Fakulta informačních technologií
Vysoké učení technické v Brně

TIN
Teoretická informatika

2. domáca úloha

Adrián Tóth (xtotha01) 3. decembra 2018

Obsah

1 Príklad číslo 1

1.1 (a)

Definice 4.29 [1](str. č. 97) Označme ZAV_n pre $n \geq 0$ jazyky setávající ze všech vyvážených řetězců závorek n typů. Tyto jazyky – označované též jako Dyckovy jazyky – jsou generovány gramatikami s pravidly tvaru: $S \rightarrow [^1\ S\ ]^1 \mid [^2\ S\ ]^2 \mid ... \mid [^n\ S\ ]^n \mid SS \mid \varepsilon$

Z hore uvedenej definície pre náš príklad vyplýva, že náš Dyckov jazyk L je generovaný gramatikou G_D, definovanou ako $G_D = (\{S'\}, \{[,]\}, P, S')$ kde množina prepisovacích pravidiel P je daná ako

$$S' \rightarrow \varepsilon \mid S'S' \mid [\ S'\ ]$$

ktorá obsahuje iba jeden typ zátvoriek ktorými sú [a].

Pre každé slovo $w \in L$, pre ktoré platí že $w \neq \varepsilon$, muselo byť aspoň raz použité pravidlo $S' \rightarrow [\ S'\ ]$ v derivácii. S využitím pravidla $S' \rightarrow [\ S'\ ]$ vygenerujeme jeden pár zátvoriek, pričom medzi zátvorkami sa nachádza neterminál S', ktorým je možné ďalej aplikovať ďalšie pravidlá a generovať reťazec u za pomoci gramatiky G_D – presnejšie reťazec u patriaci do jazyka L. V prípade použitia pravidla $S' \rightarrow S'S'$ pred pravidlom $S' \rightarrow [\ S'\ ]$ vieme generovať ďalší reťazec na pravej strane čo odpovedá reťazcu v generovaným G_D ktorý patrí do L, t.j. vieme generovať za $[u]$ reťazec v patriaci do jazyka L za pomoci G_D. Pri viacnásobnom použití pravidla $S' \rightarrow S'S'$ toto tvrdenie stále platí, pretože za $[u]$ sa vezme najľavejšia derivácia $S' \rightarrow [\ S'\ ]$ a v bude reprezentované pravou časťou od symbolu].

Takže, každé slovo $w \in L$, pre ktoré platí že $w \neq \varepsilon$, vieme zapísať v tvare $[u]v$ kde $u, v \in L$ pretože

$$S' \underset{G_D}{\Rightarrow} S'S' \underset{G_D}{\Rightarrow} [S']S' \underset{G_D}{\Rightarrow}^* [u]v$$

Keďže $S' \underset{G_D}{\Rightarrow}^* u$ a $S \underset{G_D}{\Rightarrow}^* v$ ktoré patria do jazyka L, tak $S' \underset{G_D}{\Rightarrow}^* [u]v$ tiež patrí do jazyka, keďže $u, v \in L$ a $[,] \in L$. Je zrejmé, že ak $S' \underset{G_D}{\Rightarrow}^* w$ a $S' \underset{G_D}{\Rightarrow}^* [u]v$ tak potom platí že $S' \underset{G_D}{\Rightarrow}^* w = [u]v$ t.j. $w = [u]v$.

1.2 (b)

Báza

Báza je bázový prípad **pre $\mathbf{i = 0}$**.

Pre $i = 0$ platí, že počet [a počet] v reťazci w je rovno nule z čoho vyplýva, že reťazec w sa musí rovnať ε. Formálne, pre $i = 0$ platí, že $\#_[(w) = 0 \wedge \#_](w) = 0$ z čoho vyplýva, že $w = \varepsilon$.

