Teoretická informatika TIN

Sbírka příkladů

Milan Češka et al.

Verze z 11.12. 2023

Poděkovaní: Některé příklady byly převzaty ze sbírky Černá et al. Formální jazyky a automaty na FI MUNI.

Obsah

1 Formální jazyky a základní operace **2**

2 Základní konstrukce pro regulární jazyky **4**

3 Transformace konečných automatů a regulárních gramatik **4**

4 Rovnice nad regulárními výrazy **8**

5 Pumping lemma **9**

6 Myhill-Nerodova věta **10**

7 Uzávěrové vlastnosti regulárních jazyků **11**

8 Rozhodnutelné problémy pro regulární jazyky **11**

9 Základní konstrukce pro bezkontextové jazyky **12**

10 Transformace bezkontextových gramatik **13**

11 Jednoznačnost a determinismus bezkontextových gramatik **15**

12 Fix-point algoritmy pro bezkontextové gramatiky **16**

13 Pumping lemma pro bezkontextové jazyky **17**

14 Uzávěrové vlastnosti bezkontextových gramatik **17**

15 Základní konstrukce Turingových strojů **18**

16 Konstrukce vícepáskových Turingových strojů **18**

17 Důkazy (částečné) rozhodnutelnosti **19**

18 Rozhodnutelnost a redukce, diagonalizace **19**

19 Logika a rozhodnutelnost, neúplnost **21**

20 Asymptotická složitost **24**

21 Konstrukce TS s danou složitostí **25**

22 Složitost algoritmů a amortizovaná složitost **25**

23 Třídy složitosti a NP-úplnost **27**

Použitá notace

- $\#_a(w)$ – počet symbolů $a \in \Sigma$ v řetězci $w \in \Sigma^*$
- `mod` – operace modulo
- w^R – reverze řetězce $w \in \Sigma^*$
- $\mathcal{L}_3$ – třída regulárních jazyků
- $\mathcal{L}_2$ – třída bezkontextových jazyků
- $\mathcal{L}_1$ – třída kontextových jazyků
- $\mathcal{L}_{REC}$ – třída rekurzivních jazyků
- $\mathcal{L}_0$ – třída rekurzivně vyčíslitelných jazyků
- $\texttt{card}(M)$ – kardinalita (velikost) množiny M
- $<M>$ – značí kód TS (např. viz. konstrukci univerzálního TS)
- $<n>$ – značí kód čísla $n \in \mathbb{N}$
- $<w>$ – značí kód řetězce $w \in \Sigma^*$
- $\texttt{steps}(M, w)$ – počet kroků, který TS M provede na slově w

1 Formální jazyky a základní operace

1. Nechť $L_1 = \{\epsilon, a, b, bb\},\ L_2 = \{a, aa, bb\}$. Určete výčtem všech řetězců jazyky $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \setminus L_2$, $L_1 \cdot L_2$, $L_2 \cdot L_1$, L_1^2 a prvních 10 prvků jazyka L_1^* v uspořádání podle délky řetězců.

   Řešení:
   $L_1 \cup L_2 = \{\epsilon, a, b, bb, aa\}$
   $L_1 \cap L_2 = \{a, bb\}$
   $L_1 \setminus L_2 = \{\epsilon, b\}$
   $L_1 \cdot L_2 = \{a, aa, bb, aaa, abb, ba, baa, bbb, bba, bbaa, bbbb\}$
   $L_2 \cdot L_1 = \{a, aa, bb, ab, abb, aaa, aab, aabb, bba, bbb, bbbb\}$
   $L_1^2 = \{\epsilon, a, b, aa, ab, abb, ba, bb, bbb, bba, bbbb\}$
   $L_1^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, abb \ldots\}$

2. Popište (neformálně) tyto jazyky a jejich vzájemný vztah:

   (a) $L_1 = \{a, b\}^* \cdot \{b, a\}^*$
   (b) $L_2 = \{a, b\}^* \cdot \{a\}^* \cdot \{b\}^*$
   (c) $L_3 = \{a\}^* \cdot \{a, b\}^*$

   Řešení: $L_1 = L_2 = L_3 = \{a, b\}^*$ tj. jazyk všech slov na abecedou $\{a, b\}$.

