1. príklad

Zostrojte Turingov stroj, ktorý vracia TRUE/FALSE (teda akceptuje alebo zamieta), podľa toho, či slovo na páske patrí alebo nepatrí do jazyka:

(a) $L_1 = \{ww^R \mid w \in \{a, b\}^*\}$

(b)* $L_2 = \{ww \mid w \in \{a, b\}^*\}$

(d) $L_3 = \{|w|_a = |w|_b \mid w \in \{a, b\}^*\}$;

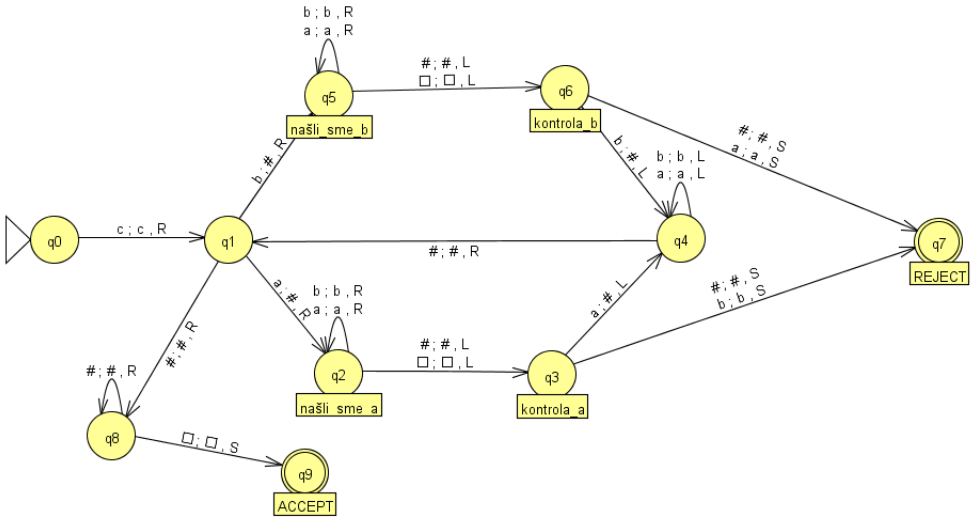
Riešenie 1. príkladu

Riešenie (a) Pri riešení tejto úlohi si treba uvedomiť, že w^R je reverz slova, čiže slovo napísané odzadu. Do jazyka tak budú patriť napríklad slová *aaabbaaa*, *ababbaba* alebo *bbaabb*. Písmeno na začiatku a na konci musia byť rovnaké, treba tak iba posúvať „začiatok" a „koniec" a kontrolovať.

Ako začneme čítať slovo, vedia nám vzniknúť dve vetvy. Buď čítame a a teda hľadáme na konci slova a, alebo čítame na začiatku b a aj na konci slova hľadáme b. Na toto pamätanie nám budú slúžiť stavy (ako pri KA). Ako si poznačíme, že sme nejaké písmeno už prečítali? Môžeme na to využiť napríklad #. Tak sa nám budú postupne na začiatku a konci slova meniť znaky na mriežky a teda mriežky budú naše začiatky a konce. Vždy pôjdeme od mriežky k mriežke a na začiatku slova (za centom alebo neskôr za mriežkou) prečítame symbol a budeme ho hľadať aj na konci slova (pred blankom a neskôr pred mriežkou), ak sú rovnaké, zmeníme znak na mriežku a ideme späť na začiatok. Ak sú rôzne, zamietame.

Ešte sa môže stať, že slovo má nepárnu dĺžku, čiže k nejakému znaku už nenájdeme druhý znak (na páske už zostanú iba mriežky). Vtedy tiež zamietame.

Výsledný TS z cvičení vidíte na obrázku, alebo si toto riešenie môžete mojím a vyskúšať.



