Týden 5

Přednáška-pondělí

Navázali jsme na předchozí přednášku.

Demonstrovali jsme si základní pojmy teorie bezkontextových gramatik. Ukázali jsme si *(levou) derivaci* slova $(((a \cdot a) + b)^*)$, příslušný *derivační strom*, apod.

Připomněli jsme definici *bezkontextové gramatiky* jako struktury

$$G = (\Pi, \Sigma, S, P)$$

a *jazyka generovaného gramatikou*

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}\,.$$

Pak jsme se vrátili k příkladu jazyka $RV(\{a, b\})$ a upravili jej tak, že v řetězcích (regulárních výrazech) vynecháváme tečku pro zřetězení a nemusíme plně uzávorkovávat. Např. výraz $(((a \cdot a) + b)^*)$ můžeme zapsat $(aa + b)^*$. Takto upravený jazyk generuje např. gramatika

$$R \longrightarrow \emptyset \mid \varepsilon \mid a \mid b \mid R + R \mid RR \mid R^* \mid (R)\,.$$

Všimli jsme si ovšem, že např. slovo $aa+b$ má v této gramatice dva různé derivační stromy; tedy tato gramatika *není jednoznačná*. (Příčinou je tady fakt, že naše dohodnutá priorita operátorů není v gramatice reflektována.)

Demonstrovali jsme, že v tomto případě lze nalézt ekvivalentní gramatiku (tedy gramatiku generující tentýž jazyk), která jednoznačná je. Úvahami nad strukturou regulárních výrazů jsme postupně došli ke gramatice

$$\begin{aligned} R &\longrightarrow T + R \mid T \\ T &\longrightarrow FT \mid F \\ F &\longrightarrow F^* \mid (R) \mid C \\ C &\longrightarrow \emptyset \mid \varepsilon \mid a \mid b \end{aligned}$$

Pomohla nám úvaha, že T (Term) reprezentuje ty regulární výrazy, které nejsou ve tvaru $R_1 + R_2$ (pro dva regulární výrazy R_1, R_2), a F (Factor) reprezentuje ty výrazy, které nejsou ve tvaru $R_1 + R_2$ ani $R_1 R_2$.

Uvedli jsme pojem *(vnitřně) jednoznačný bezkontextový jazyk* a několik souvisejících poznámek.

Započali jsme poslední příklad ze cvičení (gramatika pro booleovské formule) a zakončili konstrukcí gramatik pro dva jednoduché jazyky:

Promyšlením tvarů slov jsme zkonstruovali gramatiky pro jazyk palindromů $\{w \in \{a, b\}^* \mid w = w^R\}$

$$S \longrightarrow \varepsilon \mid a \mid b \mid aSa \mid bSb$$

a pro jazyk posloupností v abecedě $\{\,(,),[,]\,\}$, které odpovídají správnému uzávorkování

$$S \longrightarrow \varepsilon \mid SS \mid (S) \mid [S]\,.$$

Přednáška-čtvrtek

Věnovali jsme se dvoucestným (či dvousměrným) konečným automatům. Takový (deterministický) automat, stručně 2-KA, je definován jako struktura

$$A = (Q, \Sigma, \#, \$, \delta, q_0, F)$$

kde Q, Σ, q_0, F mají stejný význam jako u standardních (jednosměrných) konečných automatů a $\#, \$ \notin \Sigma$ jsou speciální symboly označující levou a pravou zarážku. Přechodová funkce je nyní typu

$$\delta : (Q - F) \times (\Sigma \cup \{\#, \$\}) \to Q \times \{-1, +1\}\,.$$

Řídicí jednotka automatu A je čtecí hlavou spojena s páskou, na niž je na začátku zapsáno vstupní slovo $w \in \Sigma^*$ ohraničené na levé straně zarážkou # a na pravé straně zarážkou \$. Např. příslušná 'instrukce' $(q, a) \to (q', -1)$ znamená, že když automat A je ve stavu q a čte na pásce symbol a, pak přejde do stavu q' a posune hlavu o jedno políčko doleva ($+1$ znamená doprava). Na levé zarážce ovšem může pokračovat jen doprava, na pravé zarážce jen doleva. Další změna je v tom, že v přijímajícím stavu, prvku množiny F, výpočet končí (takový stav je opravdu *koncový*). Slovo je přijato, jestliže výpočet nad ním skončí (v přijímajícím stavu); není přijato, pokud je výpočet nekonečný. Z technických důvodů můžeme předpokládat, že do koncového stavu může automat přejít jen při čtení pravé zarážky \$.