3. Uspořádejte (podle množinové inkluze) tyto jazyky:

   (a) $L_1 = \{a, b\}^*$
   (b) $L_2 = \{a, b, c\}^*$
   (c) $L_3 = \{a\}^*\{b\}^*\{c\}^*$
   (d) $L_4 = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$
   (e) $L_5 = \{w \in \{a, b, c\}^* \mid \#_a(w) \geq \#_b(w)\}$
   (f) $L_6 = \{w \in \{a, b, c\}^* \mid \exists n > 2 : \#_a(w)^n + \#_b(w)^n = \#_c(w)^n\}$

Řešení: Platí následující vztahy

$L_2 \supset L_1 \supset L_4$
$L_2 \supset L_3$
$L_2 \supset L_5 \supset L_4$
$L_2 \supset L_6$

4. Pomocí jazyků $L_0 = \{0\}$ a $L_1 = \{1\}$ nad abecedou $\Sigma = \{0, 1\}$ a základních jazykových operací zkonstruujte následující jazyky:

   (a) $L_2 = \{w \in \Sigma^* \mid w$ obsahuje maximálně tři výskyty symbolu 1 $\}$

   (b) $L_3 = \{w \in \Sigma^* \mid w$ vyjadřuje liché číslo v binární notaci (bez vedoucích 0) $\}$

   (c) $L_4 = \{w \in \Sigma^* \mid w$ obsahuje buď jeden anebo sudý počet symbolů 1 $\}$

   (d) $L_5 = \{w \in \Sigma^* \mid w$ obsahuje sudý počet podřetězců "01"nebo "10"$\}$

   Řešení příkladu a):

$$L_2 = L_0^* \cup L_0^* L_1 L_0^* \cup (L_0^* L_1 L_0^*)^2 \cup (L_0^* L_1 L_0^*)^3$$

5. Uvažme $\Sigma = \{a, b\}$. Rozhodněte a dokažte, zda platí následující tvrzení:

   (a) $\forall L_1, L_2, L_3 \subseteq \Sigma^* : L_1 \subseteq L_2 \iff L_1 \cdot L_3 \subseteq L_2 \cdot L_3$

   (b) $\exists L_1, L_2, L_3 \subseteq \Sigma^* : L_1 \subseteq L_2 \iff L_1 \cdot L_3 \subseteq L_2 \cdot L_3$

   (c) $\forall L_1, L_2 \subseteq \Sigma^* : L_1$ a L_2 jsou konečné $\iff L_1 \cdot L_2$ je konečný

   (d) $\exists L_1, L_2 \subseteq \Sigma^* : L_1$ a L_2 jsou konečné $\iff L_1 \cdot L_2$ je konečný

   (e) $\forall L_1 \subseteq \Sigma^* : L_1^*$ je nekonečný

   (f) $\exists L_1 \subseteq \Sigma^* : L_1^*$ je nekonečný

   (g) $\forall L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$

   (h) $\exists L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$

   (i) $\forall L_1 \subseteq \Sigma^* : (L_1^*)^* = L_1^*$

   (j) $\exists L_1 \subseteq \Sigma^* : (L_1^*)^* = L_1^*$

   (k) $\forall L_1 \subseteq \Sigma^* : L_1^* = L_1^+ \iff \varepsilon \in L_1$

   (l) $\exists L_1 \subseteq \Sigma^* : L_1^* = L_1^+ \iff \varepsilon \in L_1$

   Řešení příkladu a): Toto tvrzení neplatí, zejména neplatí tato implikace

$$\forall L_1, L_2, L_3 \subseteq \Sigma^* : L_1 \cdot L_3 \subseteq L_2 \cdot L_3 \Rightarrow L_1 \subseteq L_2$$

   Uvažme například jazyky $L_1 = \{a\}, L_2 = \{b\}$ a $L_3 = \emptyset$. Pak $L_1 \cdot L_3 = L_2 \cdot L_3 = \emptyset$ ale neplatí, že $L_1 \subseteq L_2$.

   Řešení příkladu b): Toto tvrzení platí. Stačí zvolit například $L_1 = L_2 = \{a\}$ a $L_3 = \{\epsilon\}$. Pak $L_1 \cdot L_3 = L_2 \cdot L_3 = L_1 = L_2 = \{a\}$.

6. *Určete všechny jazyky $L \subseteq \{a, b, c\}^*$, pro které platí $L^* = L$, a uspořádejte je dle množinové inkluze.

   Řešení: Takových jazyků je nekonečně mnoho. Patří sem například jazyky $\{\epsilon\} \subset \{a\}^* \subset \{a, b\}^* \subset \{a, b, c\}^*$. Například je zřejmé, že $(\{a\}^*)^* = \{a\}^*$. Dále sem patří například jazyky $\{a^i\}^*$ pro libovolné i. Toto nám dává existenci nekonečně mnoha těchto jazyků. Otázkou zůstává jestli těchto jazyků je spočetně mnoho – tento problém je nad rámec našich znalostí.