Návrh riešenia (b): Podobný príklad sme riešili na predchádzajúcom cvičení, vtedy však medzi slovami w bola mriežka. Tú teraz nemáme, čiže si musíme stred slova zistiť sami. To viete spraviť tak, že za každý znak na začiatkus lova nájdete znak na konci slova a postupnou iteráciou prídete k strednému znaku (dvojici). Narozdiel od riešenia v časti (a), teraz nemôžete prepisovať znaky na mriežky (alebo niečo iné rovnaké), keďže po nájdení polovice slova budete musieť skontrolovať, či je slovo v prvej polovici rovnaké ako slovo v druhej polovici. Preto odporúčam prepisovať a na A a b na B – tak si stále uchováte informáciu o tom, aké je tam písmenko, a zároveň viete, ktoré ste už prečítali a rátali (veľké písmená).

Kedy nájdete stred? Keď po poslednom znaku na začiatku slova bude vpravo tiež veľký znak. Potom už len treba skontrolovať, či slovo od polovice po blank je rovnaké ako slovo od centu do polovice. Na to viete využiť už klasické prepisovanie na mriežky (alebo ľuvoboľný iný označovač). Začnete práve od stredu slova, prečítate prvý znak v druhej polovici a vraciate sa na začiatok. Alebo začnete od stredu slova, prečítate posledný znak v prvej polovici a idete na koniec slova skontrolovať, či je tam rovnaký symbol.

Čo ak zistíte, že slovo je nepárnej dĺžky? Tak asi nebude tvaru ww.

Návrh riešenia (c): chceme zistiť, či je v slove rovnako veľa áčok ako béčok. Môžeme to spraviť tak, že vždy, keď v slove narazíme na a, označíme si, že sme ho prečítali tak, že ho prepíšeme na A, a budeme k nemu hľadať b. Keď ho nájdeme, tiež ho prepíšeme na B, aby sme vedeli, že už sme ho

zarátali. Bude to fungovať opačne – ak nájdeme b, budeme k nemu hľadať a. Toto budeme robiť dovtedy, kým v našom slove buď:

- nebudú všetky písmená veľké, čiže slovo má rovnaký počet áčok a béčok, a akceptujeme ho,
- pre a už v slove nenájdeme b, čiže áčok je viac ako béčok a slovo zamietame,
- pre b už v slove nenájdeme a, čiže béčok je viac ako áčok a slovo zamietame.

Skúste si navrhnúť vlastný TS a porovnajte ho s mojím. Ten nemá až také elegantné riešenie, páskou prechádzame viackrát a znaky berieme náhodne. Určite vymyslíte aj krajšie riešenie. Vyskúšajte všetky možné dobré a aj zlé slová, aby ste si overili, či naozaj funguje správne.

2. príklad

Zostrojte viacpáskový Turingov stroj, ktorý rozoznáva jazyk:

(a) $L_1 = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$

(b) $L_2 = \{w \in \{0,1\}^* \mid |w|_0 = 3|w|_1\}$

(c)* $L_3 = \{w \in \{0,1,2\}^* \mid |w|_1 = 2|w|_2 = |w|_3\}$;

Riešenie 2. príkladu

Skôr ako začneme riešiť, zopakujeme si jednotlivé pojmy. Obyčajné Turingove stroje sú stroje s jednou vstupno-výstupnou páskou, ktorá začína centom a za vstupným slovom je nekonečne veľa blankov. Viacpáskové Turingove stroje obsahujú jednu vstupnú pásku, ktorá začína centom a končí dolárom. Táto vstupná páska je read-only, to znamená že na ňu nevieme nič zapisovať. Ďalej máme viac pomocných (pracovných) pások, ktoré obsahujú na začiatku iba cent a nekonečne veľa blankov. Na tieto môžeme písať, mazať, môžeme tam zapisovať výsledok a podobne. Jednopáskový TS[1] bude stroj, ktorý má jednu vstupnú a jednu pomocnú (pracovnú) pásku. Dvojpáskový bude mať dve pomocné (pracovné) pásky. Každá páska má vlastnú čítaciu hlavu, čo znamená, že ak sa na jednej hýbeme, na ďalšej môžeme aj stáť alebo sa hýbať opačným smerom (samozrejme, v rámci pravidiel, že

[1] v JFLAP treba vybrať dvojpáskový – do počtu pások rátajú aj vstupnú

nejdeme pred cent a za dolár). Z toho vyplýva, že každý prechod bude v tvare[2] *čítam, posúvam | čítam, píšem, posúvam | čítam, píšem, posúvam ...* - keďže na vstupnej páske nevieme nič prepisovať.

