

Teoretická informatika (TIN) – 2024/2025

Úkol 2

(max. zisk 8 bodů – 10 bodů níže odpovídá 1 bodu v hodnocení předmětu)

- ✓ 1. Uvažujme abecedu Σ , t.z., symbol $R \notin \Sigma$, a následující kódování deterministického konečného automatu do Turingova stroje: pro $A = (Q, \Sigma, \delta, q_0, F)$ sestrojíme TS $M_{sim}(A) = (Q \cup \{q_0^M, q_f^M\}, \Sigma, \Sigma \cup \{\Delta\}, \delta_M, q_0^M, q_f^M)$, kde $Q \cap \{q_0^M, q_f^M\} = \emptyset$ a δ_M je definována následovně:

- $\delta_M(q_0^M, \Delta) = (q_0, R)$
- $\forall f \in F : \delta_M(f, \Delta) = (q_f^M, R)$
- $\delta_M(q, a) = (p, R) \Leftrightarrow \delta(q, a) = p$

Množina kódů turingových strojů vzniklých transformací DKA $KA = \{\langle M \rangle \mid M = M_{sim}(A)$ pro nějaký DKA $A\}$ je rozhodnutelná, protože lze jednoduše ověřit tvar přechodové funkce.

Rozhodněte a dokažte, zda následující jazyky jsou (resp. nejsou) rekurzivní (resp. rekursivně vyčíslitelné):

- $L_1 = \{\langle M \rangle \# \langle w \rangle \mid w \notin L(M) \wedge \langle M \rangle \in KA\}$
- $L_2 = \{\langle M \rangle \# \langle w \rangle \mid w \notin L(M) \wedge \langle M \rangle \notin KA\}$

20 bodů

- ✓ 2. Jan a Eliška si vymysleli novou hru. Mají barevné křídy o b ($b \geq 2$) barvách. Na chodník si nakreslili křídou kolečka a některá z nich propojili čárami. Teď by rádi do každého kolečka namalovali x ($x \geq 2$) barevných značek (barvy značek v jednom kolečku se mohou opakovat) tak, aby kolečka propojená čárou nebyla označena stejně. Pořadí značek v kolečku nehráje roli. Otázka zní, jestli je takového označení možné.

Formálně definujme hru Jana a Elišky jako n-tici $H = (K, C, b, x)$, kde

- K je konečná množina koleček,
- $C \subseteq \{\{a, b\} \mid a, b \in K\}$ je množina čar,
- $b \geq 2$ je počet barv,
- $x \geq 2$ požadovaný počet značek.

Hra H má řešení, pokud existuje zobrazení $O : K \times \langle 1, b \rangle \rightarrow \mathbb{N}$ takové, že

- $\forall a \in K : \sum_{i=1}^b O(a, i) = x$ (počet značek v jednom kolečku je roven x)
- $\forall \{a, b\} \in C \exists i : O(a, i) \neq O(b, i)$ (označení dvojice míst spojených čárou se liší)

Dokažte, že problém existence řešení pro hru H je NP-úplný.

(Pozn: Pomůže Vám NP-úplnost některého z problémů uvedených zde:

https://en.wikipedia.org/wiki/NP-completeness#NP-complete_problems

v odstavci „NP-complete problems“.)

15 bodů

- ✓ 3. Uvažujeme funkci `get_next`, která má na vstupu řetězec nad abecedou $\Sigma = \{A, \dots, Z\}$ a jeho délku l . Funkce vrátí následující řetězec vzhledem k lexikografickému uspořádání. Funkce `next_char` vrací následující znak latinské abecedy. Analyzujte a zdůvodněte amortizovanou časovou složitost libovolné posloupnosti n operací $str := get_next(str, l)$. Na začátku je zafixována konstanta $l > 0$ a počáteční hodnota $str = A^l$ (řetězec obsahující l symbolů A).

Předpokládejme uniformní cenové kriterium, kde každý řádek má cenu 1 (vyjímkou je řádek 7 s cenou 0).

```

1 Function get_next(char [] str, int l)
2   fin := false;
3   while ¬(fin) ∧ l > 0 do
4     l := l - 1;
5     if str[l] = Z then
6       str[l] := A;
7     else
8       next_char(str[l]);
9       fin := true;
10  return str;
```

15 bodů

- ✓ 4. Uvažujme funkci *find_suffix*, která má na vstupu pole čísel *array* o velikosti *size* (chybné vstupy neuvažujte) a která se snaží nalést v rámci pole *suffix* takový, že součet čísel v tomto suffixu je roven hodnotě *final*.