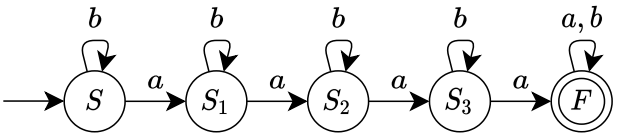
2 Základní konstrukce pro regulární jazyky

1. Zkonstruujte konečné automaty a regulární gramatiky pro tyto jazyky:

   (a) $L = \{w \in \{a,b\}^* \mid \#_a(w) > 3\}$
   (b) $L = \{w \in \{a,b\}^* \mid \#_a(w) < 3\}$
   (c) $L = \{w \in \{a,b\}^* \mid \#_a(w) \bmod 3 = \#_b(w) \bmod 2\}$
   (d) $L = a^* \cdot (bb + cc)^* \ (\{a\}^* \cdot \{bb, cc\}^*)$

   Řešení příkladu a):

   KA A, $L(A) = L$:



   Gramatika G, $L(G) = L$:

   $G = (\{S, S_1, S_2, S_3, F\}, \{a, b\}, P, S)$ kde P:

$$\begin{aligned} S &\to bS \mid aS_1 \\ S_1 &\to bS_1 \mid aS_2 \\ S_2 &\to bS_2 \mid aS_3 \\ S_3 &\to bS_3 \mid aF \\ F &\to aF \mid bF \mid \varepsilon \end{aligned}$$

2. Pomocí regulárních výrazů popište tyto jazyky:

   (a) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) > 2\}$
   (b) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) < 2\}$
   (c) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) \bmod 2 = 0\}$

   Řešení příkladu a):

$$L = (b+c)^* a (b+c)^* a (b+c)^* a (a+b+c)^*$$

3 Transformace konečných automatů a regulárních gramatik

1. Převeďte následující nedeterministické automaty na ekvivalentní deterministické automaty. Postupujte dle algoritmu z přednášky. Není třeba uvádět nedosažitelné stavy

   (a) $A = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$, kde δ je definována jako

$$\begin{aligned} \delta(q_0, a) &= \{q_0, q_1\}, & \delta(q_0, b) &= \{q_0\}, \\ \delta(q_1, a) &= \{q_2\}, & \delta(q_1, b) &= \{q_0, q_1\}, \\ \delta(q_2, a) &= \emptyset, & \delta(q_2, b) &= \emptyset. \end{aligned}$$

   (b) $A = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$, kde δ je definována jako

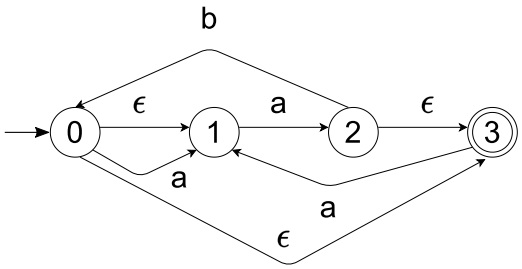
$$\begin{aligned} \delta(q_0, a) &= \{q_0\}, & \delta(q_0, b) &= \{q_0, q_1, q_2\}, \\ \delta(q_1, a) &= \{q_2\}, & \delta(q_1, b) &= \{q_0\}, \\ \delta(q_2, a) &= \emptyset, & \delta(q_2, b) &= \{q_2\}. \end{aligned}$$

Řešení příkladu a):

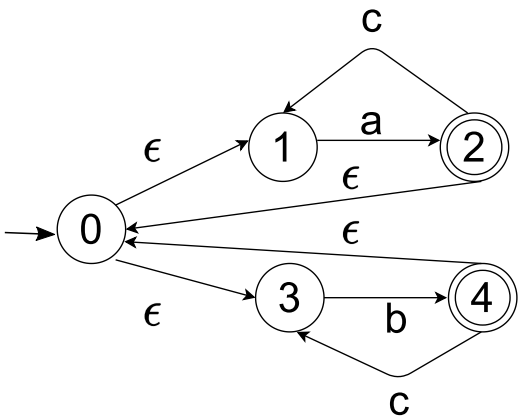
		a	b
$\rightarrow$	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$
$\leftarrow$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$

2. Převeďte následující rozšířené automaty (tj. automaty s ϵ-kroky) na ekvivalentní deterministické automaty. Postupujte dle algoritmu z přednášky. Není třeba uvádět nedosažitelné stavy

   (a) Automat na obrázku níže.



   (b) Automat na obrázku níže.



Řešení příkladu a):

iniciální stav $A = \epsilon$-uzávěr$(\{0\}) = \{0, 1, 3\}$
$\delta(A, a) = \epsilon$-uzávěr$(\{1, 2\}) = \{1, 2, 3\} = B$
$\delta(A, b) = \epsilon$-uzávěr$(\emptyset) = \emptyset = C$
$\delta(B, a) = \epsilon$-uzávěr$(\{1, 2\}) = \{1, 2, 3\} = B$
$\delta(B, b) = \epsilon$-uzávěr$(\{0\}) = \{0, 1, 3\} = A$
$\delta(C, a) = \delta(C, b) = C$
$F = \{A, B\}$

3. Následující automaty zadané tabulkou převeďte do redukované formy (tj. zkonstruujte ekvivalentní minimální automat

(a)

	a	b
1	3	1
→ 2	9	4
3	-	1
← 4	9	4
5	8	5
6	5	4
← 7	6	9
8	11	-
9	7	9
10	12	3
11	8	1
12	-	10