Riešenie (a): chceme zistiť, či vo vstupnom slove je rovnaký počet núl ako jednotiek. Pri obyčajnom TS sme museli prechádzať pásku veľakrát (ako napr. v 1 (c)), ale keďže máme pomocné pásky, vieme ich využiť na to, aby sme to zistili už počas jedného prechodu vstupným slovom. Mohli by sme napríklad pre každú nulu na vstupe zapísať jednu nulu na prvú pomocnú pásku (1pp) a za každú jednotku jednu nulu z 1pp vymazať. Bolo to to dobré riešenie?

Pozrime sa, čo by sa stalo so slovom 010101 – najprv by sme na 1pp napísali 0, potom ju vymazali, potom opäť napísali 0, zas vymazali a ešte raz napísali nulu a vzápätí ju vymazali. 1pp by tak zostala prázdna (až na cent) a slovo by sme mohli akceptovať. Čo by sa ale stalo pri slove 101010? Je to tiež slovo, ktoré by malo byť akceptované, ale teraz by sme sa najprv snažili mazať niečo z prvej pásky, hoci by ešte bola prázdna. Čo nie je dobrý prístup.

Dobre, možno ste si povedali, že to, či zapisujeme na 1pp nuly alebo jednotky bude závisieť od toho, čo je prvé na vstupe. Tak si zas ukážme slovo, ktoré nám tento nápad pokazí. Napríklad 100011 – na 1pp by sme písali jednotky, ale hneď na treťom znaku by sme sa snažili vymazať znak z prázdnej pásky. Takže asi ani toto nie je korektný postup.

Možno to len nevieme spraviť pri jednom prechode úplne jednoducho. Tak skúsme spraviť iné riešenie, ktoré možno nebude tak efektívne, a následne sa skúsime vrátiť k tomu, ako to spraviť jedným prechodom.

Čo keby sme si na 1pp napísali nuly zo vstupného slova, pri jednotkách na vstupe by sme sa iba posunuli doprava. Keď by sme prišli na koniec vstupného slova, vracali by sme sa po vstupnej páske doľava a za každú jednotku by sme vymazali 0 z 1pp. Bol by tento postup dobrý pre všetky tri prechádzajúce slová? Áno. Pri všetkých z nich by sme najprv napísali tri nuly na 1pp a následne, pri prechode páskou sprava doľava, by sme ich všetky tri vymazali. Tak by nakonci zostala 1pp prázdna. Čiže náš TS by akceptoval slovo vtedy, keď by sme sa na vstupnej páske dostali na cent a zároveň by bola 1pp prádzna (teda by sme sa tam tiež dostali na cent).

[2] v JFLAP zas nastáva trochu zmena, keďže v ich TS môžete na vstupnú pásku písať, treba tam dávať aj tieto prechody, tj. x, x, SMER, kde x je nejaký znak. Aby ste vedeli vytvoriť TS podľa našej definície, dajte si pozor, aby ste na vstupnej páske nič neprepísali, a pri testovaní zadávajte aj okraje pásky ¢xxx, resp. na pomocné pásky dajte vždy na začiatok ¢. Ja vo svojich TS ¢ nahrádzam c a $ S.

