Postův korespondenční problém

Postův korespondenční problém

Definice 10.1

- Postův systém nad abecedou Σ je dán neprázdným seznamem S dvojic neprázdných řetězců nad Σ, $S = \langle(\alpha_1, \beta_1), ..., (\alpha_k, \beta_k)\rangle$, $\alpha_i, \beta_i \in \Sigma^+$, $k \geq 1$.
- Řešením Postova systému je každá neprázdná posloupnost přirozených čísel $I = \langle i_1, i_2, ..., i_m\rangle$, $1 \leq i_j \leq k$, $m \geq 1$, taková, že:

$$\alpha_{i_1}\alpha_{i_2}...\alpha_{i_m} = \beta_{i_1}\beta_{i_2}...\beta_{i_m}$$

  (Pozor: m není omezené a indexy se mohou opakovat!)
- Postův problém (PCP) zní: Existuje pro daný Postův systém řešení?

Příklad 10.1

- Uvažujme Postův systém $S_1 = \{(b, bbb), (babbb, ba), (ba, a)\}$ nad $\Sigma = \{a, b\}$. Tento systém má řešení $I = \langle 2, 1, 1, 3\rangle$: $babbb\ b\ b\ ba = ba\ bbb\ bbb\ a$.
- Naopak Postův systém $S_2 = \{(ab, abb), (a, ba), (b, bb)\}$ nad $\Sigma = \{a, b\}$ nemá řešení, protože $|\alpha_i| < |\beta_i|$ pro $i = 1, 2, 3$.

Nerozhodnutelnost PCP

Věta 10.1 **Postův korespondenční problém je nerozhodnutelný.**

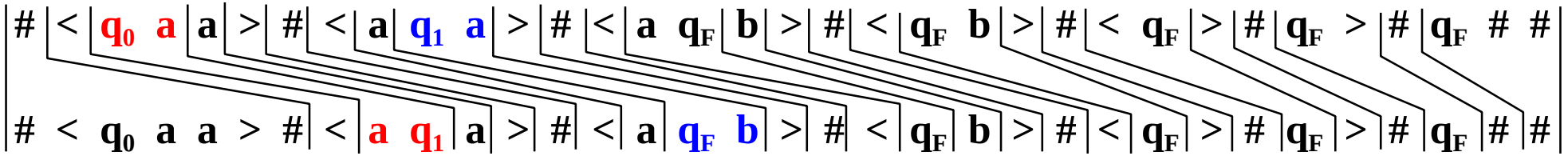
Důkaz. (Idea) Dá se ukázat, že nerozhodnutelnost PCP plyne z nerozhodnutelnosti tzv. iniciálního PCP, u kterého požadujeme, aby řešení začínalo vždy jedničkou. Nerozhodnutelnost iniciálního PCP se dá ukázat redukcí z problému náležitosti pro TS:

- Konfiguraci výpočtu TS lze zakódovat do řetězce: použitý obsah pásky ohraničíme speciálními značkami (a jen ten si pamatujeme), řídicí stav vložíme na aktuální pozici hlavy na pásce.
- Posloupnost konfigurací TS při přijetí řetězce budeme reprezentovat jako konkatenaci řetězců, která vzniká řešením PCP.
- Jedna z uvažovaných konkatenací bude celou dobu (až na poslední fázi) delší: na začátku bude obsahovat počáteční konfiguraci a pak bude vždy o krok napřed. V posledním fázi výpočtu konkatenace „zarovnáme“ (bude-li možné výpočet simulovaného TS ukončit přijetím).
- Výpočet TS budeme modelovat tak, že vždy jednu konkatenaci postupně prodloužíme o aktuální konfiguraci simulovaného TS a současně v druhé konkatenaci vygenerujeme novou konfiguraci TS.

Důkaz pokračuje dále.

Pokračování důkazu.

- Jednotlivé dvojice PCP budou modelovat následující kroky:
  - Vložení počáteční konfigurace simulovaného TS do jedné z konkatenací např. pravostranné $(\#, \#\text{počáteční\_konfigurace})$, $\# \notin \Gamma$ používáme jako oddělovač konfigurací.
  - Kopírování symbolů na pásce před a po aktuální pozici hlavy (z, z) pro každé $z \in \Gamma \cup \{\#, <, >\}$, kde $<$, $>$ ohraničují použitou část pásky.
  - Základní změna konfigurace: přepis $\delta(q_1, a) = (q_2, b)$: $(q_1 a, q_2 b)$, posuv doprava $\delta(q_1, a) = (q_2, R)$: $(q_1 a, a q_2)$, posuv doleva $\delta(q_1, b) = (q_2, L)$: $(a q_1 b, q_2 a b)$ pro každé $a \in \Gamma \cup \{<\}$. Navíc je zapotřebí ošetřit nájezd na $>$: čtení Δ, rozšiřování použité části pásky.
  - Pravidla pro „zarovnání“ obou konkatenací při přijetí: na levé straně umožníme přidat symbol v okolí q_F, aniž bychom ho přidali na pravé straně.
- Simulace výpočtu TS, který načte a, posune hlavu doprava, přepíše a na b a zastaví, na vstupu aa by pak vypadala takto:



- Obecná korektnost konstrukce se dá dokázat indukcí nad délkou výpočtu.

