Kapitola: Registrové stroje

V literatúre existujú mnohé variácie registrových strojov: niektoré len rozpoznávajú jazyk, niektoré počítajú (čiastočnú) funkciu na prirodzených číslach, líšia sa povolenou sadou inštrukcií a formálnymi detailmi, ale v princípe ide stále o jeden a ten istý model. My si ukážeme dva takéto formalizmy.

1.1 Dvojsmerný počítadlový konečný automat

Dvojsmerný konečný automat s n počítadlami je automat s read-only vstupnou páskou a n premennými, pričom každá premenná môže obsahovať ľubovoľné nezáporné celé číslo. Nasledujúci krok výpočtu pre tento automat je určený jeho stavom, práve čítaným písmenom vstupu a tým, ktoré z premenných obsahujú nulu a ktoré nie. V rámci jedného kroku výpočtu automat môže zmeniť stav, posunúť čítaciu hlavu a o 1 zvýšiť/znížiť hodnotu každej premennej.

Formálne, deterministický dvojsmerný konečný automat s n počítadlami je šestica $(n, K, \Sigma, \delta, q_0, F)$, kde K je konečná množina stavov, Σ je konečná vstupná abeceda taká, že $\mathbf{L}, \mathbf{R} \notin \Sigma$, $\delta : K \times (\Sigma \cup \{\mathbf{L}, \mathbf{R}\}) \times \{0,1\}^n \to K \times \{-1,0,1\} \times \{-1,0,1\}^n$ je prechodová funkcia[1], q_0 je začiatočný stav a F je množina akceptačných stavov.

Konfigurácia takéhoto automatu je prvok z $\Sigma^* \times K \times \mathbb{N} \times \mathbb{N}^n$, kde jednotlivé zložky predstavujú slovo, na ktorom prebieha výpočet, aktuálny stav, polohu čítacej hlavy a obsah jednotlivých premenných.

Formálnu definíciu relácie $\vdash$ kroku výpočtu si čitateľ ľahko doplní.

Jazyk rozpoznávaný takýmto automatom je množina:

$$\left\{ w \;\middle|\; \exists f \in F : \; \exists a, b_1, \ldots, b_n \in \mathbb{N} : \; (w, q_0, 1, \underbrace{0, \ldots, 0}_{n}) \vdash^* (w, f, a, b_1, \ldots, b_n) \right\}$$

1.1.1 Výpočtová sila modelu

Zjavne pre $n = 0$ dostávame presne klasické deterministické dvojsmerné konečné automaty, o ktorých je známe, že trieda jazykov, ktoré vedia rozpoznávať, je práve trieda regulárnych jazykov.

Pre $n = 1$ máme k dispozícii jedno počítadlo. S jedným počítadlom si kúsok pomôžeme, ale ešte to nestačí. Konkrétny výsledok (Ďuriš, Galil: Fooling a two way automaton or one pushdown store is better than one counter for two way machines, TCS 21 (1982) 39-53): Jazyk

$$\{x_0 \# x_1 \# \ldots x_k \# \mid k \geq 1 \;\wedge\; \forall i : x_i \in \{0,1\}^* \;\wedge\; \exists i > 0 : x_0 = x_i\}$$

sa nedá rozpoznať dvojsmerným deterministickým automatom s jedným počítadlom. Na druhej strane, tento jazyk sa dá rozpoznať deterministickým automatom so zásobníkom: naplníme doň $x_1 \# \ldots x_k \#$ a potom sa vrátime na začiatok slova a postupne každé x_i porovnáme s x_0.

Ako sme na tom zdola? Určite vieme aj bez použitia počítadla rozpoznávať hocijaký regulárny jazyk. Ľahučké cvičenie je rozpoznávať jazyk $\{a^n b^n \mid n \geq 0\}$. O trošku ťažšie, ale stále ešte sa dá: jazyk všetkých palindrómov nad danou abecedou.

Ale tam hranica nekončí. Dá sa napríklad rozpoznať aj jazyky $\{0^n 1^{n^2} \mid n \geq 1\}$ a $\{0^n 1^{2^n} \mid n \geq 1\}$, ktoré nie sú ani len bezkontextové, viď http://portal.acm.org/citation.cfm?id=193820.193835.

V nasledujúcich častiach ukážeme dnes už klasický Minského výsledok: už pre $n = 2$ sú počítadlové automaty schopné simulovať Turingove stroje.

1.1.2 Zásobník pomocou dvoch počítadiel

Keď máme k dispozícii dve počítadlá, vieme pomocou nich simulovať zásobník s ľubovoľne veľkou abecedou. (Veľkosť abecedy samozrejme musí byť vopred známa konštanta.)

Ako na to? Predpokladajme, že chceme simulovať zásobník s z-prvkovou abecedou. Bez ujmy na všeobecnosti nech je táto abeceda množina $\{1, \ldots, z\}$. Na konkrétny obsah zásobníka sa teraz môžeme dívať ako na konkrétne číslo v sústave so základom $z+1$. Znak na vrchu zásobníka bude zodpovedať najmenej významnej cifre, znak pod

[1] Navyše od δ požadujeme, aby sa z ľavej „zarážky“ $\mathbf{L}$ automat nemohol pohnúť doľava ani z pravej doprava.

ním druhej najmenej významnej, atď. (Vďaka tomu, že sme v našej abecede nepoužili nulu, je takto definované zobrazenie z obsahov zásobníka do prirodzených čísel skutočne injektívne.)

