Teoretická informatika – Domáca úloha 2.

Michal Ľaš (xlasmi00)

8. decembra 2024

Príklad 1.

Jazyk $L_1 = \{\langle M\rangle\#\langle w\rangle | w \notin L(M) \wedge \langle M\rangle \in KA\}$ **je rekurzívny**, pretože vieme rozhodnúť či TS $M \in KA$ a pre takýto stroj bude vždy platiť, že je úplný (vždy zastaví), lebo vznikol z deterministického konečného automatu (DKA). Pravidlá prevodu DKA na TS ukazujú, že TS môže posúvať hlavu len doprava a neobsahuje pravidlá, ktoré prepisujú symboly pásky. Nehrozí teda vznik cyklenia. To nám zároveň implikuje, že vieme overiť či $w \notin L(M)$. Ako dôkaz tu je popis **úplného** TS T takého, že $L(T) = L_1$:

1. TS T má na svojom vstupe reťazec nad abecedou $\{0, 1, \#\}$
2. TS T overí či jeho vstup je v tvare $\langle M\rangle\#\langle w\rangle$, kde $\langle M\rangle$ je kód TS M a $\langle w\rangle$ je validný kód reťazca. Ak tomu tak nie je, tak TS T odmieta, inak pokračuje.
3. TS T overí, či TS M patrí do množiny KA (napríklad správnu štruktúru prechodovej funkcie δ_M). Ak tomu tak nie je, tak TS T odmieta, inak pokračuje.
4. TS T simuluje správanie stroja M na reťazci w. Ak M odmietne TS T akceptuje, ak M akceptuje TS T odmietne.
5. TS T zastaví na všetkých vstupoch, pretože problém $M \in KA$ je rozhodnuteľný a ak TS $M \in KA$, tak TS M je transformáciu nejakého DKA, ktorý zastavuje na všetkých vstupoch.
6. Preto L_1 je rekurzívny (rozhodnuteľný) jazyk.

Jazyk $L_2 = \{\langle M\rangle\#\langle w\rangle | w \notin L(M) \wedge \langle M\rangle \notin KA\}$ **nie je ani rekurzívne vyčísliteľný**. Rovnako ako v predošlom prípade vieme overiť, že TS $M \notin KA$, ale v tomto prípade môže byť TS M aj neúplný a môže cykliť, lebo TS $M \notin KA$. Navyše problém $w \notin L(M)$ je vlastne *co-MP* (doplnok k *membership (MP)* problému) a ten nie je ani rekurzívne vyčísliteľný. Ako dôkaz uvediem redukciu *co-HP* $\leq L_2$, čím sa ukáže, že L_2 je aspoň taký ťažký ako *co-HP* (co-HP je taktiež problém, ktorý nie je ani rekurzívne vyčísliteľný).

- Požadovaná redukcia je funkcia $\sigma : \{0, 1, \#\}^* \to \{0, 1, \#\}^*$ definovaná ako $\sigma(\langle M\rangle\#\langle w\rangle) = \langle M'\rangle\#\langle w'\rangle$.
- Pokiaľ vstup $\langle M\rangle\#\langle w\rangle$ nie je korektná inštancia *co-HP*, tak funkcia σ vracia kód TS M' taký, že $L(M') = \Sigma^*$, teda akceptuje každý vstup ($\langle M'\rangle\#\langle w'\rangle \notin L_2$). Inak σ vracia kód TS M', ktorý pracuje nasledovne:
  1. M' spustí simuláciu stroja M na vstupe w (kód M a w je priamo uložený v kóde M').
  2. Pokiaľ simulácia cyklí, tak M' cyklí pre každý svoj vstup w'.
  3. Pokiaľ simulácia skončí, tak M' akceptuje každý svoj vstup w'.

- TS M' **nevznikol** z DKA, zároveň obsahuje simuláciu stroja M, ktorý je inštanciou *co-HP* problému, teda platí, že $M' \notin KA$. Ďalej platí:

$$\langle M\rangle\#\langle w\rangle \in \text{co-HP} \Rightarrow w' \notin L(M'), \text{pretože } L(M') = \emptyset \wedge M' \notin KA \Rightarrow \langle M'\rangle\#\langle w'\rangle \in L_2$$
$$\langle M\rangle\#\langle w\rangle \notin \text{co-HP} \Rightarrow w' \in L(M'), \text{pretože } L(M') = \Sigma^* \wedge M' \notin KA \Rightarrow \langle M'\rangle\#\langle w'\rangle \notin L_2 \Leftrightarrow$$
$$\langle M\rangle\#\langle w\rangle \in \text{co-HP} \Leftrightarrow \sigma(\langle M\rangle\#\langle w\rangle) \in L_2.$$

- Popísanú konštrukciu TS M' je možné implementovať pomocou úplného TS a teda funkcia σ je totálna, rekurzívne vyčísliteľná funkcia.