Keďže, existuje pravidlo $S' \rightarrow \varepsilon$ v gramatike G_D a taktiež, existuje pravidlo $S \rightarrow \varepsilon$ v gramatike G, potom existujú derivácie $S' \underset{G_D}{\Rightarrow} \varepsilon$ a $S \underset{G}{\Rightarrow} \varepsilon$ a tak platí, že $w = \varepsilon \wedge w \in L \wedge w \in L(G)$.

Indukčný predpoklad

$S \Rightarrow^* w$ kde $w \in L \wedge w \in L(G) \wedge w = [u]v$ čo platí pre všetky j kde $j < i$ pričom j, i značí počet zátvoriek typu [a počet zátvoriek typu].

Pre i

$\#_{[}(u) + \#_{[}(v) \stackrel{?}{=} i$ analogicky pre $\#_{]}(u) + \#_{]}(v) \stackrel{?}{=} i$

1.) $\#_{[}(u) = 0 \Rightarrow \#_{[}(v) = i \quad \vee \quad \#_{[}(v) = 0 \Rightarrow \#_{[}(u) = i$

Ak je buď $\#_{[}(u)$ alebo buď $\#_{[}(v)$ rovné nule, tak ho vieme vygenerovať z pravidla S na základe indukčnej bázi.

Ak je buď $\#_{[}(u)$ alebo buď $\#_{[}(v)$ rovné i, tak ten prvok prepíšeme pomocou vzorca

$$w' = [u']v' \Rightarrow \#_{[}(u') + \#_{[}(v') = i - 1 = j, \text{ analogicky } \#_{]}(u') + \#_{]}(v') = i - 1 = j$$

Na základe indukčného predpokladu vieme z S vygenerovať u' a v'.

$$S \Rightarrow [S]S \Rightarrow^* [u']v' = w'$$

kde

$$\#_{[}(u') + \#_{[}(v') = i, \text{ analogicky } \#_{]}(u') + \#_{]}(v') = i$$

2.) $\#_{[}(u) \neq 0 \quad \wedge \quad \#_{[}(v) \neq 0 \quad \wedge \quad \#_{[}(u) + \#_{[}(v) \leq i$

Keď $\#_{[}(u)$ a $\#_{[}(v)$ sú nenulové, tak musí platiť že

$$\#_{[}(u) < i \wedge \#_{[}(v) < i$$

t.j.

$$\#_{[}(u) = i - M = j_1 \wedge \#_{[}(v) = i - N = j_2 \text{ kde } M, N \in \mathbb{N} \setminus \{0\}$$

Pre i + 1

$S \Rightarrow^* w$ kde $w \in L$ pre ktoré platí, že

$$\#_{[}(w) = i + 1$$

a z definicie Dyckovho jazyka platí, že

$$\#_{]}(w) = i + 1$$

Potom vieme w zapísať ako $w = [u]v$ podľa bodu (a)(kapitola 1.1) $\Rightarrow \#_{[}(u) + \#_{[}(v) = i$, analogicky musí platiť $\#_{]}(u) + \#_{]}(v) = i$.

$S \Rightarrow [S]S \Rightarrow^* [u]v = w$ kde $\#_{[}(u) + \#_{[}(v) = i + 1$, analogicky $\#_{]}(u) + \#_{]}(v) = i + 1$.

2 Príklad číslo 2

Veta 4.19 [1](str. č. 92): Nechť L je bezkontextový jazyk. Pak existuje konstanta $k > 0$ taková že je-li $z \in L$ a $|z| \geq k$, pak lze z napsat ve tvaru:

$$z = uvwxy, vx \neq \varepsilon, |vwx| \leq k$$

a pro všechna $i \geq 0$ je $uv^iwx^iy \in L$.

Nech L_{primes} je bezkontextový jazyk.

Tak existuje celočíselná konštanta $k > 0$ taká, že ak $z \in L$ a $|z| \geq k$, tak

$$z = uvwxy \wedge vx \neq \varepsilon \wedge |vwx| \leq k \wedge uv^iwx^iy \in L \text{ kde } i \geq 0$$

Zvoľme prvočíslo r väčšie ako ako k t.j. $r \geq k$ kde r je prvočíslo.

