Úloha 5.1

Michal Hitka

28. marca 2001

Zadanie

Zostrojte deterministický turingov stroj, ktorý ak dostane na vstup kód ľubovoľného turingovho stroja A, pretransformuje ho na kód turingovho stroja A' takého, že:

1. A' nemá definovaný krok zo žiadneho akceptačného stavu,
2. $L(A') = L(A)$.

Detailne zdôvodnite správnosť svojej konštrukcie.

Riešenie

Pomocné tvrdenia

Aby sa nám neskôr ľahšie postupovalo, zadefinujme si teraz pre naše účely pomocný pojem, a síce ekvivalentnosť kódov TM.

Definícia: Budeme hovoriť, že dva kódy turingovych strojov sú ekvivalentné, ak platí:

1. obidva kódy sú „korektné“,
2. časti kódu jedného TM, definujúce zakódovanie K, Σ, Γ, q_0 a F, sú zhodné so zodpovedajúcimi časťami kódu druhého turingovho stroja,
3. definície prechodovej δ-funkcie sa líšia jedine v krokoch z akceptačných stavov.

Vychádzajúc z uvedenej definície, tvrdím, že platí nasledovné tvrdenie:

Lema: Nech A, A' sú turingove stroje také, že ich kódy sú ekvivalentné. Potom platí $L(A) = L(A')$.

Dôkaz: Najprv sa pozrieme na inklúziu $L(A) \subseteq L(A')$. Nech teda $w \in L(A)$, potom ale existuje akceptujúci výpočet pre slovo w v tvare $(qw) \vdash^* (uq_Fv)$.

Teraz rozoberieme dva prípady. Ak v tomto výpočte nebol použitý krok z akceptačného stavu, potom je zrejme slovo w akceptované aj turingovym strojom A', pretože z definície ekvivalentnosti kódov TM vieme, že platí $K = K', \Sigma = \Sigma', \Gamma = \Gamma', F = F'$ a až na kroky z akceptačných stavov sú rovnako definované aj δ a δ' – keďže však výpočet slova w takéto kroky neobsahuje, je tento výpočet rovnako akceptačný aj v A'.

Ak naopak krok z akceptačného stavu v tomto výpočte bol použitý, potom sa výpočet pre w dá napísať aj v tvare $(qw) \vdash^* (u'q'_Fv') \vdash^* (uq_Fv)$, kde $q'_F \in F$. To však znamená, že aj výpočet $(qw) \vdash^* (u'q'_Fv')$ je akceptačným výpočtom pre slovo w v A. Stačí nám zvoliť prvú takúto významnú konfiguráciu a „useknutím“ zvyšku výpočtu dostaneme akceptačný výpočet, neobsahujúci kroky z akceptačného stavu, pričom už pred chvíľou sme ukázali, že tento je teda aj akceptačným výpočtom pre slovo w v A'.

Opačná inklúzia sa dokáže rovnako, len vymenením TM A a A'. $\surd$

Konštrukcia TM T

Teraz zostrojíme turingov stroj T nasledovne:

$$\begin{aligned} T &= (K, \Sigma, \Gamma, \delta, q_{init[0]}, F) \\ \Sigma &= \{0, 1, \#\} \\ \Gamma &= \{B, D, 1_a, 1_b\} \cup \Sigma \\ F &= \{q_{end}\} \end{aligned}$$

Definícia jeho prechodovej funkcie je uvedená nižšie, množina pracovných stavov je tiež zrejmá z definície δ.