Uvažujme nyní standardní konečný automat A a jazyk $L = \{u \mid uu \in L(A)\}$. Otázka je, jestli je L zaručeně regulární jazyk, tedy jestli existuje KA A' tak, že $L(A') = L$. To není na první pohled zřejmé. Jednoduché ale je zkonstruovat 2-KA A' tak, že $L(A') = L$. Takový A' dostane slovo $\#u\$$, odsimuluje A na u, po příjezdu na \$ si zapamatuje aktuální stav automatu A, přejede na začátek a pokračuje v simulaci automatu A při druhém běhu na u.

Zamysleme se, jestli 2-KA umějí víc než KA, tedy jestli takové automaty rozpoznají i některé neregulární jazyky. Uvažujme tedy 2-KA $A = (Q, \Sigma, \#, \$, \delta, q_0, F)$ a podívejme se na množinu $\{w \backslash L(A) \mid w \in \Sigma^*\}$. Víme, že jestliže je tato množina konečná, pak je $L(A)$ regulární.

Podívejme se, jak vypadá výpočet A nad slovem $\#wu\$$. Jedna možnost je, že nikdy neopustí w; pak $w \backslash L(A) = \emptyset$. Jinak dojde k situaci, kdy výpočet poprvé opustí w vpravo. Stav, ve kterém výpočet poprvé vstoupí na u, záleží pouze na w, označme jej q_w. Když pak výpočet vstoupí na slovo w znovu (zprava) ve stavu q, pak buď už w neopustí, nebo se po nějaké době vrátí doprava na u ve stavu q'; stav q' je plně určen slovem w a oním vstupním stavem q. Je klíčové si uvědomit, že pro zjištění, zda slovo wu je automatem A přijato, nepotřebujeme znát kompletní slovo w, ale stačí nám informace označená $Beh(w)$ ('behaviour', tj. chování slova w), což je dvojice $Beh(w) = (q, f)$, kde $q \in Q \cup \{\perp\}$ a $f : Q \to Q \cup \{\perp\}$. Když je první komponenta rovna $\perp$, znamená to, že počáteční výpočet w nikdy neopustí (doprava); jinak tato komponenta udává výše zmíněný stav q_w. Druhá komponenta je funkcí; pro q určuje $f(q)$ onen návratový stav q', když výpočet vjede na

slovo w zprava ve stavu q. Pochopitelně $f(q) = \perp$, jestliže po příjezdu na w zprava ve stavu q už slovo w nebude opuštěno (doprava).

Takže i jazyk $w\backslash L(A)$ je plně určen hodnotou $Beh(w)$. Jinými slovy, když $Beh(w) = Beh(w')$, tak $w\backslash L(A) = w'\backslash L(A)$. Ovšem funkce Beh nabývá jen konečně mnoha hodnot, takže množina $\{w\backslash L(A) \mid w \in \Sigma^*\}$ je opravdu konečná a jazyk $L(A)$ je tedy regulární. Dokázali jsme tedy:

Věta. Dvoucestné konečné automaty rozpoznávají právě regulární jazyky.

Partie textu k prostudování

Části 4.1., 4.2., 4.3., 4.4. (bezkontextové gramatiky).

(Máte si udělat přinejmenším dobrou první představu a zamyslet se nad příklady, speciálně těmi plánovanými na cvičení, ať se můžete na cvičení aktivně účastnit a případné problémy si tam objasnit.)