- Analyzujte časovou složitost funkce *find_suffix* v nejlepším případě
- Analyzujte časovou složitost funkce *find_suffix* v nejhorším případě.
- Navrhněte funkci *find_opt*, která bude dávat stejný výsledek jako funkce *find_suffix*, ale bude mít lepší asymptotickou složitost v nejhorším případě.
- Analyzujte časovou složitost funkce *find_opt* v nejhorším případě.

Předpokládejme uniformní cenové kriterium, kde každý řádek má cenu 1.

```

1 Function find_suffix(int * array, int size, int final)
2   | int i, j;
3   | i := 0;
4   | while i < size do
5   |   | j := i;
6   |   | int tmp := 0;
7   |   | while j < size do
8   |   |   | tmp := tmp + array[j];
9   |   |   | j := j + 1;
10  |   | if tmp = final then
11  |   |   | return ANO
12  |   |   | i := i + 1;
13  | return NE
```

15 bodů

- ✓ 5. Mějme teorii *T* se signaturou $\langle \{Kral_0, Dama_0, Vez_0, Kun_0, Pesec_0\}, \{ohrozuje_{/2}, =_{/2}\} \rangle$ (= je standardní rovnost) se speciálními axiomy

$$\begin{aligned} \forall x(x = Kral \vee x = Dama \vee x = Vez \vee x = Kun \vee x = Pesec) \\ \forall x \neg ohrozuje(x, Kral) \\ \forall x, y(ohrozuje(x, y) \wedge ohrozuje(y, x) \Rightarrow x \neq Kun) \\ \forall x(ohrozuje(Pesec, x) \Rightarrow ohrozuje(x, Pesec) \vee x = Kun) \end{aligned}$$

- i) Rozhodněte a stručně zdůvodněte, zda *T* je: a) bezesporná, b) úplná a c) rozhodnutelná (tj. množina důsledků *T* je rozhodnutelná).

15 bodů

4. Uvažujeme funkci `find_suffix`, která má na vstupu pole čísel `array` o velikosti `size` (chybné vstupy neuvažujte) a která se snaží nalézt v rámci pole suffix takový, že součet čísel v tomto suffixu je roven hodnotě `final`.

- Analyzuje časovou složitost funkce `find_suffix` v nejlepším případě $O(1)$ (je totožné s celkovou složitostí pole: $O(size)$)
- Analyzuje časovou složitost funkce `find_suffix` v nejhorším případě. $O(size^2)$ (počítání sumy všechny = final)

Navrhnete funkci `find_opt`, která bude dávat stejný výsledek jako funkce `find_suffix`, ale bude mít lepší asymptotickou složitost v nejhorším případě.

Analyzuje časovou složitost funkce `find_opt` v nejhorším případě.

Předpokládáme uniformní cenové kriterium, kde každý řádek má cenu 1.

```

1 Function find_suffix(int *array, int size, int final)
2     int i, j;
3     i := 0;
4     while i < size do
5         j := i;
6         int tmp := 0;
7         while j < size do
8             tmp := tmp + array[j];
9             j := j + 1;
10        if tmp = final then
11            | return AND
12            i := i + 1;
13        return NE

```

Složitosti zde

function find_opt (int *array, int size, int final) {

assert (size ≥ 2 ∧ size > 0);

int i = size - 1;

int tmp = array[i--];

while tmp < final AND i ≥ 0 {

tmp += array[i--];

}

return tmp == final;

}

$O(size)$ (~ poskládání se seřazené výdy max jednou)

3. Uvažujeme funkci `get_next`, která má na vstupu řetězec nad abecedou $\Sigma = \{A, \dots, Z\}$ a jeho délku l . Funkce vráci následující řetězec vzhledem k lexikografickému uspořádání. Funkce `next_char` vrací následující znak latinské abecedy. Analyzuje a zdůvodňete amortizovanou časovou složitost libovolné posloupnosti n operací $str := get_next(str, l)$. Na začátku je zafixována konstanta $l > 0$ a počáteční hodnota $str = A^l$ (řetězec obsahující l symbolů A).

