FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Teoretická informatika (TIN) – 2019/2020
Úkol 2

Jakub Svoboda (xsvobo0z)

12. 12. 2019

1. Příklad číslo 1

Dokažte, že jazyk $L = \{w \in \{a,b\}^* \mid \#_a(w) = \#_b(w)\}$ je bezkontextový.

(a) $G = (\{S\}, \{a,b\}, P, S)$ s pravidly:
$S \rightarrow aSb \mid bSa \mid abS \mid baS \mid \varepsilon$

(b) Dokazujeme, že I. $L(G) \subseteq L$ a II. $L \subseteq L(G)$

I. Bázový případ pro i = 0:

$\varepsilon \in L(G)$ protože $S \rightarrow \varepsilon \ \wedge \ \varepsilon \in L$, tedy pro i=0 teze platí.

Indukční krok:

Nechť pro i platí: $w, |w| \leq i \wedge w \in L(G) \rightarrow w \in L$

Ukážeme platnost i pro délku i+2.

Slovo délku i+2 generuje: $S \rightarrow aSb \mid bSa \mid abS \mid baS \rightarrow^* aw'b \mid bw'a \mid abw' \mid baw'$

Dle indukčního předpokladu: $S \rightarrow^* w' \ \wedge \ |w'| \leq i \ \rightarrow \ w' \in L$ tedy má tvar $\{a,b\}^* \wedge \#_a(w') = \#_b(w')$

$w = \{a,b\}^* w' \{a,b\}^* \wedge \#_a(w) = \#_b(w) \in L.$

II. Bázový krok:

$\varepsilon \in L \ \wedge \ S \rightarrow \varepsilon$, tedy tvrzení pro i=0 platí.

Indukční krok:

Nechť pro i platí: $|w| \leq i \ \wedge \ w \in L \ \rightarrow \ S \rightarrow^* w$

Ukážeme platnost pro i+2.

Slovo délky i+2 generuje: $\{a,b\}^* \wedge \#_a(w) = \#_b(w) = i+1$, tedy jednu z možností:

$aw'b, bw'a, abw', baw' \wedge |w'| = i \wedge w' \in L$

Dle indukčního předpokladu $S \rightarrow^* w'$. w lze generovat $S \rightarrow aSb \mid bSa \mid abS \mid baS \rightarrow aw'b \mid bw'a \mid abw' \mid baw'$.

Jelikož jsme dokázáli platnost $L(G) \subseteq L$ a $L \subseteq L(G)$, je tedy zřejmé, že $L(G) = L$.

2. Příklad číslo 2

3. Příklad číslo 3

4. Příklad číslo 4

(a) Dokažte, že pro každý TS M nad abecedou {0,1} a řetězcem $w \in \{0,1\}^*$ 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 $w \in L(M)$.

Vstupní páska TS má tvar $\Delta w \Delta^\omega$. Obsah této pásky je nutné přetransformovat na konfiguraci $C = (pos, dir, grid)$. Můžeme uvažovat, že startovní pozice Karla bude na pozici (1,0) a jeho výchozí směr

natočení bude nahoru(↑). Konečná parciální funkce *grid* pak bude záviset na obsahu vstupní pásky. Tu nastavavíme tak, že první (levý) symbol Δ umístíme pod Karla a řetězec w se poté bude nacházet na pozicích napravo od něj. Jelikož však toto zakódovaní musíme provést pomocí šroubů, zvolíme si následující kódování:

Δ = 0 šroubů
0 = 1 šroub
1 = 2 šrouby
# = 3 šrouby

Znak # bude v tomto případě sloužit k detekci propadnutí hlavy (přílišnému posunu doleva). Tento znak bude vložen před samotný obsah pásky. Ta bude tedy mít tvar před zakódováním #ΔwΔ$^{\omega}$.