(b)

	a	b
↔ 1	3	2
2	6	4
3	3	5
← 4	4	2
5	10	8
6	6	7
← 7	7	5
← 8	8	2
← 9	11	2
10	10	9
11	11	5

Řešení příkladu a):

Nejdříve automat zúplníme, tj. dodáme přechody do nekoncového stavu 13:

$$\delta(3,a) = 13 \quad \delta(8,b) = 13 \quad \delta(12,a) = 13 \quad \delta(13,a) = \delta(13,b) = 13$$

Dále spočítáme množinu dosažitelných stavů Q z iniciálního stavu 2.

$$Q = \{1,2,3,4,5,6,7,8,9,11,13\}$$

$\equiv_0$

		a	b
→ I	1	I	I
	2	I	II
	3	I	I
	5	I	I
	6	I	II
	8	I	I
	9	II	I
	11	I	I
	13	I	I
← II	4	I	II
	7	I	I

$\equiv_1$

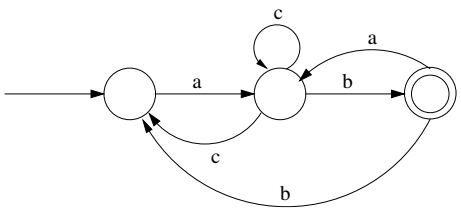
		a	b
I	1	I	I
	3	I	I
	5	I	I
	8	I	I
	11	I	I
	13	I	I
→ II	2	III	IV
	6	I	IV
III	9	V	III
← IV	4	III	IV
← V	7	II	III

$\equiv_2 = \equiv_3$

		a	b
I	1	I	I
	3	I	I
	5	I	I
	8	I	I
	11	I	I
	13	I	I
→ II	2	IV	V
III	6	I	V
IV	9	VI	IV
← V	4	IV	V
← VI	7	III	IV

4. Následující konečné automaty převeďte na ekvivalentní regulární gramatiky

   (a) M_1 z obrázku 1:

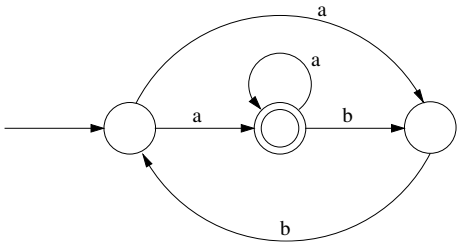


Obrázek 1: KA M_1

   (b) M_2 z obrázku 2:

Řešení příkladu a): $G = (\{S_1, S_2, S_3\}, \{a, b, c\}, P, S_1)$ kde P :

$$\begin{aligned} S_1 &\rightarrow aS_2 \\ S_2 &\rightarrow cS_2 \mid cS_1 \mid bS_3 \\ S_3 &\rightarrow aS_2 \mid bS_1 \mid \epsilon \end{aligned}$$



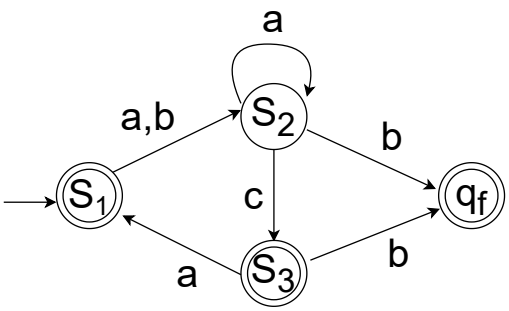
Obrázek 2: KA M_2

5. Následující regulární gramatiky převeďte na ekvivalentní konečné automaty.

   (a) $G = (\{S_1, S_2, S_3\}, \{a, b, c\}, P, S_1)$ kde P :
   $$S_1 \rightarrow \epsilon \mid aS_2 \mid bS_2$$
   $$S_2 \rightarrow cS_3 \mid aS_2 \mid b$$
   $$S_3 \rightarrow \epsilon \mid aS_1 \mid b$$

   (b) $G = (\{S_1, S_2, S_3\}, \{a, b, c\}, P, S_1)$ kde P :
   $$S_1 \rightarrow a \mid bS_1 \mid bS_3$$
   $$S_2 \rightarrow \epsilon \mid aS_2 \mid b$$
   $$S_3 \rightarrow aS_2 \mid c$$

   Řešení příkladu a): Na obrázku.