Definícia δ-funkcie

Táto časť má na starosti iba na jeden prechod umiestniť znaky 1_a a 1_b (ak sú definované δ-funkcia aj akceptačné stavy).

$$\begin{array}{lcl}
\left.\begin{array}{lcl}
\delta(q_{init[i]}, 0) & = & \{(q_{init[i]}, 0, 1)\} \\
\delta(q_{init[i]}, 1) & = & \{(q_{init[i]}, 1, 1)\} \\
\delta(q_{init[i]}, \#) & = & \{(q'_{init[i]}, \#, 1)\} \\
\delta(q'_{init[i]}, 0) & = & \{(q_{init[i]}, 0, 1)\} \\
\delta(q'_{init[i]}, 1) & = & \{(q_{init[i]}, 1, 1)\}
\end{array}\right\} & \forall i \in \{0, \ldots, 4\} & \\
\delta(q'_{init[0]}, \#) = \{(q_{init[1]}, \#, 1)\} & & \\
\delta(q'_{init[1]}, \#) = \{(q_{init[2]}, \#, 1)\} & & \\
\delta(q'_{init[2]}, \#) = \{(q_{init[a]}, \#, 1)\} & & \\
\delta(q_{init[a]}, 1) = \{(q_{init[3]}, 1_a, 1)\} & & \\
\delta(q_{init[a]}, \#) = \{(q_{end}, \#, 0)\} & & \\
\delta(q'_{init[3]}, \#) = \{(q_{init[4]}, \#, 1)\} & & \\
\delta(q'_{init[4]}, \#) = \{(q_{init[b]}, \#, 1)\} & & \\
\delta(q_{init[b]}, 1) = \{(q_{check}, 1_b, 0)\} & & \\
\delta(q_{init[b]}, B) = \{(q_{finalize}, B, 0)\} & &
\end{array} \tag{1}$$

Tu pracujeme v pravej časti – nájdeme ďalší symbol na porovnanie a potom ho necháme porovnať... Najprv sa posúvame doprava až po 1_b. Túto prepíšeme naspäť na 1 a 1_b posunieme ďalej (ak je ďalej 0, vrátime sa doľava a označíme poslednú 1 znovu ako 1_b). Potom ideme doľava porovnať nájdený symbol.

$$\begin{array}{lcl}
\delta(q_{check}, x) & = & \{(q_{check}, x, 1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q_{check}, 1_b) & = & \{(q'_{check}, 1, 1)\} \\
\delta(q'_{check}, 1) & = & \{(q_{check[1]}, 1_b, -1)\} \\
\delta(q'_{check}, 0) & = & \{(q_{check[0]seek}, 0, -1)\} \\
\delta(q_{check[0]seek}, 1) & = & \{(q_{check[0]}, 1_b, -1)\}
\end{array} \tag{2}$$

Ak sme prečítali 1, ideme doľava až po 1_a. Túto prepíšeme znovu na 1 a posunieme sa doprava. Ak tu nájdeme 1, mohol by to byť krok z akceptačného stavu, a ideme porovnávať ďalší symbol. Inak budeme pokračovať v hľadaní na ďalšej časti definície δ-funkcie (q_{retry*}).

$$\begin{array}{lcl}
\delta(q_{check[1]}, x) & = & \{(q_{check[1]}, x, -1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q_{check[1]}, 1_a) & = & \{(q'_{check[1]}, 1, 1)\} \\
\delta(q'_{check[1]}, 0) & = & \{(q_{retry[1a]seek}, 0, 1)\} \\
\delta(q'_{check[1]}, 1) & = & \{(q_{check}, 1_a, 1)\}
\end{array} \tag{3}$$

Ak sme prečítali 0, ideme doľava až po 1_a. Túto prepíšeme znovu na 1 a posunieme sa doprava. Ak tu nájdeme 0, ide o krok z akceptačného stavu, a teda ho vymažeme. Inak budeme pokračovať v hľadaní na ďalšej časti definície δ-funkcie (q_{retry*}).

$$\begin{array}{lcl}
\delta(q_{check[0]}, x) & = & \{(q_{check[0]}, x, -1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q_{check[0]}, 1_a) & = & \{(q'_{check[0]}, 1, 1)\} \\
\delta(q'_{check[0]}, 0) & = & \{(q_{deleteseek}, 0, 1)\} \\
\delta(q'_{check[0]}, 1) & = & \{(q_{retry[1a]seek}, 1, 1)\}
\end{array} \tag{4}$$