□

Nerozhodnutelnost redukcí z PCP

❖ Redukce z PCP (resp. jeho doplňku) se velmi často používají k důkazům, že určitý problém není rozhodnutelný (resp. není ani částečně rozhodnutelný).

❖ Jako příklad uvedeme důkaz faktu, že problém prázdnosti jazyka dané kontextové gramatiky není ani částečně rozhodnutelný:

- Použijeme redukci z komplementu PCP. Redukce přiřadí seznamu $S = (\alpha_1, \beta_1), ..., (\alpha_k, \beta_k)$, definujícímu instanci PCP, kontextovou gramatiku G takovou, že PCP založený na S nemá řešení právě tehdy, když $L(G) = \emptyset$.
- Uvažme jazyky L_α, L_β nad $\Sigma \cup \{\#, 1, ..., k\}$ (předp. $\Sigma \cap \{\#, 1, ..., k\} = \emptyset$):
  - $L_\alpha = \{\alpha_{i_1} ... \alpha_{i_m} \# i_m ... i_1 \mid 1 \leq i_j \leq k, j = 1, ..., m, m \geq 1\}$,
  - $L_\beta = \{\beta_{i_1} ... \beta_{i_m} \# i_m ... i_1 \mid 1 \leq i_j \leq k, j = 1, ..., m, m \geq 1\}$.
- Je zřejmé, že L_α, L_β jsou kontextové (dokonce deterministické bezkontextové) a tudíž $L_\alpha \cap L_\beta$ je také kontextový jazyk (věta 8.10) a můžeme tedy efektivně sestavit gramatiku G, která tento jazyk generuje (např. konstrukcí přes LOA).
- $L_\alpha \cap L_\beta$ zřejmě obsahuje právě řetězce $u\#v$, kde v odpovídá reverzi řešení dané instance PCP.
- Hledaná redukce tedy přiřadí dané instanci PCP gramatiku G.

□

Souhrn některých vlastností jazyků

❖ Uvedeme nyní souhrn některých důležitých vlastností různých tříd jazyků; některé jsme dokázali, důkazy jiných lze nalézt v literatuře[a] (u otázek nerozhodnutelnosti se často užívá redukce z PCP) – R = rozhodnutelný, N = nerozhodnutelný, A = vždy splněno:

	Reg	DCF	CF	CS	Rec	RE
$w \in L(G)$?	R	R	R	R	R	N
$L(G)$ prázdný? konečný?	R	R	R	N	N	N
$L(G) = \Sigma^*$?	R	R	N	N	N	N
$L(G) = R$, $R \in \mathcal{L}_3$?	R	R	N	N	N	N
$L(G_1) = L(G_2)$?	R	R	N	N	N	N
$L(G_1) \subseteq L(G_2)$?	R	N	N	N	N	N
$L(G_1) \in \mathcal{L}_3$?	A	R	N	N	N	N
$L(G_1) \cap L(G_2)$ je stejného typu?	A	N	N	A	A	A
$L(G_1) \cup L(G_2)$ je stejného typu?	A	N	A	A	A	A
Komplement $L(G)$ je stejného typu?	A	A	N	A	A	N
$L(G_1).L(G_2)$ je stejného typu?	A	N	A	A	A	A
$L(G)^*$ je stejného typu?	A	N	A	A	A	A
Je G víceznačná?	R	N	N	N	N	N

[a] Např. I. Černá, M. Křetínský, A. Kučera. Automaty a formální jazyky I. FI MU, 1999.

Riceova věta

Nerozhodnutelnost je pravidlo, ne výjimka.

Riceova věta – první část

Věta 10.2 ***Každá*** **netriviální vlastnost rekurzívně vyčíslitelných jazyků je nerozhodnutelná.**

Definice 10.2 Budiž dána abeceda Σ. Vlastnost rekurzívně vyčíslitelných množin je zobrazení $P : \{$ rekurzívně vyčíslitelné podmnožiny množiny $\Sigma^* \} \to \{\bot, \top\}$, kde $\top$, resp. $\bot$ reprezentují pravdu, resp. nepravdu.

Příklad 10.2 Vlastnost prázdnosti můžeme reprezentovat jako zobrazení

$$P(A) = \begin{cases} \top, \text{ jestliže } A = \emptyset, \\ \bot, \text{ jestliže } A \neq \emptyset. \end{cases}$$

❖ Zdůrazněme, že nyní mluvíme o vlastnostech rekurzívně vyčíslitelných množin, *nikoliv* TS, které je přijímají – následující vlastnosti tedy nejsou vlastnostmi r. v. množin:

- TS M má alespoň 2005 stavů.
- TS M zastaví na všech vstupech.