Cvičení

Na začátku diskuse opravené první zápočtové písemky.

Prezentace referátů

Referát č. 7 (Hltavý algoritmus 1)

Vysvětlete (na vhodně zvoleném případu), jak lze problém (ze studijního textu)

> *Název problému*: *Výběr aktivit*
>
> *Vstup*: množina konečně mnoha aktivit $\{1, 2, \ldots, n\}$ s pevně určenými časovými intervaly (s_1, f_1), (s_2, f_2), $\ldots$, (s_n, f_n), kde $(\forall i, 1 \leq i \leq n) : s_i < f_i$
> *Výstup*: množina obsahující největší možný počet vzájemně kompatibilních aktivit (tj. aktivit s vzájemně se nepřekrývajícími intervaly)

řešit hltavým (greedy) algoritmem.

Ukažte myšlenku indukce podle počtu aktivit n prokazující, že uvedený přístup skutečně vede k optimálnímu řešení.

Referát č. 8 (Hltavý algoritmus 2)

Připomeňte na vhodně zvoleném případu, jak se řeší problém konstrukce minimální kostry grafu hltavým přístupem, a ilustrujte myšlenku důkazu toho, že tento přístup skutečně vede k optimu. (Můžete vyjít z popisu ve studijním textu a podle potřeby použít další materiály.)

Příklady

Příklad 5.1

Uvažujme jazyk $L = \{ w \in \{a, b\}^* \mid |w| \geq 1 \text{ a } |w|_a = |w|_b \}$.
Charakterizujte slova z $L^2 = L \cdot L$. Je pravda, že $L = L^2$? Platí případně alespoň jedna z inkluzí $L \subseteq L^2$, $L^2 \subseteq L$?
Charakterizujte slova z $L - L^2$.
Na základě předešlých úvah navrhněte bezkontextovou gramatiku generující L.

Příklad 5.2

Snažte se co nejvýstižněji charakterizovat jazyk generovaný gramatikou
$S \longrightarrow bSS \,|\, a$

Příklad 5.3

Navrhněte bezkontextové gramatiky generující následující jazyky:

- $L_1 = \{ w \in \{a, b\}^* \mid w \text{ obsahuje podslovo } baab \}$
- $L_2 = \{ w \in \{a, b\}^* \mid |w|_b \bmod 3 = 0 \}$
- $L_3 = \{ ww^R \mid w \in \{a, b\}^* \}$
- $L_4 = \{ 0^n 1^m 0^n \mid m, n \geq 0 \}$
- $L_5 = \{ 0^n 1^m \mid 1 \leq n \leq m \leq 2n \}$

Příklad 5.4

(podle plánu začato na přednášce; při nedostatku času dokončit příště)
Uvažujme jazyk sestávající ze všech booleovských formulí s proměnnými $x1, x2, \ldots$ a logickými spojkami $\neg, \wedge, \vee$; mohou se v nich používat závorky $(,)$, ale není nutné plně závorkovat. Každá taková formule je tedy řetězcem v abecedě

$$\Sigma = \{ x, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \neg, \wedge, \vee, (,) \} ;$$

jako příklad může sloužit řetězec $(\neg x15 \vee x2 \wedge x5) \wedge \neg x21 \vee \neg(x2 \vee x5)$, který do jazyka patří. (Samozřejmě zde můžeme preferovat přehlednější zápis $(\neg x_{15} \vee x_2 \wedge x_5) \wedge \neg x_{21} \vee \neg(x_2 \vee x_5)$, ale to není podstatné.)
Navrhněte co nejjednodušší bezkontextovou gramatiku generující uvedený jazyk.
Takto navržená (jednoduchá) gramatika asi není jednoznačná; ověřte. Zkonstruujte pak pro stejný jazyk jednoznačnou gramatiku, u níž derivační stromy přirozeně odpovídají obvyklé prioritě operátorů: negace váže silněji než konjunkce a konjunkce váže silněji než disjunkce.