Oddeľovací symbol # na začiatku pravidla prepíšeme na špeciálny symbol D.

$$\begin{array}{lcl}
\delta(q_{deleteseek}, 0) & = & \{q_{deleteseek}, 0, -1)\} \\
\delta(q_{deleteseek}, 1) & = & \{q_{deleteseek}, 1, -1)\} \\
\delta(q_{deleteseek}, \#) & = & \{q_{delete}, D, -1)\}
\end{array} \tag{5}$$

Teraz budeme mazať tak, že všetko od prvého symbolu B pred vetnou formou po symbol D (vrátane) posunieme o jedno políčko doprava, čím premažeme znak napravo od symbolu D (ak tam už nie je #).

$$\begin{array}{lcl}
\delta(q_{delete}, x) & = & \{q_{delete}, x, -1)\} \; \forall x \in \{0, 1, \#\} \\
\delta(q_{delete}, B) & = & \{q_{deleteshift[B]}, B, 1)\} \\
\delta(q_{deleteshift[x]}, 0) & = & \{(q_{deleteshift[0]}, x, 1)\} \quad \forall x \in \{0, 1, B\} \\
\delta(q_{deleteshift[x]}, 1) & = & \{(q_{deleteshift[1]}, x, 1)\} \quad \forall x \in \{0, 1, B\} \\
\delta(q_{deleteshift[x]}, D) & = & \{(q_{deleteshift[D]}, x, 1)\} \quad \forall x \in \{0, 1, B\} \\
\delta(q_{deleteshift[D]}, 0) & = & \{(q_{delete}, D, -1)\} \\
\delta(q_{deleteshift[D]}, 1) & = & \{(q_{delete}, D, -1)\} \\
\delta(q_{deleteshift[D]}, \#) & = & \{(q_{deletecheck}, \#, -1)\}
\end{array} \tag{6}$$

Po vymazaní overíme, či nevznikla sekvencia ###. Ak áno, pridáme ďalší znak # a popri posúvaní doprava upravíme 1_b na 1, vďaka čomu môžeme skončiť (δ už nemá ďalšie pravidlá a 1_a sa už na páske nenachádza, takže ho netreba odstraňovať).

$$\begin{array}{lcl}
\delta(q_{deletecheck}, 0) & = & \{(q_{retry[1a]seek}, 0, 1)\} \\
\delta(q_{deletecheck}, \#) & = & \{(q'_{deletecheck}, \#, 1)\} \\
\delta(q'_{deletecheck}, \#) & = & \{(q''_{deletecheck}, \#, 1)\} \\
\delta(q''_{deletecheck}, \#) & = & \{(q_{deleteadd[\#]}, \#, 1)\} \\
\delta(q''_{deletecheck}, 1) & = & \{(q_{retry[1a]seek}, 1, -1)\} \\
\delta(q_{deleteadd[x]}, y) & = & \{(q_{deleteadd[y]}, x, 1)\} \quad \forall x, y \in \{0, 1, \#\} \\
\delta(q_{deleteadd[x]}, 1_b) & = & \{(q_{deleteadd[1]}, x, 1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q_{deleteadd[x]}, B) & = & \{(q_{end}, x, 0)\} \quad \forall x \in \{0, 1, \#\}
\end{array} \tag{7}$$

Tu sa iba posunieme za najbližší # (čiže na ďalšie pravidlo) a zapíšeme 1_a.

$$\begin{aligned}
\delta(q_{retry[1a]seek}, 0) &= \{q_{retry[1a]seek}, 0, 1)\} \\
\delta(q_{retry[1a]seek}, 1) &= \{q_{retry[1a]seek}, 1, 1)\} \\
\delta(q_{retry[1a]seek}, \#) &= \{q_{retry[1a]}, \#, 1)\} \\
\delta(q_{retry[1a]}, 1) &= \{q_{retry[1b]seek}, 1_a, 1\} \\
\delta(q_{retry[1a]}, \#) &= \{q_{next[1a]seek}, \#, -1\}
\end{aligned} \tag{8}$$

