Týden 7

Přednáška

Nebezkontextové jazyky

Metodou obrázků derivačních stromů z části 5.3. jsme si ukázali, proč jazyk

$$L = \{a^n b^n c^n \mid n \geq 0\}$$

není bezkontextový. Porozuměli jsme tak pumping lemmatu (neboli $uvwxy$-teorému) pro bezkontextové jazyky:

> *Věta.* Nechť L je bezkontextový jazyk. Pak existuje přirozené číslo n tž. každé slovo $z \in L$, $|z| \geq n$, (tedy každé 'dlouhé' slovo z jazyka L) lze psát ve tvaru $z = uvwxy$, přičemž platí
>
> - $vx \neq \varepsilon$,
> - $|vwx| \leq n$,
> - pro vš. $i \geq 0$ je $uv^i wx^i y \in L$.

Intuici k poznání nebezkontextových jazyků jsme si posílili zvážením jazyků

$$L_1 = \{ww \mid w \in \{0,1\}^*\},$$
$$L_2 = \{\, 0^m 1^n 0^m \mid m = 2n \,\},$$

u nichž obou jsme usoudili, že ne všechna jejich dlouhá slova lze napsat v patřičném tvaru $uvwxy$, a že tedy nejsou bezkontextové. (U jazyka L_1 jde např. o slova tvaru $0^n 1^n 0^n 1^n$.)

Uzávěrové vlastnosti třídy CFL

Připomněli jsme si, že snadno umíme ukázat, že CFL je uzavřena vůči sjednocení.
Pak jsme si všimli, že jazyky $L_1 = \{a^i b^j c^k \mid i = j\}$ a $L_2 = \{a^i b^j c^k \mid j = k\}$ jsou bezkontextové, ale $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ je jazyk, o němž víme, že bezkontextový není.
Tedy CFL není uzavřena na průnik. Díky de Morganovým pravidlům ihned vidíme, že CFL není uzavřena ani na doplněk.
(Jelikož $(L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})})$, tak z uzavřenosti na doplněk by díky uzavřenosti na sjednocení vyplynula uzavřenost na průnik.)
Poznámka. Jak jsme si už řekli, třída DCFL, tj. třída jazyků rozpoznatelných deterministickými zásobníkovými automaty (přijímacími stavy), je vlastní podtřídou CFL. Dá se ukázat, že (pod)třída DCFL je uzavřena vůči doplňku; není ovšem uzavřena vůči průniku (jak dokazují výše uvedené jazyky $\{a^i b^j c^k \mid i = j\}$, $L_2 = \{a^i b^j c^k \mid j = k\}$), a tedy není uzavřena ani vůči sjednocení.

(Výpočetní) problémy, rozhodovací (ANO/NE) problémy, ...

Připomněli jsme si obecné definice a konkrétní problémy, jako např. SAT [problém splnitelnosti booleovských formulí], problém minimální kostry grafu (i v rozhodovací verzi).
Je samozřejmé, že po pochopení problému musí být každý schopen uvést konkrétní příklady instancí (vstupů) a příslušných výstupů ... .
Také jsme diskutovali přirozené způsoby kódování vstupů a výstupů slovy ve vhodné abecedě (která lze nakonec zakódovat „binárně“, tedy pomocí slov v abecedě $\{0,1\}$).

Turingovy stroje

Připomněli jsme si, co bylo cílem (třiadvacetiletého) Alana Turinga, když v roce 1936 zaváděl tzv. Turingův stroj, a nastínili jsme význam tzv. *Church-Turingovy teze*, která se stručně dá vyjádřit sloganem

$$\textit{algoritmus} = \textit{Turingův stroj.}$$

Naznačili jsme konstrukci konkrétního Turingova stroje M_1 s dvěma koncovými stavy q_{accept} a q_{reject}, který přijímá (nebezkontextový) jazyk

$$\{a^n b^n c^n \mid n \geq 1\}\,.$$

Jedná se vlastně o sestavení (jako vždy, pokud možno přehledného, okomentovaného) programu s jediným typem instrukcí; jeho začátek může vypadat následovně.

$(q_0, a) \to (q_1, \bar{a}, +1)$
$(q_1, x) \to (q_1, x, +1)$ pro $x \in \{a, \bar{b}\}$
$(q_1, b) \to (q_2, \bar{b}, +1)$
...