Potom platí, že

$$a^r \in L \wedge |a^r| = r \text{ kde } r \geq k \implies a^r = uvwxy \wedge vx \neq \varepsilon \wedge |vwx| \leq k \wedge uv^iwx^iy \in L \text{ pre } i \geq 0$$

Nech

$$v = a^m \Rightarrow |v| = m$$
$$x = a^n \Rightarrow |x| = n$$
$$w = a^o \Rightarrow |w| = o$$

Tak musí platiť že $m + n > 0$ pretože $vx \neq \varepsilon$ a $k \geq m + n + o$ pretože $|vwx| \leq k$.

Zvoľme $i = r + 1$, potom

$$uv^{r+1}wx^{r+1}y \in L$$
$$|uv^{r+1}wx^{r+1}y| = |uvwxy| + |v^r| + |x^r| = r + r \cdot m + r \cdot n = r \cdot (1 + m + n) \text{ čo nie je prvočíslo}$$

A z toho vyplýva spor pretože

$$uv^{r+1}wx^{r+1}y \notin L$$

Takže jazyk L_{primes} nie je bezkontextový jazyk.

3 Príklad číslo 3

3.1 Nerozhodnuteľnosť

Problém môžeme charakterizovať jazykom L pre ktorý platí

$$L = \{ \langle M_L \rangle \mid M_L \text{ je } TS : \exists w \in \textit{Affine} : w \in L(M_L) \}$$

Problém členstva je charakterizovaný jazykom MP pre ktorý platí

$$MP = \{ \langle M_{MP} \rangle \# w \mid M_{MP} \text{ je } TS \text{ ktorý prijme } w \}$$

Zostavíme redukciu

$$\sigma : \{0, 1, \#\}^* \longrightarrow \{0, 1\}^* \text{ z jazyka } MP \text{ na } L$$

TS M_σ implementujúci σ priradí každému vstupu $x \in \{0, 1, \#\}^*$ reťazec $\langle M_x \rangle$, kde M_x je TS, ktorý na vstupu $y \in \{0, 1\}^*$ pracuje následovne:

1. M_x zmaže svoj vstup y.
2. Zapíše na pásku reťazec x.
3. M_x posúdi, zda $x = x_1\#x_2$ pre x_1, ktorý je kódom TS, a x_2, ktorý je kódom jeho vstupu. Pokiaľ nie, odmietne.
4. Inak M_x simuluje činnosť TS s kódom x_1 na reťazci s kódom x_2.
   - Ak x_1 prijme x_2, tak M_x prijme.
   - Ak x_1 odmietne x_2, tak M_x odmietne.
   - Inak cyklí.

M_σ je možné implementovať úplným TS. Konečne tento TS vypíše kód M_x, ktorý sa skladá zo štyroch komponent, ktoré odpovedajú vyššie uvedeným krokom. Tri z nich sú pritom konštantné (nezávisia na x) – konkrétne (1) zmazanie pásky, (2) test na dobré sformovanie instancie MP a (3) simulácia daného TS na danom vstupe (pomocou úplného TS). TS implementujúci tieto kroky, ktoré evidentne existujú, môžeme pripraviť vopred a M_σ vypíše kód spolu s kódom na predanie riadenia. Zostáva vygenerovať kód TS, ktorý zapíše na pásku dané $x = a_1a_2...a_n$. To je možné ale ľahko realizovať pomocou TS $Ra_1Ra_2R...Ra_n$.

Skúmajme možné jazyky TS M_x:

- $L(M_x) = \emptyset \iff$ (x nie je správne sformovaná instancia MP) alebo ($x = x_1\#x_2$ a TS s kódom x_1 na reťazci s kódom x_2 odmietne) alebo ($x = x_1\#x_2$ a TS s kódom x_1 na reťazci s kódom x_2 neskončí t.j. cyklí)
- $L(M_x) = \Sigma^* \iff$ (x je správne sformovaná instancia MP, kde $x = x_1\#x_2$ a TS s kódom x_1 na reťazci s kódom x_2 prijme)

Ak $L(M_x) = \Sigma^*$ je zrejmé, že jazyk $L(M_x)$ iste obsahuje aspoň jeden reťazec ktorý patrí do jazyka *Affine*.