Kedy by náš TS zamietal? Ak by sme chceli mazať niečo z prázdnej 1pp. Alebo ak by na 1pp ešte zostali nuly, ale na vstupe by sme sa už dostali na cent. Skúsme si celý postup spísať v bodoch:

- čítame vstupnú pásku zľava doprava
  - ak čítam na vstupe 0, posuniem sa doprava, na 1pp napíšeme 0 a posunieme sa doprava,
  - ak čítam na vstupe 1, posuniem sa doprava, na 1pp nepíšeme nič a stojíme,
- keď prídeme na dolár na vstupe začneme čítať vstupnú pásku sprava doľava
  - ak čítame na vstupe 0, posuniem sa doľava, na 1pp nerobíme nič a stojíme (stáť môžeme na 0, ale aj na cente – napr. pre slovo 0011),
  - ak čítame na vstupe 1, posuniem sa doľava, na 1pp:
    - ak je tam 0, vymažeme ju a posunieme sa doľava,
    - ak je tam už cent, slovo zamietame (v slove bolo viac jednotiek ako núl)
  - ak čítame na vstupe cent, stojíme a na 1pp:
    - ak je tam tiež cent, slovo akceptujeme (má rovnaký počet núl a jednotiek),
    - ak je tam nula, slovo zamietame (v slovo bolo viac núl ako jednotiek)

Teraz nám už len zostáva tento TS zostrojiť. Skúste ísť podľa návodu a následne porovnajte s tým mojím. Nezabudnite definovať celý TS – tj. aj vstupnú abecedu $\{0, 1\}$ (z čoho sa skladá vstupné slovo) a pracovnú abecedu $\{¢, \$, \sqcup, 0, 1, \sqcup\}$ (aké všetky znaky vieme prečítať). Zamyslite sa, čo sa stane s prázdnym slovom na vstupe. Bude akceptované a má byť akceptované?

Napadli vám aj iné možnosti ako tento príklad riešiť? Určite sa to dá pomocou dvojpáskového TS, kde na jednej páske budete mať nuly a na druhej jednotky a potom iba skontrolujete, či ich je rovnako veľa. Prípadne iné obmeny. No ale dá sa to vyriešiť aj tak, že po vstupnom slove prejdeme naozaj iba raz a budeme mať iba jednu pomocnú pásku? Áno, dá. Na začiatku sme zistili, že ak by sme si zapisovali na 1pp nuly a za každú jednotku by sme jednu nulu z 1pp vymazali, problematické by bolo, keby sme mali prázdnu

pásku a chceli by sme mazať. To môžeme vyriešiť napríklad tak, že keď je páska prázdna, ale mali by sme jeden symbol vymazať, napíšeme si na pásku napríklad $-$. Následne, ak ideme zapísať nulu ale už je tam mínus, toto mínus vymažeme. Tu si však treba dávať pozor na to, ako sa hýbe hlava na 1pp, kedy mažeme, kedy sa kam posúvame a kedy určite vieme, že môžeme akceptovať a kedy zamietať. Skúste si taký TS navrhnúť, prípadne sa môžete inšpirovať tým mojím.

Riešenie (b): ako budeme kontrolovať slová, v ktorých je trikrát viac núl ako jednotiek? Môžeme upraviť nejaký z postupov v predchádzajúcom príklade? Samozrejme. Tak skúsme najprv načrtnúť, ako by sme tento príklad riešili s jednopáskovým TS.

Najprv vstupné slovo čítame zľava doprava. Za každú prečítanú jednotku na 1pp zapíšeme tri jednotky. Pri prečítaní nuly sa iba posunieme doprava a na 1pp nerobíme nič. Keď narazíme na dolár, začneme slovo čítať sprava doľava. Teraz vždy, keď prečítame nulu, jednu jednotku z 1pp vymažeme (tiež vymazávame sprava doľava). Keďže sme si na začiatku strojnásobili počet jednotiek, slovo akceptujeme, ak sa na oboch páskach dostaneme naraz na cent. Ak bude 1pp prázdna skôr ako sme stihli prečítať celé vstupné slovo, slovo má viac núl ako tri krát počet jednotiek. Ak sa na vstupnej páske dostaneme na cent a na 1pp budú ešte jednotky, tak slovo bude mať menej núl ako tri krát počet jednotiek. V oboch týchto prípadoch zamietame.