Príklad 2.

Hľadanie riešenia pre hru Jána a Elišky (hra H) veľmi nápadne pripomína problém farbenia grafov (graph coloring problem), ktorý je NP-úplný. Redukciou Graph coloring problému (GC) na problém existencie riešenia pre hru H (GC $\leq$ H) sa ukáže, že problém existencie riešenia pre H je aspoň tak ťažký ako problém farbenia grafov. Algoritmus pre redukciu problému existencie riešenia pre H na problém farbenia grafov môže vyzerať nasledovne:
Vstup: máme graf $G = (V, E)$ a chceme rozhodnúť či G je k-farebný.
Prevod na hru H z grafu $G = (V, E)$:

- Vrcholy V grafu G prevedieme na kruhy K.
- Hrany E grafu G prevedieme na čiary C.
- Počet farieb $b = k$.
- Počet značiek x nastavíme na 1, čím zjednodušíme problém na základnú farebnosť. Každý kruh bude mať iba jednu značku, rovnako ako má každý vrchol jednu farbu pri probléme farbenia grafov.

Týmto spôsobov dostaneme problém farbenia grafov, ktorý je NP-úplný. Problém značkovania kruhov je teda aspoň taký ťažký ako problém farebnosti grafu. Preto je aj problém značkovania kruhov NP-úplný. Hra H je len zložitejšia verzia problému farbenia grafov, pretože vo vrcholoch môže byť viac kombinácií farieb a značiek.

Príklad 3.

Úlohu budem riešiť pomocou metódy účtov. Problém by som analyzoval nasledovne, môžu nastať 3 prípady:

1. Vo funkcií *get_next* **nedôjde** k presahu z $Z \rightarrow A$. Vtedy bude cena 8 operácií (riadky: 2,3,4,5,8,9,3,10).
2. Vo funkcií *get_next* **dôjde** k presahu/om zo $Z \rightarrow A$. Vtedy bude cena 8 operácií ako v predošlom bode $+4k$ operácií (riadky: 3,4,5,6), kde k značí počet znakov Z od konca reťazca.
3. Vo funkcií *get_next* **dôjde** k presahu zo reťazca $Z^l \rightarrow A^l$. Vtedy bude cena 3 operácie (riadky: 2,3,10) $+4l$ (riadky: 3,4,5,6), kde l je dĺžka reťazca.

Uvažujme klasickú abecedu, ktorá má 26 znakov. Každý 26. riadok bude presah a ten si musíme "predplatiť". Výnimka bude posledný presah, kde je cena o niečo nižšia.
Myšlienka je, že každých 26 volaní funkcie *get_next* si musíme predplatiť $\frac{4}{26}$. Každých 26×26 volaní si musíme predplatiť ďalších $\frac{4}{26}$ ako "dvojitý"presah. Každých $26 \times 26 \times 26$ si musíme ... A takto by sa dalo pokračovať. Takýto

Vstup	[A^l]	[A . . . AB]	. . .	[A . . . AZ]	[A . . . BA]	. . .	[A . . . AZZ]	[A . . . BAA]	. . .	[Z^l]
Cena	8	8		8	8		8	8		3
Navyše				4			2×4			4

Tabuľka 1: Ilustrácia cien iterácií funkcie *get_next* v úlohe 2.

výpočet sa dá vyjadriť ako $\sum_{i=1}^{n} \frac{4}{26^i} = 0,15\overline{9}$. **Každá operácia tak bude stáť $\mathbf{8 + 0,15\overline{9}}$** okrem prípadu z bodu 3., tá bude stáť $\mathbf{3 + 0,15\overline{9}}$. Túto myšlienku ilustruje tabuľka 1.
Predpokladajme, že znaky A až Z zakódujeme číslicami 0 až 25. Invariant by sa dal vyjadriť nasledovne:
Vstup: $[X_1, X_2, \ldots, X_n]$, kde $X_i \in \langle 0, 25 \rangle$ (zakódované písmená abecedy).
$X_n \times 0,15\overline{9} + 26 \times X_{n-1} \times \frac{0,15\overline{9}}{26} + 26^2 \times X_{n-2} \times \frac{0,15\overline{9}}{26^2} + \cdots = \sum_{i=0}^{n} X_i * 0,15\overline{9}$

Príklad *get_next(AZZ, 3)*:

- **Na účte:** $0 \times 0,15\overline{9} + 25 \times 0,15\overline{9} \times 0,15\overline{9} = 7,\overline{9}$
- **Cena operácie:** $8 + (2 \times 4) = 16$
- **Kredity:** $8 + 0,15\overline{9} = 8,15\overline{9}$

$7,\overline{9} + 8,15\overline{9} - 16 = 0,15\overline{9}$ čo zodpovedá invariantu.

