11. Postův korespondenční problém. Redukce. Algoritmicky nerozhodnutelné problémy z teorie jazyků.

Algoritmicky nerozhodnutelné problémy

Nerozhodnutelnost nějakého problému znamená, že neexistuje algoritmus schopný tento problém řešit.

Definicí pojmu algoritmus se zabývá tzv. **Churchova teze**, která říká:

Každý proces, který lze intuitivně nazvat algoritmem lze simulovat na Turingově stroji.

I když jde jenom o tezi, podporují její platnost následující argumenty:

- Všechny ostatní matematické formalismy jsou ekvivalentní co do výpočetní síly TS.
- Všechna možná rozšíření TS nemají vliv na výpočetní sílu.
- Není algoritmus, který by nešlo simulovat na TS.

Pro důkaz, že nějaký problém není algoritmicky řešitelný (tj. problém není rozhodnutelný) lze použít dvě matematické metody: *diagonalizace* a *redukce*.

Tyto postupy využívají schopnost TS simulovat práci jiného TS a to tak, že TS a nějaké jeho vstupní slovo je zakódováno do formy vstupu a následně je jeho práce simulována.

Příklady algoritmicky nerozhodnutelných problémů:

Pro daný program a přesnou specifikaci co má tento program dělat. Potřebujeme verifikovat, zda program skutečně dělá to, co od něj očekáváme. Tento úkol bychom chtěli automatizovat. Tento typ problémů není obecně rozhodnutelný tj. řešitelný počítačem.

Tento problém lze přeformulovat tak, že máme daný nějaký TS *M*, a slovo *w* zjistit zda *M* slovo akceptuje nebo neakceptuje (nebo cyklí).

Tento problém se také nazývá **problémem příslušnosti.** Pro daný TS *M* a slovo *w* chceme určit, zda slovo *w* buď přísluší nebo nepřísluší do jazyka *L(M)* (tj. zda slovo *w* akceptuje nebo neakceptuje).

PP = { <M>#<w> | stroj *M* akceptuje *w* }

Tento problém je částečně rozhodnutelný, ale není rozhodnutelný.

Důkaz částečné nerozhodnutelnosti se podá snadno sestrojením univerzálního TS, který na vstup dostane řetězec <M>#<w>. K důkazu nerozhodnutelnost, lze použít například Cantorovu diagonalizační metodu.

Redukce

Příklad redukce

Problém zastavení. Pro libovolný daný TS *M* a libovolné slovo *w* určit zda výpočet *M* na *w* je konečný či nikoliv. Tento problém je jen částečně rozhodnutelný.

PZ = { <M>#<w> | výpočet *M* na *w* je konečný }, tj. jazyk PZ není rekurzivní.

Důkaz nerozhodnutelnosti je podán tzv. **redukcí.**

Předpokládejme, že problém je rozhodnutelný, pak existuje úplný TS *M* akceptující PZ. Redukce spočívá na ukázání, že s takovým strojem lze sestrojit i úplný TS *N* akceptující jazyk PP (Problém příslušnosti).

Důkaz se provede sporem. Předpokládejme, že existuje úplný TS *T,* akceptující PZ. Pokud by takový stroj existoval, bylo by možné s jeho pomocí sestrojit úplný TS *N* akceptující jazyk PP. Jelikož PP není rekurzivní, dojdeme ke sporu.

Stroj *N* by pracoval takto:

1. Simuloval by výpočet stroje *T* na vstupu <M>#<w>.
2. Pokud by stroj akceptoval, pak by byl výpočet konečný (varianta cyklení je vypuštěna) a stroj pak může rovnou simulovat výpočet stroje *M* na *w*. Akceptuje jen pokud *N* akceptuje.
3. Pokud by stroj neakceptoval <M>#<w>, pak víme, že i výpočet *M* na *w* cyklil, a stačí zamítnout.