Teraz už ľahko ukážeme, že σ zachováva členstvo $\langle M_x \rangle \in L \Leftrightarrow L(M_x) = \Sigma^* \Leftrightarrow x = x_1\#x_2$ kde x_1 je kód TS, ktorý zastaví na vstupe s kódem $x_2 \Leftrightarrow x \in MP$.

3.2 Čiastočná rozhodnuteľnosť

Majme $TS\ M$ pre ktorý platí, že $\exists w \in \mathit{Affine} : w \in L(M)$ kde jazyk *Affine* je rekurzívny jazyk.

K čiastočnému rozhodovaniu uvedeného problému môžeme využiť $TS\ M'$ ktorý na svojej prvej páske simuluje beh $TS\ M$ pre jednotlivé možné vstupné reťazce podľa určitého usporiadania (viz dole uvedený príklad) a na druhej páske vykonáva kontrolu podmienky patričnosti reťazca do jazyka *Affine*.

$$
\begin{array}{l}
konf_M(\varepsilon, 0) \\
konf_M(\varepsilon, 1)\#konf_M(0, 0) \\
konf_M(\varepsilon, 2)\#konf_M(0, 1)\#konf_M(1, 0) \\
konf_M(\varepsilon, 3)\#konf_M(0, 2)\#konf_M(1, 1)\#konf_M(00, 0) \\
konf_M(\varepsilon, 4)\#konf_M(0, 3)\#konf_M(1, 2)\#konf_M(00, 1)\#konf_M(01, 0) \\
\ldots
\end{array}
$$

M' nemôže len systematicky generovať vstupy pre $TS\ M$ a na nich nechať $TS\ M$ neobmedzene bežať pretože hrozí zacyklenie.

$TS\ M'$ ale môže mať na svojej páske súčastne rozbehnutú simuláciu $TS\ M$ pre ľubovoľný počet vstupných reťazcov, jednotlivé konfigurácie pásky budú vhodne oddelené.

$TS\ M'$ môže vždy prejsť všetky aktuálne rozbehnuté simulácie a na každej vykonať práve jeden krok. Pokiaľ v niektorom prípade dôjde k prijatiu reťazca $TS\ M$, vykoná kontrolu pratričnosti prijatého reťazca do jazyka *Affine* na druhej páske. Ak kontrola patričnosti prijatého reťazca skončila úspechom, t.j. reťazec bol prijatý $TS\ M$ a súčastne patrí do jazyka *Affine* môžeme tvrdiť, že existuje reťazec ktorý patrí do *Affine* a súčastne do $L(M)$. Ak $TS\ M$ neprijal reťazec alebo reťazec nepatrí do jazyka *Affine*, tak $TS\ M'$ pridá páskovú konfiguráciu pre ďalší reťazec a celý krok opakuje.

$TS\ M'$ vykonáva kontrolu patričnosti reťazca do jazyka *Affine* na druhej páske tak, že TODO.

Funkčnosť $TS\ M'$ môžeme chápať tak, že najskôr $TS\ M'$ spustí simuláciu $TS\ M$ na prvej páske. Ak $TS\ M$ zastaví a prijme určitý reťazec, tak tento reťazec $TS\ M'$ overí na druhej páske zda patrí do jazyka *Affine* t.j. skontroluje podmienku patričnosti tohto reťazca do jazyka *Affine*.