Príklad *get_next(ZZZ, 3)*:

- **Na účte:** $25 \times 0,15\overline{9} + 25 \times 0,15\overline{9} \times 0,15\overline{9} = 11,\overline{9}$
- **Cena operácie:** $3 + (4 \times 3) = 15$
- **Kredit za inštrukciu:** $3 + 0,15\overline{9} = 3,15\overline{9}$

$11,\overline{9} + 3,15\overline{9} - 15 = 0,15\overline{9}$ čo zodpovedá invariantu.
Toto sa dá zobecniť na ľubovoľný vstup. Amortizovaná časová zložitosť n operácií je teda:
$n \times 8,15\overline{9}$ (t.j. $n \times max\{8 + 0.15\overline{9}, 3 + 0,15\overline{9}\}$) $\leq \mathbf{8, 16n} \in \mathbf{O(n)}$.

Príklad 4.

Časová zložitosť v najlepšom prípade je $O(n)$, pretože minimálne musíme spočítať všetky prvky poľa. Najlepší prípad teda je, že *final* sa rovná súčtu všetkých prvkov poľa.
Časová zložitosť v najhoršom prípade je $O(\frac{n(n+1)}{2}) \in O(n^2)$. Nastane v prípade, že súčet čísel v žiadnom suffixe sa nerovná hodnote *final*. Musíme prejsť všetky suffixy, ktorých je n a cez každý suffix preiterovať.
Algoritmus *find_opt(int *array, int size, int final)* je napísaný v Algorithm 1.
Jeho časová zložitosť v najhoršom prípade je $O(n)$ v prípade keď súčet čísel celého poľa sa rovná *final* alebo súčet čísel žiadneho suffixu sa nerovná *final*.

```
1 Function find_opt (int *array, int size, int final)
2     int i := size − 1
3     int tmp := 0
4     while i ≥ 0 do
5         tmp := tmp + array[i]
6         i := i − 1
7         if tmp = final then
8             return ANO
9     return NE
```

Algorithm 1: Algoritmus *find_opt*

Príklad 5.

a) Bezespornosť: Dokáže sa tak, že sa nájdeme model pre túto teóriu. Ak existuje model, kde všetky axiómy platia, tak teória je *bezesporná*. Model $\mathcal{M}_1$ tejto teórie by mohol vyzerať nasledovne (interpretácia I_1):

- Doména $\mathcal{D}_{I_1} = \{Kr, Da, Ve, Ku, Pe\}$
- Interpretácia funkčných symbolov: $\alpha_{I_1}(Kral_{/0}) = Kr$, $\alpha_{I_1}(Dama_{/0}) = Da$, $\alpha_{I_1}(Vez_{/0}) = Ve$, $\alpha_{I_1}(Kun_{/0}) = Ku$, $\alpha_{I_1}(Pesec_{/0}) = Pe$
- Interpretácia predikátových symbolov $\alpha_{I_1}(ohrozuje_{/2}) = \emptyset$

Overenie axiómov:

1. $\forall x(x = Kral \vee x = Dama \vee x = Vez \vee x = Kun \vee x = Pesec)$: Platí pre $\forall x \in \mathcal{D}_{I_1}$, každý funkčný symbol vracia prvok z množiny $\mathcal{D}_{I_1}$. Axióm teda platí.
2. $\forall x \neg ohrozuje(x, Kral)$: $ohrozuje(x, Kral) = false$ pre všetky $x \in \mathcal{D}_{I_1}$ v $\mathcal{M}_1$, lebo $\alpha_{I_1}(ohrozuje_{/2}) = \emptyset$. Tento axióm rovnako platí.
3. $\forall x, y(ohrozuje(x, y) \wedge ohrozuje(y, x) \Rightarrow x \neq Kun)$: V tomto modeli je $ohrozuje(x, y)$ vždy $false$ pre všetky $x, y \in \mathcal{D}_{I_1}$, takže predpoklad je vždy nepravdivý a implikácia je splnená pre všetky x, y.
4. $\forall x(ohrozuje(Pesec, x) \Rightarrow ohrozuje(x, Pesec) \vee x = Kun)$: Rovnako ako v predošlom bode, predpoklad implikácie je vždy nepravdivý, takže implikácia vždy platí.

