Týden 9

Upozornění: Jelikož 13.4.2009 je velikonoční pondělí, bude čtvrteční přednáška v tomto (9.) týdnu standardní přednáškou, nikoli tou zamýšlenou pro hlubší zájemce. (Cvičení v 10. týdnu bude na ni navazovat.)

Přednáška:pondělí+čtvrtek

Turingovy stroje, (výpočetní) problémy

Dokončení minulé přednášky.

Simulace mezi variantami Turingových strojů

Zavedli jsme přirozenou definici dvoupáskového Turingova stroje (2P-TS) jako struktury $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, kde

$$\delta : (Q - F) \times \Gamma \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\} \times \Gamma \times \{-1, 0, +1\},$$

a zkonstruovali jsme takový stroj, který řeší problém „Zdvojení slova“. Sestrojili jsme přitom následující množinu instrukcí.

$(q_0, x, \square) \to (q_0, x, +1, x, +1)$ pro $x \in \{a, b\}$
$(q_0, \square, \square) \to (q_1, \square, -1, \square, 0)$
$(q_1, x, \square) \to (q_1, x, -1, \square, 0)$ pro $x \in \{a, b\}$
$(q_1, \square, \square) \to (q_2, \square, +1, \square, 0)$
$(q_2, x, \square) \to (q_2, x, +1, x, +1)$ pro $x \in \{a, b\}$
$(q_2, \square, \square) \to (q_{halt}, \square, 0, \square, 0)$

Pak jsme si ujasnili, jak lze obecný dvoupáskový Turingův stroj (2P-TS) M simulovat jednopáskovým dvouhlavým Turingovým strojem (1P-2H-TS) M'. Stroje M, M' mají tedy stejné *(vstupně/výstupní) chování*, tj. realizují tutéž (částečnou) funkci $f_M = f_{M'}$.

> (Idea spočívá v použití „dvoustopé pásky“, tedy místo abecedy Γ stroje M má stroj M' abecedu $\Gamma' = \Gamma \cup (\Gamma \times \Gamma)$. První hlava mění jen „horní stopu“, druhá jen „dolní stopu“. Přitom se musí ošetřit případný zapisovací konflikt, když hlavy stojí na stejném políčku pásky.)

Navázali jsme náčrtem simulace obecného 1P-2H-TS M' standardním strojem, tedy jednopáskovým jednohlavým Turingovým strojem.

> (Ten si na pásce musí označovat místa, kde by stály simulované hlavy, a „trochu se naběhá“.)

Model RAM

Ve studijním textu je detailně popsán model RAM, který je realističtějším modelem počítače než Turingův stroj. Je tam uveden i konkrétní RAM (tedy konkrétní program = posloupnost instrukcí), který řeší následující problém.

VSTUP: Neprázdná posloupnost kladných celých čísel ukončená nulou.

VÝSTUP: Odchylky jednotlivých čísel od aritmetického průměru zadané posloupnosti zaokrouhleného dolů.

Činnost zmíněného RAMu je také přiblížena jednou z animací.
Každému by ovšem mělo být jasné, že podívat se na definici, příklad a animaci RAMu je něco zcela jiného než konkrétní RAM sestavit. Teprve „ošahání si“ jednotlivých instrukcí RAMu při konkrétním programování nás může přivést ke skutečnému porozumění, pasivní pohled na animaci to sotva může nahradit.
Proto jsme se pustili do společného sestrojení RAMu řešícího výše zmíněný problém. (Uvažovali jsme celá nenulová čísla místo kladných, ale to je nepodstatné.) Uvědomili jsme si, že nejdříve musíme pochopitelně navrhnout algoritmus, který pak budeme programovat. Ten jsme popsali následujícím pseudokódem pascalského typu.

```
(* variables: *)
A: array [1.. ] of integer;
cislo, pocet, soucet, prumer: integer;

pocet:=0; soucet:=0; read(cislo);

while cislo ≠ 0 do
{ pocet:=pocet+1; A[pocet]:=cislo; soucet:=soucet+cislo; read(cislo) };

prumer:= soucet div pocet; (* celociselne deleni *)

for i:=1 to pocet do { write(A[i]-prumer) };
```

