto 1;
15 turn-left;
16 turn-left;
17 step;
18 turn-left;
19 turn-left;
20 turn-left;
21 return 1;
22 return 0;
```

Obrázek 1: Příklady programů v jazyce **Karel@TIN**

Obrázek 1 obsahuje příklady programů v jazyce **Karel@TIN**.

Dokažte, že programovací jazyk **Karel@TIN** je Turingovsky úplný, tj., dokažte, že

(a) pro každý TS M nad abecedou $\{0,1\}$ a řetězec $w \in \{0,1\}^*$ lze sestrojit program P_M v jazyce **Karel@TIN** a zvolit počáteční konfiguraci prostředí C_M tak, že P_M skončí s návratovou hodnotou 1 právě tehdy, když $w \in L(M)$;

(b) pro každý program P v jazyce **Karel@TIN** a počáteční konfiguraci C lze sestrojit TS M_P a řetězec $w \in \{0,1\}^*$ tak, že $w \in L(M_P)$ právě tehdy, když robot Karel po interpretaci programu P z počáteční konfigurace C skončí s návratovou hodnotou 1.

15 bodů

Příklad 1

Máme jazyk $L = \{w \in \{a,b\}^* \mid \#_a(w) = \#_b(w)\}$. Navrhněme gramatiku G, která jazyk L generuje:

$$G = (\{S\}, \{a,b\}, \{S \to \varepsilon \mid SaSb \mid aSbS \mid SbSa \mid bSaS\}, S)$$

Nyní indukcí k délce slova dokažme, že gramatika G skutečně generuje jazyk L. Tedy, že $L = L(G)$. Postupně ukážeme, že $L(G) \subseteq L$ a $L \subseteq L(G)$, z toho poté přímo vyplývá, že $L = L(G)$. Nechť i je délka slova.

Tvrzení 1: jestliže řetězec w je generován gramatikou G, pak $w \in L$. Tedy, že $L(G) \subseteq L$.

Základní případy:

- $i = 0$. Jediný řetězec délky 0 je ε (prázdný řetězec). Jelikož derivace $S \Rightarrow \varepsilon$ je součástí gramatiky G, tak $\varepsilon \in L(G)$. Zároveň ε obsahuje stejný počet symbolů a a b (0), proto $\varepsilon \in L$. Z toho vyplývá, že pro $i = 0$ tvrzení $L(G) \subseteq L$ platí.
- $i = 2$. Možné řetězce délky 2 nad $\{a,b\}^*$ jsou aa, ab, ba, bb. Jelikož derivace $S \Rightarrow SaSb \Rightarrow aSb \Rightarrow ab$ a $S \Rightarrow SbSa \Rightarrow bSa \Rightarrow ba$ jsou součástí gramatiky G, tak $ab, ba \in L(G)$. Obdobné derivace, které by generovaly řetězce aa, bb nejsou součástí gramatiky G, proto $aa, bb \notin L(G)$. Zároveň řetězce ab, ba obsahují stejný počet symbolů a a b (1), proto $ab, ba \in L$. Řetězce aa, bb neobsahují stejný počet symbolů a a b, proto $aa, bb \notin L$. Z toho vyplývá, že pro $i = 2$ tvrzení $L(G) \subseteq L$ platí.

Indukční hypotéza:

- Předpokládejme, že pro všechny řetězce w takové, že $|w| \leq i$, platí $w \in L(G) \Rightarrow w \in L$.

Indukční krok:

- Ukažme platnost tvrzení pro délku slova $i + 2$. Je zřejmé, že řetězce délky $i + 1$ nejsou součástí jazyka $L(G)$, jelikož všechna pravidla generují sudý počet terminálních symbolů.
- Uvažujme řetězec $w_j \in L(G)$ délky $i + 2$. Mohou nastat následující situace:
  - $S \Rightarrow aSbS \Rightarrow^* au_1bv_1 = w_1$, kde $|u_1| + |v_1| = i$
  - $S \Rightarrow SaSb \Rightarrow^* u_2av_2b = w_2$, kde $|u_2| + |v_2| = i$
  - $S \Rightarrow bSaS \Rightarrow^* bu_3av_3 = w_3$, kde $|u_3| + |v_3| = i$
  - $S \Rightarrow SbSa \Rightarrow^* u_4bv_4a = w_4$, kde $|u_4| + |v_4| = i$
- Na základě indukční hypotézy, kdy předpokládáme, že z počátečního neterminálu S dokážeme po konečném počtu kroků vygenerovat slovo délky maximálně i, můžeme uvážit podmínky $|u_j| + |v_j| = i$ pro $j \in \{1,2,3,4\}$.