Číslo predstavujúce obsah zásobníka budeme mať uložené v jednom z našich dvoch počítadiel. To druhé budeme používať ako „pomocnú premennú“, keď budeme chcieť obsah zásobníka meniť.

Potrebujeme vedieť odsimulovať dve operácie: výber vrchnej hodnoty zo zásobníka a vloženie novej hodnoty na vrch zásobníka. Prvá operácia zodpovedá deleniu $z+1$ so zvyškom: ak číslo Z predstavuje neprázdny zásobník, tak $Z \bmod (z+1)$ je znak na jeho vrchu a $\lfloor Z/(z+1) \rfloor$ je číslo predstavujúce obsah zásobníka po odobratí tohto znaku. Vkladanie novej hodnoty je inverzný proces – teda potrebujeme číslo predstavujúce starý obsah zásobníka najskôr vynásobiť $z+1$ a následne k nemu pripočítať novú hodnotu.

Obe tieto veci vieme v našom modeli robiť. Delenie so zvyškom vyzerá nasledovne: Ak treba, vynulujeme pomocné počítadlo. Ďalej si spravíme cyklus tvorený $z+1$ pomocnými stavmi. V tomto cykle znižujeme obsah počítadla predstavujúceho zásobník, až kým neklesne na nulu, a zakaždým, keď sa vrátime do stavu, kde sme začínali, zvýšime o 1 hodnotu pomocného počítadla. Keď nám prvé počítadlo klesne na nulu, v druhom máme celú časť podielu a aktuálny stav zodpovedá zvyšku – teda symbolu, ktorý sme práve vybrali zo zásobníka. Na záver už len „presypeme“ obsah druhého počítadla do prvého: druhé znižujeme a zároveň prvé zvyšujeme, až kým druhé neklesne na nulu.

1.1.3 Potenciálne nekonečná páska pomocou dvoch zásobníkov

Pomocou dvoch zásobníkov si ľahko vieme pamätať pásku Turingovho stroja: Zoberieme jej neprázdnu časť a rozstrihneme ju na mieste, kde sa nachádza jeho hlava. Ľavú časť uložíme do jedného zásobníka a pravú do druhého, tak aby na vrchu zásobníka ležali znaky, ktoré sa nachádzajú najbližšie k polohe hlavy. Všetky zmeny pásky sa teda dejú lokálne, na vrchu zásobníkov. Pohyb hlavy vieme simulovať vhodným presýpaním znakov z jedného zásobníka do druhého.

1.1.4 Turingov stroj pomocou dvoch počítadiel

Už by sme vedeli simulovať Turingov stroj pomocou štyroch počítadiel: totiž pomocou štyroch počítadiel vieme simulovať dva zásobníky, a pomocou dvoch zásobníkov vieme simulovať Turingov stroj.

(Rozmyslite si, čo presne bude náš počítadlový automat musieť spraviť, skôr než bude môcť začať simuláciu konkrétneho Turingovho stroja. Tiež si rozmyslite, že by nám tie počítadlá stačili aj tri – ale ako uvidíme o chvíľu, je to vlastne jedno.)

Teraz prichádza posledný krok: ukážeme, ako štyri počítadlá (alebo ľubovoľný iný konečný, vopred známy počet) simulovať pomocou dvoch.

Na to použijeme klasický, Leibnizom inšpirovaný trik: ak mali pôvodné 4 počítadlá hodnoty a, b, c, d, my budeme mať namiesto nich jedno s hodnotou $H = 2^a 3^b 5^c 7^d$. Ak máme k dispozícii ešte jedno pomocné počítadlo, vieme simulovať všetky operácie, ktoré potrebujeme: napr. zníženie a o 1 odsimulujeme tak, že H vydelíme dvoma.

Dostávame teda želané tvrdenie: dvojsmerné deterministické konečné automaty s dvoma počítadlami sú už rovnako silné ako Turingove stroje.

1.1.5 Zamyslenie na záver

Ako je to s jednosmernými deterministickými počítadlovými automatmi s 1, 2, 3, 4 počítadlami?

1.2 Registrový stroj počítajúci funkciu

Program pre náš registrový stroj je konečná postupnosť inštrukcií. Vykonávanie programu začína prvou inštrukciou a končí v okamihu, kedy by sa mala vykonať neexistujúca inštrukcia.

Náš stroj pozná tri typy inštrukcií:

- `INC x` – zväčši hodnotu v registri `x`
- `DEC x` – zmenši hodnotu v registri `x`

- `ZERO x y [z]` – ak je v registri `x` nula, pokračuj inštrukciou `y` [inak pokračuj inštrukciou `z`]

Výpočet stroja začína v konfigurácii v ktorej je v registri 0 uložený vstup a ostatné registre obsahujú nuly. Za výstup považujeme hodnotu v registri 1 v okamihu kedy výpočet skončí. (Toto pre niektoré vstupy a programy samozrejme nemusí nikdy nastať.)

1.2.1 Príklad

Nasledujúci program počíta funkciu $f(x) = x \bmod 3$. V cykle tvorenom inštrukciami 2-7 znižuje hodnotu vstupu až kým sa tá nedostane na nulu. Podľa okamihu, kedy toto nastane, program nastaví hodnotu v registri 1.

```
1. ZERO 0 10
2. DEC 0
3. ZERO 0 9
4. DEC 0
5. ZERO 0 8
6. DEC 0
7. ZERO 0 10 2
8. INC 1
9. INC 1
```