Model pre teóriu existuje a teda teória je *bezesporná*.

b) Úplnosť: Teória nie je úplná, pretože vieme nájsť 2 modely a formulu, ktorá rozlíši tieto dva modely. Majme 2 modely $\mathcal{M}_1$ (z predchádzajúceho príkladu) a $\mathcal{M}_2$ definovanú nasledovne (interpretácia I_2):

- Doména $\mathcal{D}_{I_2} = \{Kr, Da, Ve, Ku, Pe\}$
- Interpretácia funkčných symbolov: $\alpha_{I_2}(Kral_{/0}) = Kr$, $\alpha_{I_2}(Dama_{/0}) = Da$, $\alpha_{I_2}(Vez_{/0}) = Ve$, $\alpha_{I_2}(Kun_{/0}) = Ku$, $\alpha_{I_2}(Pesec_{/0}) = Pe$
- Interpretácia predikátových symbolov $\alpha_{I_2}(ohrozuje_{/2}) = \{(Da, Ku)\}$

Overenie, že axiómy stále platia:

1. $\forall x(x = Kral \vee x = Dama \vee x = Vez \vee x = Kun \vee x = Pesec)$: Platí pre $\forall x \in \mathcal{D}_{I_2}$, každý funkčný symbol vracia prvok z množiny $\mathcal{D}_{I_2}$. Axióm teda platí.
2. $\forall x \neg ohrozuje(x, Kral)$: je $false$, pre všetky $x \in \mathcal{D}_{I_2}$, pretože iba $ohrozuje(Dama, Kun) = true$. Takže axióm stále platí.
3. $\forall x, y(ohrozuje(x, y) \wedge ohrozuje(y, x) \Rightarrow x \neq Kun)$: Axióm stále platí, pretože relácia $ohrozuje$ nie je symetrická ($ohrozuje(Kun, Dama) = false$).
4. $\forall x(ohrozuje(Pesec, x) \Rightarrow ohrozuje(x, Pesec) \vee x = Kun)$: Stále platí, pretože $ohrozuje(x, y) = true$ iba pre $x = Da$ a $y = Ku$. Predpoklad implikácie je tak vždy nepravdivý.

$\mathcal{M}_1$ aj $\mathcal{M}_2$ sú platné modely teórie. Majme teraz formulu $\varphi = \exists x, y(ohrozuje(x, y))$. $\mathcal{M}_1 \vDash \neg\varphi$, ale $\mathcal{M}_2 \vDash \varphi$. Formula φ má rôzne pravdivostné hodnoty v týchto dvoch modeloch. Preto teória T nie je úplná.

c) Rozhodnuteľnosť: Teória je rozhodnuteľná, pretože sice existuje viacero modelov, ale je ich konečne veľa a ak je nejaké formula vo všetkých modeloch platná, tak môžeme povedať, že $T \vDash \varphi$. Pokiaľ je formula φ v nektorom z modelov teórie T neplatná, tak $T \nvDash \varphi$. Popis algoritmu by mohol vyzerať nasledovne:

- Teória T má konečný počet funkčných symbolov: $\{Kral_{/0}, Dama_{/0}, Vez_{/0}, Kun_{/0}, Pesec_{/0}\}$ (5 konštánt) a $ohrozuje_{/2}$ je binárny predikát.
- Prvý axióm vynucuje, že hodnoty domény musia byť jeden z funkčných symbolov, teda najviac päť rôznych hodnôt. Za takejto podmienky je možné generovať všetky možné modely $\mathcal{M}$ pre teóriu T. Počet možných modelov je konečný. Existuje $5 \times 5 = 25$ možných dvojíc pre reláciu $ohrozuje$, teda 2^{25} možných spôsobov definovania relácie $ohrozuje$ a počet rôznych mapovaní funkčných symbolov je 5^5, takže $5^5 \times 2^{25}$.
- Axiómy teórie T sú konečné a dajú sa overiť v konečnom čase.
- Ak nejaká formula φ platí vo všetkých modeloch teórie T, kde platia axiómy teórie T, potom $T \vDash \varphi$. Ak v nejakom modeli formula φ neplatí, tak $T \nvDash \varphi$.
- Algoritmus (turingov stroj) by mohol pracovať nasledovne:
  1. Generácia všetkých možných modelov (konečný proces).
  2. Overenie axiómov teórie T pre každý model (konečný proces).
  3. Vyhodnotenie formule φ.
- Výstupom je odpoveď "áno", ak $T \vDash \varphi$ alebo "nie", ak $T \nvDash \varphi$.

Záver je, že teória T je rozhodnuteľná, lebo existuje vyššie popísaný algoritmus.