Tu prejdeme doprava až po symbol 1_b a vrátime ho naspäť na začiatok kódu stavu, aby sme mohli znovu začať porovnávať.

$$\begin{aligned}
\delta(q_{retry[1b]seek}, x) &= \{q_{retry[1b]seek}, x, 1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q_{retry[1b]seek}, 1_b) &= \{q'_{retry[1b]seek}, 1, -1)\} \\
\delta(q'_{retry[1b]seek}, 1) &= \{q'_{retry[1b]seek}, 1, -1)\} \\
\delta(q'_{retry[1b]seek}, \#) &= \{q_{retry[1b]}, \#, 1\} \\
\delta(q_{retry[1b]}, 1) &= \{q_{check}, 1_b, 0\}
\end{aligned} \tag{9}$$

Posunieme sa doľava až po ## a namiesto prvej 1 umiestnime 1_a. Potom prejdeme doprava a 1_b presunieme za najbližší symbol # (ak sme na konci vstupu, úspešne sme skončili).

$$\begin{aligned}
\delta(q_{next[1a]seek}, 0) &= \{(q_{next[1a]seek}, 0, -1)\} \\
\delta(q_{next[1a]seek}, 1) &= \{(q_{next[1a]seek}, 1, -1)\} \\
\delta(q_{next[1a]seek}, \#) &= \{(q'_{next[1a]seek}, \#, -1)\} \\
\delta(q'_{next[1a]seek}, 0) &= \{(q_{next[1a]seek}, 0, -1)\} \\
\delta(q'_{next[1a]seek}, 1) &= \{(q_{next[1a]seek}, 1, -1)\} \\
\delta(q'_{next[1a]seek}, \#) &= \{(q''_{next[1a]seek}, \#, 1)\} \\
\delta(q''_{next[1a]seek}, \#) &= \{(q_{next[1a]}, \#, 1)\} \\
\delta(q_{next[1a]}, 1) &= \{(q_{next[1b]seek}, 1_a, 1)\} \\
\delta(q_{next[1b]seek}, x) &= \{(q_{next[1b]seek}, x, 1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q_{next[1b]seek}, 1_b) &= \{(q'_{next[1b]seek}, 1, 1)\} \\
\delta(q'_{next[1b]seek}, 0) &= \{(q'_{next[1b]seek}, 0, 1)\} \\
\delta(q'_{next[1b]seek}, 1) &= \{(q'_{next[1b]seek}, 1, 1)\} \\
\delta(q'_{next[1b]seek}, \#) &= \{(q_{next[1b]}, \#, 1)\} \\
\delta(q_{next[1b]}, 1) &= \{(q_{check}, 1_b, 0)\} \\
\delta(q_{next[1b]}, B) &= \{(q_{finalize}, B, 0)\}
\end{aligned} \tag{10}$$

Poupratujeme – iba zameníme 1_a a 1_b naspäť na 1 (ak sú zapísané na páske).

$$\begin{aligned}
\delta(q_{finalize}, x) &= \{(q_{finalize}, x, 1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q_{finalize}, 1_b) &= \{(q'_{finalize}, 1, -1)\} \\
\delta(q_{finalize}, B) &= \{(q'_{finalize}, B, -1)\} \\
\delta(q'_{finalize}, x) &= \{(q'_{finalize}, x, -1)\} \quad \forall x \in \{0, 1, \#\} \\
\delta(q'_{finalize}, 1_a) &= \{(q_{end}, 1, 0)\} \\
\delta(q'_{finalize}, B) &= \{(q_{end}, B, 0)\}
\end{aligned} \tag{11}$$

Rozbor δ-funkcie

Teraz sa pozrieme na prechodovú funkciu turingovho stroja T bližšie. Pokúsime jej rozborom dokázať, že ak na vstup TM T dáme korektný kód TM A, potom po skončení výpočtu na páske zostane ekvivalentný kód TM A' taký, že A' nemá definovaný krok zo žiadneho akceptačného stavu.