4 Príklad číslo 4

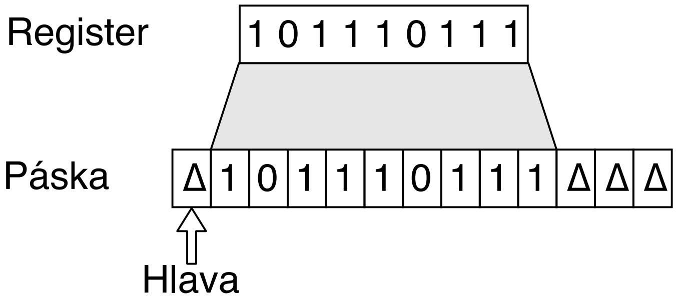
4.1 (a)

...

4.2 (b)

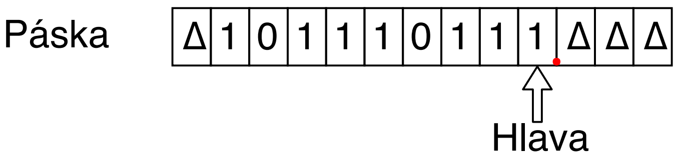
Dôkaz, že pre každý program P v jazyku *RationalC* a počiatočnú hodnotu $x_0 \in \mathbb{N}$ $(0 \in \mathbb{N})$ je možné zostrojiť TS $M_P = (Q, \Sigma, \Gamma, \delta, q_{first}, F)$ a reťazec $w \in \{0,1\}^*$ tak, že $w \in L(M_P)$ práve vtedy, keď P s počiatočnou hodnotou x_0 skončí s návratovou hodnotou 1.

Na začiatku si zapíšeme obsah registra x v ktorom sa nachádza číslo $x_0 \in \mathbb{N}$ v binárnej podobe na pásku TS M_P čo je zobrazené v ilustračnom obrázku 1.



Obr. 1: Ilustračný obrázok prevodu registra x na pásku TS M_P.

Potom, pre TS M_P je potreba inicializovať hlavu čo v našom prípade znamená, že posunieme hlavu TS M_P na *least significant bit* (*LSB*) t.j. na *most right nonblank symbol* (najpravejší neblankový symbol) čo je realizovateľné TS. Táto inicializácia nášho TS M_P je nutná z toho dôvodu, že na začiatku máme celé nezáporné číslo ale neskôr sa môžeme dopracovať ku desatinným číslam kde je potreba vedieť kde je desatinná čiarka. V TS M_P budeme desatinnú čiarku reprezentovať tak, že pozícia hlavy ukazuje na *LSB* čísla z čoho vieme, že desatinná bodka sa nachádza na pravej strane od tohto *LSB* čísla. Proces inicializácie hlavy je ilustrovaný na obrázku 2 v ktorom je hypotetická desatinná bodka znázornená červenou bodkou.



Obr. 2: Ilustračný obrázok inicializácie hlavy TS M_P na páske.

Predpokladajme, že zdrojový kód pre program P v jazyku *RationalC* má určitú formu kde každý jeden príkaz je zapísaný na osobitnom riadku kde všetky riadky sú očíslované od 0 (pre prvý riadok) až po N (posledný riadok).

Potom, pre každý jeden riadok, kde označíme číslo riadka ako N, vytvoríme jeden stav $q_N \in Q$ a pre každý príkaz na týchto riadokoch platí, že jednotlivé príkazy vieme previesť podľa nasledujúcich pravidiel.

- Príkaz `if` x % $2 == A$ `goto` B kde $A = \{0, 1\}$ a $0 \leq B \leq N_{MAX}$ na riadku N

  $TS\ M_P$ bude obsahovať nasledujúce prechodové pravidlá pre tento príkaz

  $$\delta(q_N, A) = (q_B, A)$$
  $$\delta(q_N, \bar{A}) = (q_{N+1}, \bar{A})$$

  kde $q_B, q_N, q_{N+1} \in Q$.

  Tento príkaz je možné chápať ako zmena stavu v $TS\ M_P$ pričom sa hlava neposunie a ponechá sa pôvodný symbol pod pozíciou hlavy.