6. Následující regulární výrazy převeďte na ekvivalentní regulární gramatiky

   (a) $a^+ + (ab)^*$

   (b) $((a + b)^+c)^*$

   Řešení příkladu a):

   $R = a^+ + (ab)^* = aa^* + (ab)^*$
   $G_a = (\{S_a\}, \{a\}, \{S_a \rightarrow a\}, S_a)$
   $G_b = (\{S_b\}, \{b\}, \{S_b \rightarrow b\}, S_b)$
   $G_{a^*} = (\{S_{a^*}\}, \{a\}, \{S_{a^*} \rightarrow \epsilon, S_{a^*} \rightarrow aS_{a^*}\}, S_{a^*})$ po odstranění zbytečných symbolů
   $G_{a^+} = (\{S_a, S_{a^*}\}, \{a\}, \{S_a \rightarrow aS_{a^*}, S_{a^*} \rightarrow \epsilon, S_{a^*} \rightarrow aS_{a^*}\}, S_a)$
   $G_{ab} = (\{S_a, S_b\}, \{a, b\}, \{S_a \rightarrow aS_b, S_b \rightarrow b\}, S_a)$
   $G_{(ab)^*} = (\{S_{(ab)^*}, S_b\}, \{a, b\}, \{S_{(ab)^*} \rightarrow \epsilon, S_{(ab)^*} \rightarrow aS_b, S_b \rightarrow bS_{(ab)^*}\}, S_{(ab)^*})$
   $G_R = (\{S_R, S_{a^*}, S_{(ab)^*}, S_b\}, \{a, b\},$
   $\{S_R \rightarrow aS_{a^*}, S_R \rightarrow \epsilon, S_R \rightarrow aS_b, S_{a^*} \rightarrow \epsilon, S_{a^*} \rightarrow aS_{a^*}, S_b \rightarrow bS_{(ab)^*}, S_{(ab)^*} \rightarrow \epsilon, S_{(ab)^*} \rightarrow aS_b\},$
   $S_R)$

7. Pro následující jazyky

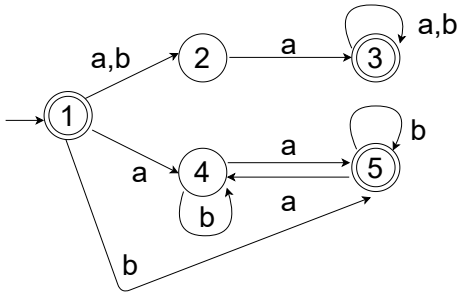
   - Sestrojte NKA pro jazyk (pokuste se efektivně využít nedeterminismus).

- Převeďte algoritmicky tento NKA na ekvivalentní DKA.
- Získaný DKA převeďte algoritmicky do redukované podoby, nebo ukažte, že již v redukované podobě je.

(a) $L = \{a,b\}\{a\}\{a,b\}^* \cup \{w \in \{a,b\}^* \mid \#_a(w) \texttt{ mod } 2 = 0\}$

(b) $L = \{w \in \{a,b\}^* \mid w$ obsahuje podslovo $bba\} \cup \{w \in \{a,b\}^* \mid |w| \texttt{ mod } 2 = 0\}$

Řešení příkladu a):
NKA pro jazyk L.



Dále tento automat determinizujeme.

		a	b
↔	$\{1\}$	$\{2,4\}$	$\{2,5\}$
	$\{2,4\}$	$\{3,5\}$	$\{4\}$
←	$\{2,5\}$	$\{3,4\}$	$\{5\}$
	$\{4\}$	$\{5\}$	$\{4\}$
←	$\{5\}$	$\{4\}$	$\{5\}$
←	$\{3,5\}$	$\{3,4\}$	$\{3,5\}$
←	$\{3,4\}$	$\{3,5\}$	$\{3,4\}$

Na závěr automat minimalizujeme.

$\equiv_0$

		a	b
↔ I	$\{1\}$	II	I
	$\{2,5\}$	I	I
	$\{5\}$	II	I
	$\{3,5\}$	I	I
	$\{3,4\}$	I	I
II	$\{2,4\}$	I	II
	$\{4\}$	I	II

$\equiv_1$

		a	b
↔ I	$\{1\}$	III	II
	$\{5\}$	III	I
← II	$\{2,5\}$	II	I
	$\{3,5\}$	II	II
	$\{3,4\}$	II	II
III	$\{2,4\}$	II	III
	$\{4\}$	I	III

$\equiv_2 = \equiv_3$

		a	b
↔ I	$\{1\}$	V	III
← II	$\{5\}$	VI	II
← III	$\{2,5\}$	IV	II
← IV	$\{3,5\}$	IV	IV
	$\{3,4\}$	IV	IV
V	$\{2,4\}$	IV	VI
VI	$\{4\}$	II	VI

8. Navrhněte a formálně popište algoritmus pro operaci průnik nad dvěma

   (a) deterministickými KA

   (b) nedeterministickými KA (nepoužívejte determinizaci).