Tento pseudokód jsme relativně přímočaře postupně přepsali pomocí instrukcí RAMu.
Pro jednoduché proměnné jsme si vyhradili paměťové buňky s následujícími adresami

```
cislo ... 2
pocet ... 3
soucet ... 4
prumer ... 5
```

Poli A jsme přiřadili základní adresu 8 (prvek A[1] jsme ukládali do buňky 9, prvek A[2] do buňky 10, atd.).
Dospěli jsme k programu

```
 1  READ
 2  JZERO 13
 3  STORE 2          16  LOAD =1
 4  LOAD 3
 5  ADD =1           17  STORE 1
 6  STORE 3          18  SUB 3
 7  STORE 1          19  JGTZ 26
 8  LOAD 2           20  LOAD *8
 9  STORE *8         21  SUB 5
10  ADD 4            22  WRITE
11  STORE 4          23  LOAD 1
12  JUMP 1           24  ADD =1
                     25  JUMP 17
13  LOAD 4
14  DIV 3            26  HALT
15  STORE 5
```

Přitom jsme si detailně promysleli význam všech instrukcí (částečně jsme si je „animovali“ na konkrétním příkladu), využití pracovního registru 0 a indexregistru 1 a rozdíly v argumentech typu “$= i$”, “i” a “$*i$” (kde i je zápis celého čísla).

Samozřejmě je vhodné si program detailně okomentovat, aby bylo jasné, že se opravdu jedná o (jednu možnost) přepsání uvedeného pseudokódu do instrukcí RAMu. (Kdo se neúčastnil tvorby RAMu na přednášce, měl by si nezávisle udělat sám; pohled na hotovou věc moc nepomůže, jak jsme již zmínili dříve.)

Simulace mezi RAMy a Turingovými stroji

Shodli jsme se, že je vcelku jasné, jak lze jakýkoli Turingův stroj s jednostranně nekonečnou páskou přímočaře simulovat RAMem.

Naopak je to technicky komplikovanější, ale myšlenkově to pro programátory zase není tak náročné – simulaci RAMu vícepáskovým Turingovým strojem ilustruje jedna z animací.

Rozhodnutelnost a nerozhodnutelnost problémů

Nejprve jsme si důkladně připomněli, co to je (v našem kontextu) *problém*. Uvědomili jsme si, že každému problému P je jednoznačně přiřazena (většinou nekonečná) „*tabulka*“ T_P s dvěma sloupci: v prvním sloupci jsou v jednotlivých řádcích vyjmenovány všechny (přípustné) vstupy (neboli instance) problému P

> (vstupy jsou zadány vhodně zvolenými kódy = slovy v určité abecedě [de facto stačí abeceda $\{0,1\}$] a jsou např. seřazeny podle délky a v rámci stejné délky lexikograficky])

a v druhém sloupci jsou uvedeny příslušné výstupy

> (v i-tém řádku je tedy v 1. sloupci i-tý vstup w a v 2. sloupci je příslušný výstup $p(w)$, kde p je zobrazení $p : IN \to OUT$ určené problémem P; tabulka T_P je prostě reprezentací zobrazení p).

V případě ANO/NE-problému se v druhém sloupci vyskytují jen výstupy Ano a Ne.

> *Poznámka.* S ohledem na budoucí úvahy si speciálně uvědomme, že pro každý vstup v tabulce T_P je definován příslušný výstup; v druhém sloupci se tedy nikde neobjeví „nedefinováno“ (znak $\perp$).

Detailně jsme si pak připomněli problém zdvojení slov v abecedě $\{0, 1\}$ (v tabulce je u každého slova u v 1. sloupci slovo uu v 2. sloupci), problém ekvivalence konečných automatů (Eq-FA), problém ekvivalence bezkontextových gramatik (Eq-CFG), problém zastavení Turingova stroje (HP, halting problem), diagonální problém zastavení (DHP).