- Príkaz x `/=` 2 na riadku N

  $TS\ M_P$ bude obsahovať nasledujúce prechodové pravidlá pre tento príkaz

  $$\delta(q_N, 0) = (q_{N+1}, L)$$
  $$\delta(q_N, 1) = (q_{N+1}, L)$$

  kde $q_N, q_{N+1} \in Q$.

  Tento príkaz je možné chápať ako posun hlavy $TS\ M_P$ do ľava.

- Príkaz x `*=` 2 na riadku N

  $TS\ M_P$ bude obsahovať nasledujúce prechodové pravidlá pre tento príkaz

  $$\delta(q_N, 0) = (q_{N+1}, R)$$
  $$\delta(q_N, 1) = (q_{N+1}, R)$$

  kde $q_N, q_{N+1} \in Q$.

  Tento príkaz je možné chápať ako posun hlavy $TS\ M_P$ do prava.

- Príkaz `return` 0 na riadku N

  $TS\ M_P$ bude obsahovať nasledujúce prechodové pravidlá pre tento príkaz

  $$\delta(q_N, 0) = (q_{reject}, 0)$$
  $$\delta(q_N, 1) = (q_{reject}, 1)$$

  kde $q_N, q_{reject} \in Q$ a $q_{reject} \in F$.

  Tento príkaz je možné chápať ako prechod $TS\ M_P$ do koncového stavu ktorý je odmietajúci.

- Príkaz `return` 1 na riadku N

  $TS\ M_P$ bude obsahovať nasledujúce prechodové pravidlá pre tento príkaz

  $$\delta(q_N, 0) = (q_{accept}, 0)$$
  $$\delta(q_N, 1) = (q_{accept}, 1)$$

  kde $q_N, q_{accept} \in Q$ a $q_{accept} \in F$.

  Tento príkaz je možné chápať ako prechod $TS\ M_P$ do koncového stavu ktorý je akceptujúci.

- Príkaz `odd`(x) na riadku N

  $TS\ M_P$ bude obsahovať nasledujúce prechodové pravidlá pre tento príkaz

  $$\delta(q_N, 0) = (q_{N+1}, 1)$$
  $$\delta(q_N, 1) = (q_{N+1}, 1)$$

  kde $q_N, q_{N+1} \in Q$.

  Tento príkaz je možné chápať ako zápis symbolu 1 na aktuálnej pozícii hlavy.

- Príkaz `even`(x) na riadku N

  $TS\ M_P$ bude obsahovať nasledujúce prechodové pravidlá pre tento príkaz

  $$\delta(q_N, 0) = (q_{N+1}, 0)$$
  $$\delta(q_N, 1) = (q_{N+1}, 0)$$

  kde $q_N, q_{N+1} \in Q$.

  Tento príkaz je možné chápať ako zápis symbolu 0 na aktuálnej pozícii hlavy.

Pri násobení a delení (pri použití príkazov x `/=` 2 a x `*=` 2) môže ale dôjsť ku tomu, že sa hlava dostane zo symbolu 0 alebo 1 na symbol blank. Môžu nastať dve možnosti a to keď sa hlava dostaneme na ľavý blank (prvý symbol pásky) alebo na pravý blank (blank za posledným znakom 0 alebo 1 ktorý je uložený na páske). Tieto prípady je potreba vyriešiť pričom môžu nastať iba vtedy, ak sme sa dostali na krajný (najpravejší alebo najľavejší) symbol 0 alebo 1 vedľa ktorého je z ľavej strany blank (pre najľavejší) alebo z pravej strany blank (pre najpravejší). Takže môžu nastať dva prípady.

- Hlava je na najpravejšom symbole 0 alebo 1 a treba vykonať príkaz x `*=` 2.