Ako by vyzeral dvojpáskový TS? Na jednu pásku by sme si mohli napísať nuly, na druhú jednotky. A tu máte zas dve možnosti – buď každú jednotku zapíšete trikrát a následne iba skontrolujete, či sú slová na 1pp a 2pp rovnako dlhé. Alebo zapíšete iba reálne počty núl a jednotiek a pri následnej kontrole za každú jednotku na 2pp prejdete tri nuly na 1pp. Vyskúšajte si jeden z týchto spôsobov zostrojiť.

Riešenie (c): prepokladám, že už asi rovno viete, ako by ste riešili tento príklad. Najjednoduchšie by, samozrejme, bolo použiť tri pomocné pásky. Na jednej mať nuly, na druhej jednotky a na tretej dvojky. Ale vieme to vyriešiť aj s dvoma pracovnými páskami? Áno, ak trochu skombinujeme oba prístupy z predchádzajúcej úlohy.

Na 1pp si budeme písať nuly – za každú nulu na vstupe sem napíšeme jednu nulu. Na 2pp za každú jednotku na vstupe napíšeme dve jednotky. No a následne, keď už budeme na konci slova, budeme odzadu kontrolovať – ak prečítame dvojku, vymažeme jednu nulu z 1pp a jednu jednotku z 2pp. Ak prečítame na vstupe všetky dvojky a obe pracovné pásky budú prázdne, slovo akceptujeme. Vyskúšajte si tento TS skonštruovať.

3. príklad

Zostrojte dvojpáskový Turingov stroj, ktorý bude simulovať Stack (zásobník).

- na vstupnej páske dostane $\uparrow,\ a,\ b$, kde $\uparrow$ znamená POP, a symboly $a,\ b$ PUSH daných symbolov do zásobníka,
- na prvej pracovnej páske bude aktuálny stav zásobníka,
- na druhej pracovnej páske výstup z funkcie POP,
- TS skončí v akceptačnom stave, ak po vykonaní celého vstupu zostane zásobník prázdny.
- *Je na vás, čo spraví TS, ak je prázdny zásobník a zavolá sa POP. Môže to „vyhodiť error", teda presunúť sa do Reject-u, alebo iba vrátiť blank.*

Idea riešenia 3. príkladu

Príklad sa dá vyriešiť presne podľa zadania. Vždy, keď bude na vstupe symbol a alebo b, napíšeme ich na 1pp a posunieme sa na nej doprava. Ak tam bude $\uparrow$, nájdeme na 1pp posledný symbol, napíšeme ho na 2pp a z 1pp ho vymažeme. Keď prejdeme celé slovo na vstupnej páske, skontrolujeme, či je 1pp (zásobník) prázdna. Vyskúšajte, prípadne si pozrite moje riešenie.

Vedeli by sme vytvoriť TS, ktorý by akceptoval také slová nad abecedou $\{\uparrow, a, b\}$, že ich vykonaním zostane prázdny zásobník, aj bez toho, že by sme si zapisovali aktuálny stav zásobníka? Mohli by ste navrhnúť, že veď iba stačí zistiť, či je počet znakov a, b rovnaký ako počet $\uparrow$ v slove. To by ale nebola pravda napríklad pre slovo $\uparrow\uparrow ab$, pri ktorom by na konci zostalo v zásobníku ab, alebo by už rovno vyhodil error pri pokuse o POP z prázdneho zásobníka. V tomto prípade je dôležité poradie operácií, preto vhodné riešenie je práve simulovanie zásobníka (samozrejme, ak by sme iba zisťovali, či je slovo z jazyka, nepotrebovali by sme 2pp).