To už vlastne bude znamenať, že sme úspešne zavŕšili naše snaženie, pretože z lemy už priamo vyplynie $L(A) = L(A')$, čím bude dokázané všetko, čo sme dokázať chceli.

Procedúra init (1)

Korektný kód TM sa dá jednoznačne rozdeliť na 6 častí oddelených ##. Nás teraz zaujímajú iba časti týkajúce sa δ-funkcie a množiny akceptačných stavov, t.j. štvrtá a šiesta časť. Chceme prvé symboly 1 v týchto častiach (sú to zrejme podľa použitého kódovania prvé symboly v týchto častiach vôbec za predpokladu, že časti sú neprázdne) označiť 1_a, respektíve 1_b. Na toto nám slúži procedúra **init** (1), ktorá sa „spustí" na začiatku výpočtu (počiatočný stav je $q_{init[0]}$).

Počítame v stave počet sekvencií ## a postupujeme doprava. Po tretej takejto sekvencii (a teda na začiatku štvrtej časti) sme v stave $q'_{init[2]}$ a ak príde 1, označíme ju ako 1_a, inak sme skončili, pretože alebo nie je kód TM A korektný, alebo nie je definovaná prechodová funkcia, takže netreba na nej nič meniť.

Podobne po piatej sekvencii ak nájdeme znak 1, označíme ho ako 1_b, inak môžeme skončiť, lebo znovu alebo kód nie je korektný, alebo nemáme definovaný akceptačný stav, čiže nemôžu byť ani definované kroky z akceptačného stavu – a teda znovu stačí po sebe upratať a skončiť.

Na základe definície korektného kódu TM teda zrejme ak máme korektný kód TM s neprázdnou množinou akceptačných stavov a s neprázdnou definíciou δ-funkcie, potom po (jedinom) vykonaní tejto procedúry na začiatku oboch častí budú symboly 1_a, resp. 1_b, hlava bude na symbole 1_b, nič iné na páske sa nezmení a spustí sa procedúra **check** (2).

Procedúra check (2, 3, 4)

K procedúre **check** (2) sa môžeme dostať vo viacerých prípadoch: ak sme pripravení na prvé porovnávanie (1), alebo ak sme ešte neskončili porovnávanie a treba pokračovať (3), alebo ak sme sa posunuli na ďalšie pravidlo δ-funkcie a začiatok kódu porovnávaného akceptačného stavu a máme začať porovnávať (9), alebo sme sa presunuli na ďalší akceptačný stav a začíname porovnávať s pravidlami δ-funkcie odznova (10).

Indukciou na dĺžku porovnaného kódu porovnávaného akceptačného stavu by sme mohli poľahky dokázať, že ak sa kód porovnávaného stavu zhoduje so stavom v ľavej strane δ-funkcie, potom sa spustí procedúra **delete** (5), inak sa spustí procedúra **retry** (8).

Dôležité si je všimnúť pár faktov. Vždy pri volaní **check** (2) je hlava niekde medzi 1_a a 1_b. Symbol 1_b sa presúva postupne až na koniec skupiny jednotiek a keď narazíme na 0, zostane na mieste. Symbol 1_a však môže aj zmiznúť na dlhší čas, a to práve vtedy, keď voláme procedúry **delete** (5) a **retry** (8). Otázka je, či sa potom tento symbol dostane na správne miesto, na to sa však pozrieme až ďalej.

Procedúra delete (5, 6, 7)

Túto procedúru môžeme rozdeliť na tri časti. Najprv sa posunieme na začiatok pravidla (5). Potom nasleduje samotné vymazanie (6) a overenie, či sme nevymazali posledné pravidlo prechodovej funkcie (7).

Tu by sa nám tiež bez väčších technických obtaží podarilo indukciou na počet vymazaných symbolov dokázať, že ak vieme vymazať k symbolov z pravidla prechodovej funkcie (aj s ľavým oddeľovačom #), potom sa nám podarí vymazať aj $k+1$ symbolov, ak má kód vymazávaného pravidla dĺžku aspoň $k+1$.