- Jelikož $u_j, v_j \in L(G)$, tak dle indukční hypotézy $u_j, v_j \in L$.
- Tvrzení $L(G) \subseteq L$ platí i pro $i + 2$.

Tvrzení 2: jestliže řetězec $w \in L$, pak w je generován gramatikou G. Tedy, že $L \subseteq L(G)$.

Základní případy:

- $i = 0$. Jediný řetězec délky 0 je ε (prázdný řetězec). Řetězec ε obsahuje stejný počet symbolů a a b (0), proto $\varepsilon \in L$. Zároveň derivace $S \Rightarrow \varepsilon$ je součástí gramatiky G, tedy $\varepsilon \in L(G)$. Z toho vyplývá, že pro $i = 0$ tvrzení $L \subseteq L(G)$ platí.
- $i = 2$. Možné řetězce délky 2 nad $\{a, b\}^*$ jsou aa, ab, ba, bb. Řetězce ab, ba obsahují stejný počet symbolů a a b (1), proto $ab, ba \in L$. Řetězce aa, bb neobsahují stejný počet symbolů a a b, proto $aa, bb \notin L$. Jelikož derivace $S \Rightarrow SaSb \Rightarrow aSb \Rightarrow ab$ a $S \Rightarrow SbSa \Rightarrow bSa \Rightarrow ba$ jsou součástí gramatiky G, tak $ab, ba \in L(G)$. Obdobné derivace, které by generovaly řetězce aa, bb nejsou součástí gramatiky G, proto $aa, bb \notin L(G)$. Z toho vyplývá, že pro $i = 2$ tvrzení $L \subseteq L(G)$ platí.

Indukční hypotéza:

- Předpokládejme, že pro všechny řetězce w takové, že $|w| \leq i$, platí $w \in L \Rightarrow w \in L(G)$.

Indukční krok:

- Ukažme platnost tvrzení pro délku slova $i + 2$. Je zřejmé, že řetězce délky $i + 1$ nejsou součástí jazyka L, jelikož nemohou obsahovat stejný počet symbolů a a b.
- Uvažujme řetězec $w_j \in L$ délky $i + 2$. Mohou nastat následující situace:
  - $w_1 = av_1b$, kde $v_1 \in L$
  - $w_2 = bv_2a$, kde $v_2 \in L$
  - $w_3 = av_3a$, kde $v_3 \in \{w \mid w \in \{a, b\}^* \wedge \#_a(w) - 2 = \#_b(w)\}$
  - $w_4 = bv_4b$, kde $v_4 \in \{w \mid w \in \{a, b\}^* \wedge \#_a(w) = \#_b(w) - 2\}$
- V gramatice G pak existují derivace:
  - $S \Rightarrow aSbS \Rightarrow aSb \Rightarrow^* w_1$
  - $S \Rightarrow bSaS \Rightarrow bSa \Rightarrow^* w_2$
  - $S \Rightarrow aSbS \Rightarrow aSbSbSa \Rightarrow^* w_3$
  - $S \Rightarrow bSaS \Rightarrow bSaSaSb \Rightarrow^* w_4$
- Tvrzení $L \subseteq L(G)$ platí i pro $i + 2$.

Závěr: bylo dokázáno tvrzení 1 i tvrzení 2, z toho vyplývá že $L = L(G)$.

Příklad 2

Problém $L^P(M) = \emptyset$ je rozhodnutelný a řeší ho například následující algoritmus.

Návrh algoritmu

- **Vstup:** Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, g_f)$
- **Výstup:** $L^P(M) = \emptyset$ nebo $L^P(M) \neq \emptyset$
- **Metoda:**

1. Postupně transformujeme turingův stroj M na rozšířený konečný automat M'.

$$M' = (Q', \Sigma, \delta', q_0, F)$$

2. Množina stavů $Q' \subseteq Q$:

$$Q' = \{q_0\} \cup \{q \mid q \in Q \wedge \exists a \in \Sigma \cup \{\Delta\}\ \exists p \in Q : (p, a) \to (q, R) \in \delta\}$$

3. Přechodová funkce δ':[1]

$$\delta' = \{(q, a) \to p \mid q, p \in Q' \wedge a \in \Sigma \cup \{\Delta\} \wedge (q, a) \to (p, R) \in \delta\} \cup$$

$$\cup \{(q, a) \to p \mid q, p \in Q' \wedge a \in \Sigma \cup \{\Delta\} \wedge \exists q_1, ..., q_n \in Q :$$

$$: (q, a) \to (q_1, a) \in \delta \wedge (q_1, a) \to (q_2, a) \in \delta \wedge ... \wedge (q_n, a) \to (p, R) \in \delta\}$$

4. Množina koncových stavů F:

$$F = \{q_f\}$$