   Řešení příkladu a):

   Vstup: Konečné deterministické automaty $A_1 = (Q_1, \Sigma_1, \delta_1, q_{0_1}, F_1)$ a $A_2 = (Q_2, \Sigma_2, \delta_2, q_{0_2}, F_2)$
   Výstup: Deterministický automat $A = (Q, \Sigma, \delta, q_0, F)$, pro který platí $L(A) = L(A_1) \cap L(A_2)$
   Metoda:
   1. $Q = Q_1 \times Q_2$ (Předpokládejme, že $Q_1 \cap Q_2 = \emptyset$. V opačném případě bychom přejmenovali stavy.)
   2. $\Sigma = \Sigma_1 \cap \Sigma_2$ (Pokud $\Sigma_1 \cap \Sigma_2 = \emptyset$, algoritmus vrací libovolný automat A, kde $L(A) = \emptyset$.)
   3. $\forall p, p' \in Q_1, q, q' \in Q_2, a \in \Sigma : \delta((p,q),a) = (p',q') \iff \delta_1(p,a) = p' \wedge \delta_2(q,a) = q'$
   4. $q_0 = (q_{0_1}, q_{0_2})$
   5. $F = F_1 \times F_2$

4 Rovnice nad regulárními výrazy

1. Řešením rovnic nad regulárními výrazy sestavte ekvivalentní regulární výraz k těmto automatům

   (a) M_1 z obrázku 1:

   (b) M_2 z obrázku 2:

   Řešení příkladu a):

   $X_0 = aX_1$
   $X_1 = cX_1 + cX_0 + bX_2$
   $X_2 = aX_1 + bX_0 + \epsilon$

Postupným dosazováním dostáváme

$X_1 = cX_1 + caX_1 + baX_1 + bbX_0 + b = (c + ca + ba + bba)X1 + b = (c + ca + ba + bba)^*b$
$X_0 = a(c + ca + ba + bba)^*b$

$L(M_1) = a(c + ca + ba + bba)^*b.$

5 Pumping lemma

1. Rozhodněte a dokažte, zda jsou následující jazyky regulární. Při dokazování, že je jazyk regulární, stačí uvést odpovídající gramatiku či automat. Při dokazování, že jazyk není regulární, použijte Pumping Lemma.

   (a) $L_1 = \{c^i w \mid i \geq 2 \wedge w \in \{a,b\}^* \wedge (\#_a(w) < \#_b(w) \vee |w| \leq 3)\}$,
   (b) $L_2 = \{w \in \{a,b\}^* \mid \#_a(w) = \#_b(w) \vee |w| \geq 3\}$,
   (c) $L_3 = \{w \mid w \in \{a,b,c\}^* \ \wedge \#_a(w) = \#_b(w) \ \wedge \ \#_c(w) = 1\}$
   (d) $L_4 = \{a^i b^{2i} c^j \mid i > 0 \wedge i < j < 2i\}$.

   Řešení příkladu a): Jazyk L_1 není regulární. Důkaz sporem. Předpokládejme, že $L_1 \in \mathcal{L}_3$. Pak dle Pumping lemmatu platí, že

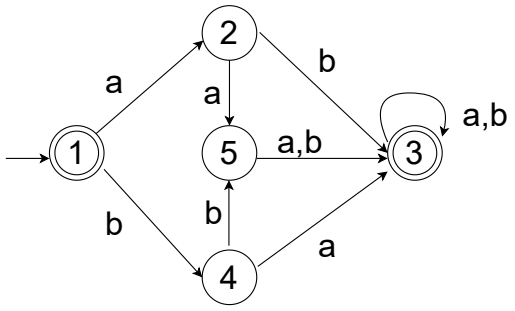
   $$\exists k > 0 : \forall w \in \Sigma^* : w \in L \ \wedge \ |w| \geq k \Rightarrow (\exists x, y, z \in \Sigma^* : w = xyz \wedge \ y \neq \epsilon \ \wedge \ |xy| \leq k \ \wedge \ \forall i \geq 0 : xy^i z \in L)$$

   Uvažme libovolné $k > 0$. Zvolme slovo $w = c^2 a^{k+1} b^{k+2}$. Platí, že $w \in L_1 \wedge |w| = 2k + 5 \geq k$. Pro každé rozdělení slova w takové, že $w = xyz \wedge \ y \neq \epsilon \ \wedge \ |xy| \leq k$, nastane alespoň jedna z následujících možností:

   - Podslovo y obsahuje znak c. Pak pro volbu $i = 0$ porušíme podmínku na počet znaků c a tudíž slovo $xy^i z \notin L_1$.
   - Podslovo y obsahuje znak a. Pak y nemůže obsahovat znak b kvůli podmínce $|xy| \leq k$, ale pro volbu $i = 2$ porušíme podmínku $\#_a(xy^i z) < \#_b(xy^i z)$ a tudíž slovo $xy^i z \notin L_1$.

   Ukázali jsme tedy, že pro libovolné validní rozdělení jsme schopni výsledné slovo "vypumpovat"ven z jazyka, což je spor s tvrzením Pumping lemmatu. Tudíž jsme dokázali, že L_1 není regulární jazyk.