Pak jsme si uvědomili, že každému *algoritmu* A odpovídá také jistá (vstupně/výstupní) tabulka T_A, která každému možnému vstupu w algoritmu A přiřazuje

- výstup vydaný algoritmem A – v případě, že běh algoritmu A pro vstup w je konečný,
- nebo speciální znak $\perp$ (nedefinováno) – v případě, že běh algoritmu A pro vstup w je nekonečný.

Jak víme, můžeme si (díky Churchově-Turingově tezi) pod pojmem algoritmus představit (např.) Turingův stroj. Každý Turingův stroj M tedy určuje příslušnou tabulku T_M. Můžeme jí říkat např.

vstupně/výstupní tabulka, zkráceně *I/O-tabulka*, stroje M.

Je to prostě (nekonečná) reprezentace vstupně/výstupního chování stroje M, tedy reprezentace příslušného I/O zobrazení $f_M : \Sigma^* \to \Gamma^* \cup \{\perp\}$, kde Σ je vstupní a Γ (celková) pásková abeceda stroje M.

> *Poznámka.* Víme, že se bez ztráty obecnosti můžeme omezit na stroje s abecedami $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \square\}$, jejichž I/O zobrazení je typu $\{0, 1\}^* \to \{0, 1\}^* \cup \{\perp\}$ (což mj. znamená, že stroje jsou navrženy tak, aby při ukončení výpočtu zůstal na pásce jediný souvislý úsek nul a jedniček [nepřerušovaný znaky $\square$]).

Pomocí uvedených pojmů tabulky problému a I/O-tabulky algoritmu (Turingova stroje) jsme vyjádřili, kdy je problém P *algoritmicky řešitelný*; v případě ANO/NE-problému hovoříme o *algoritmické rozhodnutelnosti*, či zkráceně jen *rozhodnutelnosti*.

Jen nám tedy naprosto jasné, co je myšleno, když se řekne „problém Eq-FA je rozhodnutelný“ a „problémy Eq-CFG, HP, DHP jsou nerozhodnutelné“.

Partie textu k prostudování

Problémy a algoritmy (část 6.1.), Turingovy stroje (část 6.2.). Model RAM (část 6.3.), simulace mezi výpočetními modely, Church-Turingova teze (6.4.), rozhodnutelnost a nerozhodnutelnost problémů (6.5.).

Cvičení

Na začátku diskuse opravené druhé zápočtové písemky.

Prezentace referátů

Referát č. 15 (Zásobníkové automaty)
Tvrzení „Ke každému ZA M lze sestrojit ZA M' s jedním stavem tž. $L(M) = L(M')$“ lze dokázat pomocí následující obecně popsané konstrukce.
Jednostavový ZA M', se stavem označeným s, bude mít zásobníkové symboly typu $\langle p, X, q\rangle$, kde p, q jsou stavy a X je zásobníkový symbol automatu M, a speciální počáteční zásobníkový symbol R.
Konkrétně pro $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ konstruujeme $M' = (\{s\}, \Sigma, \Gamma', \delta', s, R)$, kde $\Gamma' = (Q \times \Gamma \times Q) \cup \{R\}$ a δ' je určena následovně:

- $\delta'(s, \varepsilon, R) = \{(s, \langle q_0, Z_0, q\rangle) \mid q \in Q\}$,
- pro $(q', \varepsilon) \in \delta(q, a, X)$ $(a \in (\Sigma \cup \{\varepsilon\}))$ zařadíme do $\delta'(s, a, \langle q, X, q'\rangle)$ prvek (s, ε),
- pro $(q', A_1A_2 \dots A_n) \in \delta(q, a, X)$ $(n \geq 1)$ zařadíme do $\delta'(s, a, \langle q, X, \overline{q}\rangle)$ prvek $(s, \langle q', A_1, q_1\rangle\langle q_1, A_2, q_2\rangle \dots \langle q_{n-1}, A_n, \overline{q}\rangle)$ pro každé $\overline{q}, q_1, q_2, \dots, q_{n-1} \in Q$.