5. Rozšířený konečný automat M' převedeme dle algoritmu 3.6[2] na deterministický konečný automat M''.

$$M'' = (Q'', \Sigma, \delta'', q_0', F)$$

6. Jazyk $L(M'') = L^P(M) \in \mathcal{L}_3$ a dle věty 3.2[2] je problém neprázdnosti jazyka ve třídě $\mathcal{L}_3$ rozhodnutelný. A sice:

$$\exists w \in \Sigma^* : (q_0', w) \vdash^* (q_f, \varepsilon) \Rightarrow L^P(M) \neq \emptyset$$

$$\nexists w \in \Sigma^* : (q_0', w) \vdash^* (q_f, \varepsilon) \Rightarrow L^P(M) = \emptyset$$

[1] Je vhodné doplnit, že symbol Δ značící prázdný symbol v přechodové funkci δ je ekvivalentní se symbolem ε, značící prázdný řetězec v přechodové funkci δ'.

[2] http://www.fit.vutbr.cz/study/courses/TIN/public/Texty/TIN-studijni-text.pdf

Příklad 3

Nerozhodnutelnost problému L_{42} dokážeme zavedním redukce σ z problému nezastavení (L_{CoHP}). Nejprve si uvedené jazyky formálně definujeme.

- $L_{42} = \{\langle M\rangle \mid \text{TS } M \text{ definovaný kódem } \langle M\rangle \text{ zastaví na některém vstupu tak, že páska bude obsahovat právě 42 neblankových symbolů}\} \subseteq \{0,1\}^*$
- $L_{CoHP} = \{\langle M\rangle\#\langle w\rangle \mid \text{TS } M \text{ definovaný kódem } \langle M\rangle \text{ na vstupu } w \text{ definovaným kódem } \langle w\rangle \text{ nezastaví}\} \subseteq \{0,1,\#\}^*$

Myšlenka redukce

Funkce σ je realizována úplným turingovým strojem M_σ tak, že pro instanci $x \in L_{CoHP}$ zapíše na výstupní pásku kód turingova stroje M_x, který se chová následovně:

1. Vymaže obsah svoji vstupní pásky.
2. Na vstupní pásku zapíše řetězec $x = x_1\#x_2$, ze kterého byl sám vygenerován. Ten má uložen jako konstantu ve svém stavovém řízení.
3. Provede analýzu, zda řetězec x je platnou instancí L_{CoHP}. Pokud ne, odmítne, a tedy $L(M_x) = \emptyset$.
4. Odsymuluje běh turingova stroje s kódem x_1 na vstupu s kódem x_2 s využitím univerzálního turingova stroje, který je jeho komponentou. Pokud běh skončí, pak pokračuje dalším krokem, jinak cyklí.
5. Po dokončení běhu spočítá počet neblankových symbolů na pásce. Pokud napočítá 43, tak přijme a $L(M_x) = \{0,1\}^*$. Pokud je na pásce méně než 43 neblankových symbolů bude cyklit.

Realizace redukce

- M_σ lze realizovat jako úplný turingův stroj. M_σ vygeneruje kód turingova stroje M_x, který se skládá z několika menších komponent mezi kterými předává řízení. Jedná se o komponenty:
    1. Kód turingova stroje M_{erase} pro smazání vstupní pásky. Všechny výchozí symboly na pásce, přepíše na Δ.
    2. Kód turingova stroje M_{write_x}, který na vstupní pásku zapíše řetězec x. Posouvá hlavu doprava a zapisuje jednotlivé symboly řetězce x.
    3. Kód turingova stroje M_{valid}, který ověří, jestli řetězec $x = x_1\#x_2$ je platnou instancí L_{CoHP}. Jedná se o test na členství v regulárním jazyce.
    4. Kód univerzálního turingova stroje M_{UTS} pro simulaci chování turingova stroje na daném vstupu.
    5. Kód turingova stroje M_{more_42} pro počítání neblankových symbolů. Ten má 44 řídících stavů a v každém z nich, kromě posledního koncového, buď přes Δ cyklí a posunuje čtecí hlavou doprava, nebo přes cokoliv jiného přechází do dalšího stavu a čtecí hlavou posunuje taktéž doprava. V momentě, kdy dojde do koncového stavu, je na pásce 43, nebo více symbolů.
    6. Pomocnou logiku pro propojení výstupů s vstupy jednotlivých komponent.
- Redukce je potom funkce $\sigma : \{0,1,\#\}^* \to \{0,1\}^*$, která je realizována úplným turingovým strojem M_σ.