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:

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

   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