  Príkladový prípad hlavy a pásky

  $$\Delta 1\underline{0}\Delta\Delta\Delta$$

  Po vykonaní príkazu x `*=` 2 sa hlava dostane z najpravejšieho neblankového symbolu na pravý blankový symbol.

  $$\Delta 10\underline{\Delta}\Delta\Delta$$

  V hore uvedenom prípade by sme mali chybnú hodnotu čísla z toho dôvodu, že u nás pozícia hlavy určuje LSB a tým pádom by sme mali nevalidné číslo keďže jeho LSB by bol blankový symbol. Aby sme sa vyhli tejto situácii, pri každom posune doprava treba vykonať kontrolu zda tam je blankový symbol, ak tam je, tak ho prepíšeme na symbol 0 a posunieme hlavu na tento symbol 0, ak tam nie je, tak iba presunieme hlavu na ten symbol ktorým je 0 alebo 1.

Páska	**Hodnota**	**Riadok**	**Nasledujúci príkaz**
$\Delta\underline{1}0\Delta\Delta\Delta$	1	n	x `*=` 2
$\Delta 1\underline{0}\Delta\Delta\Delta$	2	$n+1$	x `*=` 2
$\Delta 10\underline{0}\Delta\Delta\Delta$	4	$n+2$	x `*=` 2
$\Delta 100\underline{0}\Delta\Delta\Delta$	8	$n+3$	x `*=` 2

  V tabuľke stĺpec '*páska*' zachytáva aktuálny stav pásky spolu s pozíciou hlavy čo je znázornené podčiarknutým červeným symbolom. Stĺpec '*hodnota*' vyjadruje numerickú hodnotu ktorá sa nachádza na páske, '*riadok*' udáva pozíciu v programe a '*nasledujúci príkaz*' zobrazuje príkaz ktorý bude vykonaný.

- Hlava je na najľavejšom symbole 0 alebo 1 a treba vykonať príkaz x `/=` 2.

  Príkladový prípad hlavy a pásky

  $$\Delta\underline{1}0\Delta\Delta\Delta$$

  Po vykonaní príkazu x `/=` 2 sa hlava dostane z najľavejšieho neblankového symbolu na ľavý blankový symbol.

  $$\underline{\Delta}10\Delta\Delta\Delta$$

  Ak by sme znovu vykonali tento príkaz, tak by došlo k abnormálnemu zastaveniu TS M_P pretože by nám spadla hlava. Tento prípad je možné riešiť tak, že ak sme na tomto ľavom blankovom symbole a treba vykonať príkaz pre posun do ľava t.j. vykonaným príkazom bude príkaz x `/=` 2, tak sa celý obsah pásky posunieme o jeden symbol do prava (vykonáme S_R) a pred obsah pásky zapíšeme symbol 0 pričom hlavu musíme vrátiť na pôvodné miesto – na miesto ľavého blanku. Pre ilustráciu funkčnosti viz tabuľku dole.

Páska	**Hodnota**	**Riadok**	**Nasledujúci príkaz**
$\Delta 1\underline{0}\Delta\Delta\Delta$	2	n	x `/=` 2
$\Delta\underline{1}0\Delta\Delta\Delta$	1	$n+1$	x `/=` 2
$\underline{\Delta}10\Delta\Delta\Delta$	0.5	$n+2$	x `/=` 2
$\underline{\Delta}010\Delta\Delta\Delta$	0.25	$n+3$	x `/=` 2
$\underline{\Delta}0010\Delta\Delta\Delta$	0.125	$n+4$	x `/=` 2

  V tabuľke stĺpec '*páska*' zachytáva aktuálny stav pásky spolu s pozíciou hlavy čo je znázornené podčiarknutým červeným symbolom. Stĺpec '*hodnota*' vyjadruje numerickú hodnotu ktorá sa nachádza na páske, '*riadok*' udáva pozíciu v programe a '*nasledujúci príkaz*' zobrazuje príkaz ktorý bude vykonaný.

5 Literatúra

[1] M. Češka, T. Vojnar, A. Smrčka, A. Rogalewicz: *Teoretická informatika - Studijní text.* 2018-08-23, [Online; Accessed: 2018-10-15].
URL: `http://www.fit.vutbr.cz/study/courses/TIN/public/Texty/TIN-studijni-text.pdf`