Definice 10.3 Vlastnost rekurzívně vyčíslitelných množin je netriviální, pokud není vždy pravdivá ani vždy nepravdivá.

Důkaz 1. části Riceovy věty

Důkaz.

- Nechť P je netriviální vlastnost r.v. množin. Předpokládejme beze ztráty obecnosti, že $P(\emptyset) = \bot$, pro $P(\emptyset) = \top$ můžeme postupovat analogicky.
- Jelikož P je netriviální vlastnost, existuje r.v. množina A taková, že $P(A) = \top$. Nechť K je TS přijímající A.
- Redukujeme HP na $\{\langle M\rangle \mid P(L(M)) = \top\}$. Z $\langle M\rangle\#\langle w\rangle$ sestrojíme $\sigma(\langle M\rangle\#\langle w\rangle) = \langle M'\rangle$, kde M' je 2-páskový TS, který na vstupu x:
  1. Uloží x na 2. pásku.
  2. Zapíše na 1. pásku w – w je „uložen“ v řízení M'.
  3. Odsimuluje na 1. pásce M – M je rovněž „uložen“ v řízení M'.
  4. Pokud M zastaví na w, odsimuluje K na x a přijme, pokud K přijme.
- Dostáváme:
  - M zastaví na $w \Rightarrow L(M') = A \Rightarrow P(L(M')) = P(A) = \top$,
  - M cyklí na $w \Rightarrow L(M') = \emptyset \Rightarrow P(L(M')) = P(\emptyset) = \bot$,

  A máme tedy skutečně redukci HP na $\{\langle M\rangle \mid P(L(M)) = \top\}$.
  Protože HP není rekurzivní, není rekurzivní ani $P(L(M))$ a tudíž není rozhodnutelné, zda $L(M)$ splňuje vlastnost P.

□

Riceova věta – druhá část

Definice 10.4 Vlastnost P r.v. množin nazveme monotónní, pokud pro každé dvě r.v. množiny A, B takové, že $A \subseteq B$, $P(A) \Rightarrow P(B)$.

Příklad 10.3 Mezi monotónní vlastnosti patří např.:

- A je nekonečné.
- $A = \Sigma^*$.

Naopak mezi nemonotónní vlastnosti patří např.:

- A je konečné.
- $A = \emptyset$.

Věta 10.3 ***Každá*** **netriviální nemonotónní vlastnost rekurzívně vyčíslitelných jazyků není ani částečně rozhodnutelná.**

Důkaz. Redukcí z co-HP – viz např. D. Kozen. Automata and Computability. □

Alternativy k TS

Některé alternativy k TS

❖ Mezi výpočetní mechanismy mající ekvivalentní výpočetní sílu jako TS patří např. automaty s (jednou) frontou:

- Uvažme stroj s konečným řízením, (neomezenou) FIFO frontou a přechody na nichž je možno načíst ze začátku fronty a zapsat na konec fronty symboly z frontové abecedy Γ.
- Pomocí „rotace“ fronty je zřejmě možné simulovat pásku TS.

❖ Ekvivalentní výpočetní sílu jako TS mají také zásobníkové automaty se dvěma (a více) zásobníky:

- Intuitivně: obsah pásky simulovaného TS máme v jednom zásobníku; chceme-li ho změnit (obecně nejen na vrcholu), přesuneme část do druhého zásobníku, abychom se dostali na potřebné místo, provedeme patřičnou změnu a vrátíme zpět odloženou část zásobníku.
- Poznámka: rovněž víme, že pomocí dvou zásobníků můžeme implementovat frontu.

❖ Jiným výpočetním modelem s plnou Turingovskou silou jsou automaty s čítači (pro dva a více čítačů) a operacemi $+1$, -1 a test na 0:

- Zmíněné automaty mají konečné řízení a k čítačů, přičemž v každém kroku je možné tyto čítače nezávisle inkrementovat, dekrementovat a testovat na nulu (přechod je podmíněn tím, že jistý čítač obsahuje nulu).
- Pomocí čtyř čítačů je snadné simulovat dva zásobníky:
  - U ZA postačí mít $\Gamma = \{0, 1\}$: různé symboly můžeme kódovat určitým počtem 0 oddělených 1. Obsah zásobníku má pak charakter binárně zapsaného čísla. Vložení 0 odpovídá vynásobení 2, odebrání 0 vydělení 2. Podobně je tomu s vložením/odebráním 1.
  - Binární zásobník můžeme simulovat dvěma čítači: při násobení/dělení 2 odečítáme 1 (resp. 2) z jednoho čítače a přičítáme 2 (resp. 1) k druhému.
- Postačí ovšem i čítače dva:
  - Obsah čtyř čítačů i, j, k, l je možné zakódovat jako $2^i 3^j 5^k 7^l$.
  - Přičtení/odečtení je pak možné realizovat násobením/dělením 2, 3, 5, či 7.

❖ Mezi další Turingovsky úplné výpočetní mechanismy pak patří např. λ-kalkul či parciálně-rekurzívní funkce (viz další přednáška).