Předpokládáme uniformní cenové kriterium, kde každý řádek má cenu 1 (vyjímkou je řádek 7 s cenou 0).

```

1 Function get_next(char [] str, int l)
2     fin := false;
3     while ¬(fin) ∧ l > 0 do
4         l := l - 1;
5         if str[l] = Z then
6             | str[l] := A;
7         else
8             | next_char(str[l]);
9         fin := true;
10    return str;

```

Předpokladám že $|\Sigma| = 26$

Uvažujeme rozdělení podle počtu itacií v `get_next`

	Počet iterací vlnky	Cena	Kreditky
Itv 1	1	8	$8 + \frac{4}{26}$
Itv 2	2	$8 + 4$	$8 + \frac{4}{26} + \frac{4}{26^2}$
Itv 3	3	$8 + 4 + 4$	$8 + \frac{4}{26} + \frac{4}{26^2} + \frac{4}{26^3}$
...
Itv h	h	$8 + h \cdot 4$	$8 + \frac{4}{26} + \frac{4}{26^2} + \frac{4}{26^3} + \dots$

$$\sum_{n=1}^{\infty} \frac{4}{26^n} = \frac{4}{25} \approx 0,16$$

n operací $8,16n$

složitost jedné operace vlnitě nulačí do 0 (1)

	Cena	Kreditky	POZDÍL
26 operací	$200 + 12$	$26 \cdot 8,16$	—
	212	212,16	0,32

Po 25 operacích byla cena 200 a kreditky ≈ 204 . 4 kreditky postoji pro zaplatení druhé, následující operace která je druhá prvně 0,4. kreditky zůstanou neklesají pod 0. je dodržen invariant

Amortizovaná složitost je $18,16n \in O(n)$

2. Jan a Eliška si vymysleli novou hru. Mají barevné křídla o b ($b \geq 2$) barvách. Na chodník si nakreslili křídou kolečka a některá z nich propojili čárami. Teď by rádi do každého kolečka namalovali x ($x \geq 2$) barevných značek (barvy značek v jednom kolečku se mohou opakovat) tak, aby kolečka propojená čárou nebyla označená stejně. Pořadí značek v kolečku nehraje roli. Otázka zní, jestli je takového označení možné.

Formálně definujme hru Jana a Elišky jako n-tici $H = (K, C, b, x)$, kde

- K je konečná množina koleček, Vchody
- $C \subseteq \{\{a, b\} \mid a, b \in K\}$ je množina čar, hrany
- $b \geq 2$ je počet barv,
- $x \geq 2$ požadovaný počet značek.

Hra H má řešení, pokud existuje zobrazení $O : K \times \langle 1, b \rangle \rightarrow \mathbb{N}$ takové, že

- $\forall a \in K : \sum_{i=1}^b O(a, i) = x$ (počet značek v jednom kolečku je roven x)
- $\forall \{a, b\} \in C \exists i : O(a, i) \neq O(b, i)$ (označení dvojice míst spojených čárou se liší)

Dokažte, že problém existence řešení pro hru H je NP-úplný.

(Pozn: Pomůže Vám NP-úplnost některého z problémů uvedených zde:

https://en.wikipedia.org/wiki/NP-completeness#NP-complete_problems

15 bodů

H problém patří do NP.

- NTS užitelné obecněji a
- V polynomické čase ověříme, že se jedná o validní řešení

H je NP-těžký:

- užíváme redukce: zadání problému (ℓ -obvatitelnost) $\leq_p H$
- redukční funkce $f : f((V, E, \ell)) = (\overline{E}, C, b, x)$
- $V = \overline{E} \wedge C = E$
- $\ell = \binom{b+x-1}{x}$ - 2 kombinace barev a známké utvoříme barvu.
- $(V, E, \ell) \in \ell$ -obvatitelnost
- tuto redukci lze provést v polynomickém čase □

1. Uvažujme abecedu Σ , t.z., symbol $R \notin \Sigma$, a následující kódování deterministického konečného automatu do Turingova stroje: pro $A = (Q, \Sigma, \delta, q_0, F)$ se stojíme $TS_{sim}(A) = (Q \cup \{q_0^M, q_f^M\}, \Sigma, \Sigma \cup \{\Delta\}, \delta_M, q_0^M, q_f^M)$, kde $Q \cap \{q_0^M, q_f^M\} = \emptyset$ a δ_M je definována následovně:

- $\delta_M(q_0^M, \Delta) = (q_0, R)$
- $\forall f \in F : \delta_M(f, \Delta) = (q_f^M, R)$
- $\delta_M(q, a) = (p, R) \Leftrightarrow \delta(q, a) = p$

Množina kódů turingových strojů vzniklých transformací DKA $KA = \{\langle M \rangle \mid M = M_{sim}(A)$ pro nějaký DKA $A\}$ je rozhodnutelná, protože lze jednoduše ověřit tvar přechodové funkce.