Samotné stavy TS jsou v jazyku Karel@TIN reprezentovány jako sekvence příkazů. Posuvy po pásce TS pak budou znamenat posun Karla po matici. TS M přijímá w přechodem do koncového stavu, lze tedy říci, že realizace koncového stavu bude následující:

```
q_f: return 1;
```

Symboly *q_f, q, q_0* a pod. jsou symbolickým označením řádku na kterém začíná sekvence příkazů odpovídajícím danému stavu. Realizace nějakého nekoncového stavu q z TS M můžeme pak vyjádřit následujícím pseudo kódem:

```
q:
q_0: Ověř, zda je symbol roven Δ. Pokud ne, skoč na q_1
q_0_p: Vykonej přechod definovaný pro d(q, Δ)
q_1: Ověř, zda je symbol roven 0. Pokud ne, skoč na q_2
q_1_p: Vykonej přechod definovaný pro d(q, 0)
q_2: Ověř, zda je symbol roven 1. Pokud ne, skoč na q_3
q_2_p: Vykonej přechod definovaný pro d(q, 1)
q_3: return 0; //znak #, propadnutí hlavy
```

V případě, že pro daný stav není přechod pro daný symbol definovaný, pak je sekvence příkazů `“vykonej přechod”` nahrazena za `return 0;` Vykonání definovaného přechodu by pak odpovídalo skoku na návěští značící daný stav. Samotné ověřovnání čtení symbolu, posunu po pásce a zápis symbol může být výkonán např. pomocí níže specifikovaných funkcí:

Pomocné funkce:

```
noop:
A:    turn_left;
B:    turn_left;
C:    turn_left;
D:    turn_left;

step_left:
A:    turn_left;
B:    step;
C:    turn_left;
```

```
D:    turn_left;
E:    turn_left;

step_right:
A:    turn_left;
B:    turn_left;
C:    turn_left;
D:    step;
E:    turn_left;

clear:
A:    if empty: goto D;
B:    lift_screw;
C:    if not empty: goto B
D:    noop;
```

Dále si zadefinujeme pomocné makro JUMP(state), které se expanduje na sekvenci příkazů: `if empty: goto state; if not empty: goto state;`

Zápisy:

```
zapis_1 (2 šrouby):
A:    clear;
B:    drop_screw;
C:    drop_screw;

zapis_0 (1 šroub):
A:    clear;
B:    drop_screw;

zapis_Δ (0 šroubů):
A:    clear;

zapis_# (3 šrouby, při incializaci)
A:    clear;
B:    drop_screw;
C:    drop_screw;
D:    drop_screw;
```

Čtení pod hlavou:

```
kontrola_# (tří šroubů):
A:    if empty: goto P;
B:    lift_screw;
C:    if empty: goto O;
D:    lift_screw;
```

```
E:    if empty: goto N;
F:    lift_screw;
G:    if not empty: goto M;
H:    drop_screw;
I:    drop_screw;
J:    drop_screw;
K:    zápis nebo posun (Left/Right);
L:    JUMP(P)
M:    drop_screw;
N:    drop_screw;
O:    drop_screw;
P:    noop;
```

```
kontrola_1 (dvou šroubů):
A:    if empty: goto L;
B:    lift_screw;
C:    if empty: goto K;
D:    lift_screw;
E:    if not empty: goto J;
F:    drop_screw;
G:    drop_screw;
H:    zápis nebo posun (Left/Right);
I:    JUMP(L)
J:    drop_screw;
K:    drop_screw;
L:    noop;
```

```
kontrola_0 (1 šroub):
A:    if empty: goto H;
B:    lift_screw;
C:    if not empty: goto G;
D:    drop_screw;
E:    zápis nebo posun (L/R);
F:    JUMP(H)
G:    drop_screw;
H:    noop;
```

```
kontrola_ Δ (0 šroubů):
A:    if not empty: goto C;
B:    zápis nebo posun (L/R);
C:    noop;
```

Výše popsané funkce obsahují na levé straně písmena která reprezentují číslené označení jednotlivých po sobě jdoucích řádků. Pokud program skončí ve stavu q_f, program vykoná příkaz `return 1;` a výstupem programu je tedy jednička.

Na základě výše uvedeného popisu je dokázané, že k TS M jsme schopno sestrojit program v jazykce Karel@TIN který je ekvivalentní, tedy má stejnou funkčnost jako TS.

(b) Dokažte, že 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 hodnoty konfigurace C skončí s návratovou hodnotou 1.