Stroj *N* je úplný, protože jsou simulovány jen konečné výpočty, cyklení je úplně odstraněno.

Definice redukce

Nechť A a B jsou jazyky s nějakými i různými i prázdnými abecedami. Redukce jazyka A na jazyk B je rekurzivní funkce z jedné abecedy do druhé, která všem slovům z první abecedy přiřazuje slova z abecedy druhé a naopak.

Vlastnosti redukce:

1. Existence úplného Turingova stroje, který pro každé slovo z první abecedy vypočte jeho obraz v druhé abecedě.
2. Redukce zachovává příslušnost do jazyka. Slovo z prvního jazyka se zobrazí jako slovo z druhého jazyka. Slovo mimo první jazyk se zobrazí jako slovo mimo druhý jazyk.
3. Není-li první jazyk rekurzivně spočetný, pak ani druhý jazyk není rekurzivně spočetný. (Je-li druhý jazyk rekurzivně spočetný, pak je i první jazyk rekurzivně spočetný.)
4. Není-li první jazyk rekurzivní, pak ani druhý jazyk není rekurzivní. (Je-li druhý jazyk rekurzivní, pak je i první jazyk rekurzivní.)

Důkaz nerozhodnutelnosti problému P metodou redukce se skládá ze dvou kroků.
1. Zvolíme nějaký problém N, o kterém už bylo dokázáno, že je nerozhodnutelný.
2. Prokážeme, že $N \leq P$.

Redukci můžeme, samozřejmě, využít i k důkazu rozhodnutelnosti nějakého problému P. V takovém případě
1. Zvolíme nějaký problém R, o kterém už bylo dokázáno, že je rozhodnutelný.
2. Prokážeme, že $P \leq R$.

Konstrukce redukce $A \leq B$ se skládá z následujících kroků. Nechť $A \subseteq \_*$, $B \subseteq \_*$.
1. Definujeme funkci $\sigma : \_* \rightarrow \_*$.
2. Ověříme, že funkce σ je rekurzivní (například tak, že zkonstruujeme úplný Turingův stroj (tj. algoritmus), který pro každé $w \in \_*$ vypočte $\sigma(w)$).
3. Ověříme platnost ekvivalence
$w \in A \Leftrightarrow \sigma(w) \in B$.

Rozhodnutelné problémy Pro TM

Pro daný TS *M* rozhodnout zda:

- M má alespoň *X* stavů.
- Výpočet stroje M nad vstupním slovem a^X je delší než *X*.
- Existuje slovo *w* takové, že výpočet stroje *M* nad vstupním slovem *w* je delší než nějaké *X*.

Semirozhodnutelné problémy Pro TM

Pro daný TS *M* rozhodnout zda:

- jazyk *L(M)* je neprázdný
- jazyk *L(M)* obsahuje alespoň *X* slov.

Nerozhodnutelné problémy Pro TM

Pro daný TS *M* rozhodnout zda:

- jazyk *L(M)* je prázdný
- jazyk *L(M)* obsahuje nanejvýš *X* slov
- jazyk *L(M)* je konečný
- jazyk $L(M) = R$ pro libovolný regulární jazyk *R*
- jazyk *L(M)* je regulární (tj. zda existuje regulární jazyk R takový, že $L(M) = R$)
- jazyk *L(M)* je rekurzivní (tj. zda existuje rekurzivní jazyk *L* takový, že $L(M) = L$, tj. problém, zda *M* je úplný TS).

Nerozhodnutelné problémy z teorie jazyků.

Problém prázdnosti

Je formulován jako úloha pro libovolnou danou gramatiku $\mathcal{G}$ rozhodnout, zda $L(\mathcal{G}) = \emptyset$.

Věta 5.23. *Problém prázdnosti pro třídu* bezkontextových *gramatik je* rozhodnutelný.