(Chápeme-li δ' jako množinu ‘instrukcí’, pak lze říci, že δ' je minimální množina instrukcí splňující výše uvedené podmínky.)
(Dá se ověřit, že

$$(s, w, \langle p, X, q\rangle) \vdash^*_{M'} (s, \varepsilon, \varepsilon) \iff (p, w, X) \vdash^*_{M} (q, \varepsilon, \varepsilon)\,, \tag{1}$$

a tedy každému přijímajícímu výpočtu automatu M nad slovem w odpovídá přijímající výpočet automatu M' nad w a naopak.)
Vašim úkolem je předvést aplikaci této konstrukce na zásobníkový automat M se vstupní abecedou $\{a, b\}$, zásobníkovou abecedou $\{A, B\}$, počátečním zásobníkovým symbolem A, množinou stavů $\{p, q, r\}$, počátečním stavem p a přechodovou funkcí δ definovanou následovně
$\delta(p, a, A) = \{(q, AA), (p, B)\}$,
$\delta(q, b, A) = \{(q, AA)\}$,
$\delta(p, \varepsilon, B) = \{(q, A)\}$,

$\delta(q, \varepsilon, A) = \{(r, \varepsilon)\}$,
$\delta(r, a, A) = \{(r, A)\}$,
$\delta(r, b, A) = \{(r, \varepsilon)\}$
(pro ostatní prvky def. oboru je funkční hodnota rovna $\emptyset$).
Přitom se snažte alespoň intuitivně přiblížit platnost uvedeného vztahu (1) v onom konkrétním případě.

Referát č. 16 (Simulace mezi různými variantami Turingových strojů 1)
Důkladně promyslete, popište a vysvětlete následující příklad (s návodem).

Mějme standardní Turingův stroj (předpokládající oboustranně nekonečnou pásku) $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Sestrojme k němu Turingův stroj $M' = (Q', \Sigma, \Gamma', \delta', q'_0, F')$, který předpokládá jen jednostranně (tj. pravostranně) nekonečnou pásku—tedy z nejlevější buňky (na níž stojí hlava na počátku) nemůže přejít doleva—a přitom simuluje stroj M.
Naznačíme možný způsob konstrukce:
$Q' = \{q'_0, q_1\} \cup \{q_x \mid x \in \Sigma\} \cup \{q_U \mid q \in Q\} \cup \{q_D \mid q \in Q\}$
$\Gamma' = \Sigma \cup (\Gamma \times \Gamma) \cup \{\not\in, \Box\}$
$F' = \{q_U \mid q \in F\} \cup \{q_D \mid q \in F\}$
$\delta'(q'_0, x) = (q_x, \not\in, +1)$ … pro $x \in \Sigma$
$\delta'(q_x, y) = (q_y, (x, \Box), +1)$ … pro $x, y \in \Sigma$
$\delta'(q_x, \Box) = (q_1, (x, \Box), -1)$ … pro $x \in \Sigma$
$\delta'(q_1, z) = (q_1, z, -1)$ … pro $z \neq \not\in$
$\delta'(q_1, \not\in) = ((q_0)_U, \not\in, +1)$
Obrázkem si znázorněte pásku a (na malém příkladu) počáteční fázi práce stroje M' (pozn.: asi vás napadne pojem ‘dvoustopá páska’); doplňte pak instrukce stroje M' (tedy dodefinujte zobrazení δ') tak, aby skutečně simuloval M. (Ještě kousek nápovědy: U v indexu u stavu znamená ‘up’, D znamená ‘down’).

Příklady

Příklad 9.1

Navrhněte (co nejjednodušší) dvoupáskový Turingův stroj (2P-TS) M řešící problém příslušnosti k jazyku (palindromů) $L = \{w \in \{a, b\}^* \mid w = w^R\}$. (Na minulém cvičení jste měli sestrojit standardní Turingův stroj k danému jazyku. Porovnejte obě řešení mezi sebou, a také s řešením dalšího příkladu.)

Příklad 9.2

K 2P-TS M z předchozího příkladu sestrojte standardní (jednopáskový, jednohlavový) Turingův stroj M', který řeší tentýž problém. Přitom se snažte používat obecný postup (který k libovolnému 2P-TS M sestrojí ekvivalentní TS M').