Rozhodněte a dokažte, zda následující jazyky jsou (resp. nejsou) rekursivní (resp. rekursivně vyčíslitelné):

- $L_1 = \{\langle M \rangle \# \langle w \rangle \mid w \notin L(M) \wedge \langle M \rangle \in KA\}$
- $L_2 = \{\langle M \rangle \# \langle w \rangle \mid w \notin L(M) \wedge \langle M \rangle \notin KA\}$

20 bodů

$L_1 \in \mathcal{L}_{DEC}$ protože lze sestrojit TS kódující tento jazyk a pro jazyk w vypídat odpovídající číslo.

TS bude provádět následovně:

- ① TS zkouší, zda na vstupu je požadovaný vstup " $\langle M \rangle \# \langle w \rangle$ ".
- ② TS zkouší, zda se M nachází v množině KA . Pokud ano, vrací se také odpověď.
- ③ TS simuluje TS w pro svůj M . Vzhledem k tomu že M je simulace DKA tak v řešení odpovídá zda $w \in L(M)$ a i $w \notin L(M)$.
- ④ Pokud platí že $w \notin L(M)$ vrací se také odpověď.

$$L_2 \notin \mathcal{L}_{DEC} \cap L_2 \in \mathcal{L}_{RE}$$

Dokážete to pomocí redukce

$$\text{OTP} \leq L_2 \quad \Gamma(\langle n \rangle \# \langle w \rangle) = \langle n' \rangle \# \langle w' \rangle$$

Chování TS n' : Pokud přijme vstupní instanci celého n' přijme w'

- ① pokud vstup $w' \in \{a^n b^n \mid n \geq 1\}$ tak přijme.
- ② TS spustí simulaci TS n s vstupem w .
- ③ Pokud n cyklu, myslí i n' .
- ④ Pokud simulace skončí, ts přijme w' .

$$n \text{ cyklu} \Rightarrow L(n') = \{a^n b^n \mid n \geq 0\} \text{ a } w' \notin L(n')$$

$$n \text{ zastaví} = L(n') = \Sigma^* \text{ a } w' \in L(n')$$

5. Mějme teorii T se signaturou $\langle \{Kral_0, Dama_0, Vez_0, Kun_0, Pesec_0\}, \{ohrozuje_2, =_2\} \rangle$ (= je standardní rovnost) se speciálními axiomy

$$\begin{aligned}\forall x(x = Kral \vee x = Dama \vee x = Vez \vee x = Kun \vee x = Pesec) \\ \forall x \neg ohrozuje(x, Kral) \\ \forall x, y(ohrozuje(x, y) \wedge ohrozuje(y, x) \Rightarrow x \neq Kun) \\ \forall x(ohrozuje(Pesec, x) \Rightarrow ohrozuje(x, Pesec) \vee x = Kun)\end{aligned}$$

i) Rozhodněte a stručně zdůvodněte, zda T je: a) bezesporná, b) úplná a c) rozhodnutelná (tj. množina důsledků T je rozhodnutelná).

15 bodů

Ⓐ Teorie je bezesporná pokud má model: má např takový model $I(D, d)$

$$D = \{P, K, V, D, H\}$$

$$d(ohrozuje) = \{(P, H), (H, V), (V, D)\}$$

$$d(Pesec) = P$$

$$d(Kun) = H$$

$$d(Vez) = V$$

$$d(Dama) = D$$

$$d(Kral) = K$$



Ⓑ Existuje i důkaz model

$$I_1(D_1, d_1)$$

$$D_1 = \{P, K, V, D, H\}$$

$$d_1(ohrozuje) = \emptyset$$

$$d_1(Pesec) = P$$

$$d_1(Kun) = H$$

$$d_1(Vez) = V$$

$$d_1(Dama) = D$$

$$d_1(Kral) = K$$

Výsledek: ohrozuje(Pesec, Kun)

$\forall I$ platí $a \vee f_I$ neplatí

Ⓒ Výrok je rozhodnutelný. Počet axiomů je konečný i počet modelů (kmen isomorfismus). Musí existovat tis.