Důkaz: Je obsažen v důkazu věty 3.9. □

Důsledek 5.24. *Problém prázdnosti pro třídu* regulárních *gramatik je* rozhodnutelný.

Důkaz: Protože každá regulární gramatika je současně bezkontextovou, na rozhodování problému můžeme použít algoritmus navržený pro bezkontextové gramatiky. □

Úvahu použitou v důkazu důsledku 5.24 můžeme lehce zobecnit. Nechť P je problém a S množina jeho instancí. Pak z rozhodnutelnosti problému P pro množinu instnací S plyne rozhodnutelnost tohoto problému pro každou množinu instancí S′, kde $S' \subseteq S$. Tvrzení *nemusí* platit pro množinu $S' \supset S$, což dokazuje následující věta.

Věta 5.25. *Problém prázdnosti pro třídu* kontextových *gramatik je* nerozhodnutelný.

Důsledek 5.26. *Problém prázdnosti pro třídu* gramatik typu 0 *je* nerozhodnutelný.

Důkaz: Z rozhodnutelnosti problému pro třídu gramatik typu 0 by plynula i rozhodnutelnost problému pro třídu kontextových gramatik, což je spor. □

Problém příslušnosti

Je formulován jako úloha rozhodnout pro libovolnou danou gramatiku $\mathcal{G}$ a libovolné dané slovo w, zda $w \in L(\mathcal{G})$. S tímto problémem jsme se setkali již jednou (věta 5.3) a víme tedy, že pro gramatiky typu 0 je nerozhodnutelný. Z lemmatu 4.21 plyne, že ke každé kontextové gramatice je možné zkonstruovat ekvivalentní *úplný* Turingův stroj, a proto problém příslušnosti pro kontextové gramatiky je rozhodnutelný.

Problém konečnosti

Je formulován jako úloha pro libovolnou danou gramatiku $\mathcal{G}$ rozhodnout, zda jazyk $L(\mathcal{G})$ je konečný.

Věta 5.27. *Problém konečnosti pro bezkontextové gramatiky je rozhodnutelný.*

Důkaz: Tvrzení je důsledkem lemmatu o vkládáni pro CFL (věta 3.24). Jazyk $L(\mathcal{G})$ je nekonečný tehdy a jen tehdy, když obsahuje slovo z délky $p < |z| \leq p + q$. □

Věta 5.28. *Problém konečnosti pro kontextové gramatiky je nerozhodnutelný.*

Postův korespondenční problém

Je to nerozhodnutelný problém, který vymyslel pan Emil Post v roce 1946.

Definice

Postův korespondenční problém (anglicky Post Correspondence Problem), zkráceně PKP (PCP), lze formulovat takto: jsou dány dva seznamy, $A = x_1, \ldots, x_n$ a $B = y_1, \ldots, y_n$, neprázdných slov nad abecedou Σ. Seznamy A, B nazýváme *instancí* (případem) PKP a označujeme $\langle A, B \rangle$. Daná instance PKP *má řešení*, právě když existuje konečná posloupnost přirozených čísel $i_1, i_2, \ldots i_k$, $k \geq 1$, taková, že

$$x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k} .$$

Posloupnost $i_1, i_2, \ldots i_k$ se nazývá *řešením* PKP. Postův korespondenční problém je formulován jako třída problémů rozhodnout pro libovolnou danou instanci PKP, zda má řešení.

Iniciální Postův korespondeční problém

Našim cílem je dokázat, že Postův korespondenční problém není rozhodnutelný. Použijeme metodu redukce a sestrojíme redukci problému příslušnosti pro TM (věta 5.3) na PKP. Protože sestrojit hledanou redukci přímo je poměrně (technicky) náročné, zjednodušíme si celý problém následovně. Definujeme *iniciální* Postův korespondenční problém (zkráceně inPKP) a prokážeme, že pokud je inPKP nerozhodnutelný, tak i PKP je nerozhodnutelný. Pak dokážeme, že inPKP je nerozhodnutelný.