Toto však zahŕňa iba prvé dve časti. Zostáva nám ešte ukázať, že ak sme vymazali posledné pravidlo, čím sme narušili korektnosť kódu (pretože sme vymazali naviac jeden symbol #, ktorý však neoddeľoval pravidlá, ale definíciu δ-funkcie od definície pracovnej abecedy), napravíme to a skončíme (už nie je čo porovnávať), inak spustíme procedúru **retry** (8), ktorá mimochodom zabezpečí doplnenie symbolu 1_a.

Keď sme skončili s vymazávaním v druhej časti, hlava zostala stáť na #, ktorý predtým ukončoval vymazané pravidlo. Ak je teda teraz vľavo od neho 0, končí tu predchádzajúce pravidlo. Ukončené je našim #. Môžeme sa tváriť, že sme práve neúspešne skončili porovnávanie tohto predchádzajúceho pravidla a ako v (3) teda spustíme procedúru **retry** (8).

Ak je však teraz vľavo od nášho # ďalší #, znamená to, že sme na začiatku definície δ a musíme sa pozrieť za náš # (čiže sa posunieme o dve políčka doprava). Ak sa tu nachádza 1, všetko je v poriadku (iba sa posunieme znovu na náš # doľava a urobíme to isté čo v minulom prípade), ak tu je však tretí #, vymazali sme posledné pravidlo a musíme pridať jeden krížik. Celý zvyšok pásky posunieme doprava a môžeme skončiť (popri presúvaní dáme preč 1_b).

Procedúra retry (8, 9)

Táto procedúra sa volá, ak sa pri porovnávaní stavov zistí rozdiel (3, 4), alebo ak sa po vymazaní pravidla prechádza na ďalšie pravidlo (7) – tu sa však iba simuluje predchádzajúci prípad, a keďže to nemá nijaký priamy dosah na naše úvahy, môžeme tieto prípady zahrnúť do prvej kategórie.

Vždy platí, že sa nachádzame na konci jedného pravidla a potrebujeme umiestniť symbol 1_a na začiatok ďalšieho pravidla. Jednoducho sa posúvame až po najbližší # a ak je za ním ďalší #, spustíme procedúru **next** (10). Ak však nasleduje 1, označíme ju ako 1_a. Potrebujeme ešte presunúť 1_b na začiatok kódu porovnávaného stavu. Po presunutí sa spustí procedúra **check** (2) a začne sa stav porovnávať s ďalším pravidlom.

Procedúra next (10)

Je volaná iba z **retry** (8), keď sa pri posúvaní naďalšie pravidlo δ narazí na sekvenciu ##. Potom sa hlava posunie doľava a zavolá sa táto procedúra. Táto tiež hľadá (posúvajúc sa doľava) sekvenciu ##, a teda sa zastaví na začiatku definície δ (z koncového oddeľovača δ prečíta už iba #, preto pokračuje ďalej). Následne prvý symbol vpravo od ## (čo musí byť 1) sa označí ako 1_a. Potom sa vpravo nájde 1_b a presunie sa za doprava na ďalší stav. Ak tam už ďalší stav nie je, môžeme skončiť, lebo sme už otestovali všetky akceptačné stavy a zodpovedajúce pravidlá vymazali.

Procedúra finalize (11)

Zavolá sa, ak treba upratať pásku, čiže odstrániť pracovné symboly 1_a a 1_b: postupujeme doprava, kým nenájdeme koniec pásky alebo 1_b (ktorý nahradíme za 1), potomsa otočíme a ideme doľava, kým nenájdeme koniec pásky alebo 1_a (nahradíme za 1). Nakoniec akceptujeme. Uvedomme si, že nikdy sa hlava nedostane doprava za symbol 1_b. Symbol D netreba upratať, zmizne po skončení vymazávania, vďaka čomu ak sme na vstup dostali korektný kód TM A v abecede $\{0, 1, \#\}$, dostaneme korektný kód TM A' v tej istej abecede.