4. príklad

Zostrojte viacpáskový deterministický Turingov stroj, ktorý rozoznáva jazyk:

(a) $L_1 = \{a^{n^2} \mid n > 1, n \in N\}$

(b)* $L_2 = \{a^{2^n} \mid n > 1, n \in N\}$

(c)* $L_3 = \{a^n b^{3n} \mid n \geq 1, n \in N\}$

Poznámky k 4. príkladu

Idea riešenia 4(a): Ako by sme mohli rozoznávať slová, ktoré obsahujú počet áčok rovný druhej mocnine čísla väčšieho ako jedna? No vieme si mocniny predpočítať a potom iba skontrolovať, či na vstupe je slovo rovnakej dĺžky. A ako si ich predpočítame? Použijeme na to dve pomocné pásky – na prvej budeme mať n áčok, na druhej vždy vypočítame n^2. Vieme, že najmenšie slovo akceptované týmto TS je slovo $a^{2^2} = aaaa$, takže si ho na začiatku pripravíme na pásky.

Ako zmeníme a^{n^2} na $a^{(n+1)^2}$? Vypočítajme o koľko áčok viac bude mať dlhšie slovo. Tj. vypočítajme rozdiel $(n+1)^2 - n^2 = n^2 + 2n + 1 - n^2 = 2n + 1$. To znamená, že na 2pp iba dvakrát skopírujeme obsah 1pp, pridáme ešte jedno áčko a o jedno áčko zvýšime aj 1pp.

Vždy po vypočítaní druhej mocniny skontrolujeme, či nemáme rovnaký počet ako na vstupe. Ak je už na 2pp viac áčok ako na vstupe, slovo zamietneme (zjavne nie je druhou mocninou), ak je ich menej, spravíme druhú mocninu o jedna väčšieho čísla.

Vyskúšajte si moje riešenie. Viete zostrojiť aj iný postup? Skúste pre každý zistiť, koľkokrát budete musieť prejsť všetky pásky (tj. zistite zložitosť tohto algoritmu).

Idea riešenia 4(b): budete vedieť použiť podobný postup ako v časti (a)? Ako sa správajú mocniny dvojky? Bude vám stačiť aj menej pomocných pások?

Idea riešenia 4(c): Viete využiť podobný postup ako napríklad v príklade 2(b)? Tiež máte porovnať počty dvoch znakov. Teraz o čosi jednoduchšie, lebo tie znaky budú za sebou. Odporúčam si zapísať slová, ktoré teraz vyhovujú zadaniu.

Poznámky

Hlavný rozdiel medzi obyčajným TS a viacpáskovým TS je v tom, že pri viacpáskovom je vstupná páska READ-ONLY - začína ¢ a končí $. Pracovné pásky číslujeme od 1 a na začiatku sú prázdne (až na ¢) a čítacie hlavy na začiatku. Pre každý krok viacpáskového TS musíme definovať:

- čo čítame na vstupnej páske a kam sa posúvame;
- pre každú pracovnú: čo čítame, čo zapisujeme a kam sa posúvame.

Determinizmus pri viacpáskových TS značí, že ak v jednom stave prečítame určitú sadu symbolov na páskach, jednoznačne bude stroj vedieť, kam sa posunúť. Teda nie je povolené mať zároveň prechody ako napr.: $a, R \mid b, b, R \mid c, c, L$ a prechod $a, N \mid b, b, R \mid c, c, L$

Pri domácich úlohách a skúškach je dôležitá najmä jasná idea celého TS. Tj. postup, ako pracuje, čo kde sa má stať, kedy akceptuje, kedy zamieta... Takže popis algoritmu. Potom, ak sa aj pomýlite v prechodovej funkcii, je to menší problém, akoby ste napísali iba prechodovú funkciu a nenapísali myšlienku algoritmu. Opäť, nezabúdajte na vhodné opisy stavov či častí TS (na ktoré sa môžete odkazovať v popise), a nezabudnite spraviť definíciu TS (sedmica z prednášky). Prechodovú funkciu máte už priamo v diagrame, ak ho nakreslíte. Alebo ju môžete iba napísať.