Rozdíl ve formulaci inPKP a PKP je v tom, že u iniciálního Postova problému se pro danou instanci $\langle A, B \rangle$ ptáme, zda má řešení začínající číslem 1. Přesněji, instance $\langle A, B \rangle$ iniciálního Postova korespondečního problému má řešení právě když existuje posloupnost přirozených čísel $i_1, i_2, \ldots i_k$, $k \geq 0$, taková, že

$$x_1 x_{i_1} x_{i_2} \cdots x_{i_k} = y_1 y_{i_1} y_{i_2} \cdots y_{i_k} .$$

Lemma 5.18. *Z nerozhodnutelnosti iniciálního Postova korespondečního problému plyne nerozhodnutelnost Postova korespondečního problému.*

Příklad 5.14. *Nechť A, B jsou seznamy nad abecedou $\{a, b, c\}$,*

$$A = (b, cbb, ab, c) \qquad B = (bbc, b, a, bc).$$

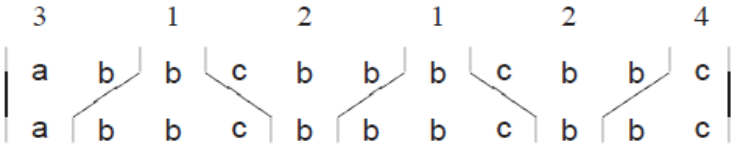
Pro lepší představu si instanci PKP můžeme znázornit jako kostky domina

$$\left\langle \frac{A}{B} \right\rangle = \left\{ \left[\frac{\mathsf{b}}{\mathsf{bbc}}\right], \left[\frac{\mathsf{cbb}}{\mathsf{b}}\right], \left[\frac{\mathsf{ab}}{\mathsf{a}}\right], \left[\frac{\mathsf{c}}{\mathsf{bc}}\right] \right\}$$

Řešením uvedené instance PKP je posloupnost 3,1,2,1,2,4 protože

$$x_3x_1x_2x_1x_2x_4 = y_3y_1y_2y_1y_2y_4 = abbcbbbcbbc.$$

Opět pro lepší představu uvádíme i grafickou prezentaci



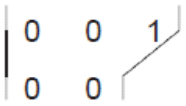
Uvedená posloupnost není jediným řešením daného případu; jsou jím např. i posloupnosti 3,1,2,1,2,1,2,4 a 3,1,2,1,2,4,3,1,2,1,2,4 a další.

Otázka 5.15. *Kolik řešení má instance PKP z předcházejícího problému?*

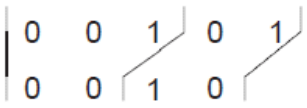
Příklad 5.16. *Mějme instanci PKP, danou seznamy A, B nad abecedou $\{0, 1\}$ takto:*

$$A = (01, 001, 11) \qquad B = (10, 00, 011).$$

Hledané řešení by muselo začínat indexem 2, protože dvojice x_1, y_1 a též x_3, y_3 se liší v již prvním symbolu (přesněji: v žádné z těchto dvojic není jedno ze slov předponou druhého). Tím máme



Ze seznamu B musíme nyní vybrat slovo, které začíná symbolem 1. Jedinou možností je slovo 10. Dostáváme



a to je situace shodná s předcházející. Proto tato instantce PKP nemá řešení (neexistuje konečná *posloupnost čísel požadovaných vlastností).*

Jsme tedy schopni pro některé konkrétní případy rozhodnout, zda mají řešení. Jak ovšem uvidíme, nelze napsat algoritmus, který by pro libovolnou instanci PKP rozhodoval, zda má či nemá řešení.

Příklad 5.17. *Iniciální Postův korespondeční problém pro seznamy A, B z příkladu 5.14 má řešení 122, protože $x_1x_1x_2x_2 = y_1y_1y_2y_2 = bbcbbcbb$.*