Zároveň jsme připomněli definici Turingova stroje jako struktury $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ (kde $\Sigma \subseteq \Gamma$ a $\Gamma \setminus \Sigma$ vždy obsahuje speciální symbol $\Box$); speciálně jsme si uvědomili, že přechodová funkce

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

je, de facto, příslušnou množinou instrukcí ...
Připomněli jsme si, že na výpočet Turingova stroje M se dá pohlížet jako na posloupnost konfigurací $K_0 \vdash_M K_1 \vdash_M K_2 \vdash_M \cdots$.
(V našem konkrétním případě stroje M_1 např. platí $(q_0 aaabbbccc) \vdash (\bar{a} q_1 aabbbccc) \vdash (\bar{a}\bar{a} q_1 abbbccc) \vdash (\bar{a}\bar{a}\bar{a} q_1 bbbccc) \vdash (\bar{a}\bar{a}\bar{a}\bar{b} q_2 bbccc) \vdash \cdots \vdash (\bar{a}\bar{a}\bar{a}\bar{b}\bar{b}\bar{b}\bar{c}\bar{c}\bar{c} q_{accept}\Box)$; ale např. $(q_0 aabbbccc) \vdash^* \bar{a}\bar{a}\bar{b}\bar{b} q_{reject} b\bar{c}\bar{c}c$).)
Uvědomili jsme si, že náš *stroj* (tedy algoritmus) *řeší problém*

NÁZEV: *Příslušnost k jazyku* $L = \{a^n b^n c^n \mid n \geq 1\}$

VSTUP: $w \in \{a, b, c\}^*$

VÝSTUP: ANO, když $w \in L$, NE jinak.

Jinými slovy, náš *stroj* (algoritmus) *rozhoduje (rozhodovací, neboli* Ano/Ne*) problém*

Název: *Příslušnost k jazyku* $L = \{a^n b^n c^n \mid n \geq 1\}$

Vstup: $w \in \{a, b, c\}^*$

Otázka: Je $w \in L$?

Pak jsme si načrtli, jak by pracoval Turingův stroj M_2, který řeší následující problém (jenž není typu Ano/Ne):

Název: *Zdvojení slova v abecedě* $\{a, b\}$.

Vstup: $w \in \{a, b\}^*$.

Výstup: ww.

Uvědomili jsme si, že každý Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ přirozeně definuje *částečné zobrazení*

$$f_M : \Sigma^* \to \Gamma^* .$$

Zobrazení je obecně částečné (tedy ne nutně totální) proto, že pro některé vstupy $w \in \Sigma^*$ se výpočet M nemusí zastavit a příslušný výstup tedy není definován; výrazem $!M(w)$ se zpravidla označuje fakt, že M se pro vstup w zastaví (a hodnota výstupu $f_M(w)$, někdy označovaná přímo jako $M(w)$, je definována).

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

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

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“.)

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 $\perp$).

Např. u *problému zdvojení slov* v abecedě $\{0, 1\}$ je možné příslušnou tabulku (zobrazení) znázornit takto:

Vstup	Příslušný výstup
ε	ε
0	00
1	11
00	0000
01	0101
10	1010
11	1111
000	000000
...	...

Analogicky si lze přirozeně představit příslušnou (nekonečnou) tabulku pro problém ekvivalence konečných automatů (Eq-FA), problém ekvivalence bezkontextových gramatik (Eq-CFG), problém zastavení Turingova stroje (HP, halting problem), diagonální problém zastavení (DHP), atd.
Uvědomme si nyní, 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

Uzávěrové vlastnosti CFL (část 5.2.). Nebezkontextové jazyky (část 5.3.). Problémy a algoritmy (část 6.1.), Turingovy stroje (část 6.2.). (Informativně) Model RAM (část 6.3.), simulace mezi výpočetními modely, Church-Turingova teze (6.4.), rozhodnutelnost a nerozhodnutelnost problémů (6.5.).

Cvičení

Příklad 7.1

Připomeňme, že jako základní jsme (u nedeterministických zásobníkových automatů) brali přijímání slova prázdným zásobníkem. Zásobníkový automat jsme definovali jako jistou strukturu $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, kde jazykem přijímaným automatem M rozumíme jazyk

$$L(M) = \{\, w \in \Sigma^* \mid \exists q \in Q : (q_0, w, Z_0) \vdash_M^* (q, \varepsilon, \varepsilon) \,\}\,.$$

U verze přijímání přijímacím stavem definujeme zásobníkový automat jako $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $F \subseteq Q$, a pak definujeme

$$L(M) = \{\, w \in \Sigma^* \mid \exists q \in F, \exists \alpha \in \Gamma^* : (q_0, w, Z_0) \vdash_M^* (q, \varepsilon, \alpha) \,\}\,.$$