   Řešení příkladu b): Jazyk L_2 je regulární. KA na obrázku.



2. Dokažte, že následující jazyky splňují tuto formuli (tj. pravou stranu Pumping lemmatu)

   $$\exists k > 0 : \forall w \in \Sigma^* : w \in L \ \wedge \ |w| \geq k \Rightarrow (\exists x, y, z \in \Sigma^* : w = xyz \wedge \ y \neq \epsilon \ \wedge \ |xy| \leq k \ \wedge \ \forall i \geq 0 : xy^i z \in L)$$

   (a) $L = \{a^i b^j \mid i, j > 0\}$
   (b) $L = \{w \in \{a,b\}^* \mid |w| > 3\}$
   (c) $L = \{w \in \{a,b\}^* \mid |w| < 3\}$

   Řešení příkladu a): Zvolme $k = 3$. Pro libovolné slovo $w \in L$ takové, že $|w| \geq 3$, platí, že $\#_a(w) \geq 2$, nebo $w = ab^j$, kde $j \geq 2$. Pokud $\#_a(w) \geq 2$, tak zvolíme rozdělení, kde $y = a$ ($y \neq \epsilon \wedge |xy| = |a| = 1 \leq 3$). Potom $xy^i z \in L$ pro libovolné $i \geq 0$, jelikož $\#_a(xy^i z) > 0$. Pokud $w = ab^j$, kde $j \geq 2$, tak zvolíme rozdělení, kde $y = b$ (pak $x = a \wedge y = b$ tedy $y \neq \epsilon \wedge |xy| = |ab| = 2 \leq 3$). Potom $xy^i z \in L$ pro libovolné $i \geq 0$, jelikož $\#_b(xy^i z) > 0$. Poznamenejme, že v obou případech neměníme pořadí znaků a a b.

6 Myhill-Nerodova věta

1. S využitím Myhill-Nerodovy věty dokažte, že následující jazyky nejsou regulární:

   (a) $L_1 = \{a^i b^j \mid i \neq j\}$

   (b) $L_2 = \{a^{2^n} \mid n > 0\}$

   (c) $L_3 = \{w \in \{a,b\}^* \mid w = w^R\}$

   Řešení příkladu a): Předpokládejme, že L_1 je regulární jazyk. Pak dle Myhill-Nerodovy věty platí, že $\exists n \in \mathbb{N} : |\Sigma^*/_{\sim_L}| = n$ (index relace prefixové ekvivalence je konečný). Nyní uvažme $n+1$ slov

   $$a^0, a^1, \ldots, a^n.$$

   Ukážeme, že žádná dvě různá slova a^i a a^j pro $0 \leq i, j \leq n$ nemohou být v relaci $\sim_L$ a tudíž $|\Sigma^*/_{\sim_L}| > n$, což je spor s naším předpokladem o $|\Sigma^*/_{\sim_L}|$. Kdyby $a^i \sim_L a^j$ pak musí platit, že $\forall w \in \Sigma^* : a^i w \in L \iff a^j w \in L$. Pokud zvolíme $w = b^i$, tak dostáváme

   $a^i b^i \notin L$
   $a^j b^i \in L$

   Tudíž $a^i \not\sim_L a^j$. Z toho plyne, že $|\Sigma^*/_{\sim_L}| > n$, což je spor.

2. S využitím Myhill-Nerodovy věty dokažte, že následující jazyky jsou regulární:

   (a) $L = \{w \in \{a,b\}^* \mid \#_a(w) > 3000\}$

   (b) $L = \{w \in \{a,b\}^* \mid \#_a(w) < 3000\}$

   (c) $L = \{w \in \{a,b\}^* \mid \#_a(w) \bmod 3 = \#_b(w) \bmod 2\}$

   Řešení příkladu a): Sestrojíme relaci pravé kongruence $\sim$ s konečným indexem a ukážeme, že L je sjednocení některých tříd rozkladu $\Sigma^*/_\sim$.

   $$u \sim v \iff \#_a(u) = \#_a(v) \vee (\#_a(u) > 3000 \wedge \#_a(v) > 3000)$$

   $\sim$ je zřejmě reflexivní i symetrická. Pokud $u \sim v \wedge v \sim w$, pak buď $\#_a(u) = \#_a(w)$ nebo $\#_a(u) > 3000 \wedge \#_a(w) > 3000$, a tudíž $u \sim w$, tj. $\sim$ je tranzitivní. $\sim$ je i pravá kongruence: Pokud $u \sim v$, pak i $ua \sim va$, jelikož $\#_a(ua) = \#_a(va) = \#_a(u) + 1$ nebo $\#_a(ua) > 3000 \wedge \#_a(va) > 3000$. Rovněž $ub \sim vb$, jelikož $\#_a(ub) = \#_a(u) \wedge \#_a(vb) = \#_a(v)$.

   $$\Sigma^*/_\sim = \{[x_i] \mid 0 \leq i \leq 3000\} \cup \{\{w \in \{a,b\}^* \mid \#_a(w) > 3000\}\} \text{ kde}$$
   $$[x_i] = \{w \in \{a,b\}^* \mid \#_a(w) = i\}$$

   a tedy $|\Sigma^*/_\sim| = 3002$. Jazyk L odpovídá jedné třídě $\Sigma^*/_\sim$, a to $\{w \in \{a,b\}^* \mid \#_a(w) > 3000\}$.