Řekněme, že k zás. automatu $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ přijímajícímu přijímacím stavem jazyk L chceme sestrojit zás. automat M_2 přijímající jazyk L prázdným zásobníkem. Stačí prostě upravit M_1 tak, že pro každý stav $q \in F$ dodáme instrukci $(q, \varepsilon, X) \to (q, \varepsilon)$ pro každé $X \in \Gamma$? Když ne, zkuste alespoň stručně navrhnout bezpečnou úpravu …

Příklad 7.2

Na přednášce jsme mj. ukázali, že třída CFL není uzavřena na doplněk. Existuje tedy jazyk $L \subseteq \Sigma^*$ (pro nějakou abecedu Σ), který je bezkontextový, přičemž jeho doplněk $\overline{L} = \Sigma^* - L$ bezkontextový není.
Víme, že jazyk $L_1 = \{ww \mid w \in \{a, b\}^*\}$ bezkontextový není. Zkuste prokázat, že $L_2 = \overline{L_1}$ bezkontextový je a představuje tak konkrétní příklad jazyka z CFL, jehož doplněk v CFL není.
Nápověda. Vystihněte co nejjednodušeji, jak vypadá slovo z L_2, které má sudou délku, tedy $2d$ pro nějaké $d \geq 1$. Pak se zkuste zamyslet, zda vám pomůže následující vztah:

$$2d = (d_1 + 1 + d_2) + (d_1 + 1 + d_2) = (d_1 + 1 + d_1) + (d_2 + 1 + d_2)\,.$$

Nakonec si tipněte, jestli se může podařit navrhnout *deterministický* zásobníkový automat přijímající L_2.
(Pomůže vám nějak u předchozí otázky, když se dozvíte, že třída DCFL je uzavřena vůči doplňku?)

Příklad 7.3

(Na základě alespoň intuitivních argumentů) určete, které z daných jazyků
jsou regulární:
jsou bezkontextové, ale ne regulární:
nejsou bezkontextové:

$L_1 = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$
$L_2 = \{w \in \{a,b\}^* \mid |w|_a$ je sudé $\}$
$L_3 = \{w \in \{a,b\}^* \mid w$ obsahuje podslovo $abba\}$
$L_4 = \{w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c\}$
$L_5 = \{w \in \{a,b\}^* \mid |w|_a$ je prvočíslo$\}$
$L_6 = \{\, 0^m 1^n \mid m \leq 2n \,\}$
$L_7 = \{\, 0^m 1^n 0^m \mid m = 2n \,\}$

Příklad 7.4

Na přednášce jsme mj. diskutovali o (výpočetních) problémech. Zformulujte, co to je (v našem kontextu) pojem *problém* a pak speciálně rozhodovací (neboli ANO/NE) problém.
Specifikujte např. problém jednoznačnosti bezkontextových gramatik, s ukázkou pozitivní a negativní instance.
Zformulujte definici (algoritmicky) rozhodnutelného (obecně *algoritmicky řešitelného*) problému ...
(Vzpomínáte si, co jsme řekli o (ne)rozhodnutelnosti problému jednoznačnosti gramatik?)

Příklad 7.5

Zformalizujte situaci obchodního cestujícího, který má soupis měst k projetí a zná vzdálenost mezi každými dvěma městy, jakožto (výpočetní) problém; jde přitom o nalezení co nejkratší cesty, při níž jsou navštívena všechna města. Pak navrhněte rozhodovací verzi problému (při níž je dán limit). Vždy uveďte příklady konkrétních instancí a příslušných výstupů. (V případě rozhodovací verze uveďte alespoň jednu pozitivní a jednu negativní instanci.)

Příklad 7.6

Navrhněte (co nejjednodušší) Turingův stroj M řešící problém příslušnosti k jazyku (palindromů) $L = \{w \in \{a,b\}^* \mid w = w^R\}$ a zadejte jej (přehledným) seznamem instrukcí.

Příklad 7.7

Navrhněte (co nejjednodušší) dvoupáskový Turingův stroj (2P-TS) M řešící problém příslušnosti k jazyku (palindromů) $L = \{w \in \{a,b\}^* \mid w = w^R\}$.
Porovnejte s řešením předchozího příkladu a později také s řešením následujícího příkladu.

Příklad 7.8

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 standardní TS M' simulující M.

Příklad 7.9

Navrhněte způsob, jak lze Turingův stroj s obecnou páskovou abecedou Γ simulovat Turingovým strojem s páskovou abecedou $\{0, 1, \square\}$.