3. Dokažte, že neexistuje úplný deterministický automat se 3 stavy, který akceptuje jazyk:

   (a) $L = \{w \in \{a,b\}^* \mid \#_a(w) \geq 3\}$

   (b) $L = \{w \in \{a,b\}^* \mid \#_a(w) \bmod 4 = 0\}$

   Řešení příkladu a): Předpokládejme, že existuje totální deterministický automat se 3 stavy. Pak dle Myhill-Nerodovy věty $|\Sigma^*/_{\sim_L}| = 3$. Uvažme 4 slova slova a^i pro $0 \leq i \leq 3$. Pro libovolnou dvojici různých slov a^i a a^j kde $0 \leq i, j \leq 3 \wedge i < j$ (pokud $j < i$ slova prohodíme) ukážeme, že $a^i \not\sim_L a^j$ a tudíž $|\Sigma^*/_{\sim_L}| > 3$, což je spor s našim předpokladem. Kdyby $a^i \sim_L a^j$ pak musí platit, že $\forall w \in \Sigma^* : a^i w \in L \iff a^j w \in L$. Pokud zvolíme $w = a^{3-j}$, tak dostáváme

   $a^i a^{3-j} = a^k$ pro $k < 3$ (připomeňme, že $i < j$) a tudíž $a^i a^{3-j} \notin L$
   $a^j a^{3-j} = a^3 \in L$

   Tudíž $a^i \not\sim_L a^j$.

4. Formálně popište relace $\sim_1, \sim_2 \subseteq \{a,b\}^* \times \{a,b\}^*$ ($\sim_1 \neq \sim_2$), které splňují Myhill-Nerodovu větu pro jazyk $L = \{w \in \{a,b\}^* \mid w \text{ obsahuje podslovo } bb\}$, tj. relace je pravá kongruence s konečným indexem a L je sjednocením některých tříd rozkladu daným touto relací.

   Řešení: Hledané relace mohou vypadat následovně:

$$u \sim_1 v \iff (u,v \in (a+ba)^*) \vee (u,v \in (a+ba)^*b) \vee (u,v \in (a+ba)^*bb(a+b)^*)$$

$$u \sim_2 v \iff (u = v = \epsilon) \vee (u,v \in (a+ba)^+) \vee (u,v \in (a+ba)^*b) \vee (u,v \in (a+ba)^*bb(a+b)^*)$$

   Uvědomme si, že $\sim_1 = \sim_L$.

7 Uzávěrové vlastnosti regulárních jazyků

1. Rozhodněte a dokažte, zda platí následující tvrzení:

   (a) $(L_1 \cap L_2 \in \mathcal{L}_3) \wedge (L_1 \cap L_2 \text{ je nekonečný}) \Rightarrow L_1 \in \mathcal{L}_3 \vee L_2 \in \mathcal{L}_3$

   (b) $L_1 \cup L_2 \in \mathcal{L}_3 \Rightarrow L_1 \in \mathcal{L}_3 \vee L_2 \in \mathcal{L}_3$

   (c) $L \in \mathcal{L}_3 \Rightarrow \Diamond L \in \mathcal{L}_3$ kde $\Diamond L = \{w \in L \mid w \in \{a,b\}^* \wedge |w| > 100\}$

   (d) $L \in \mathcal{L}_3 \Rightarrow \Diamond L \in \mathcal{L}_3$ kde $\Diamond L = \{w \in L \mid w \in \{a,b\}^* \wedge \#_a(w) = \#_b(w) = 5\}$

   (e) $L \in \mathcal{L}_3 \Rightarrow \Diamond L \in \mathcal{L}_3$ kde $\Diamond L = \{w \in L \mid w \in \{a,b\}^* \wedge \#_a(w) = \#_b(w)\}$

   Řešení příkladu a): Tvrzení neplatí. Stačí uvážit

$$L_1 = \{a^n \mid n > 0\} \cup \{b^n c^n \mid n > 0\}$$

$$L_2 = \{a^n \mid n > 0\} \cup \{c^n b^n \mid n > 0\}.$$

   Pak $L_1 \cap L_2 = \{a^n \mid n > 0\}$ je nekonečný regulární jazyk, ale $L_1 \notin \mathcal{L}_3 \wedge L_2 \notin \mathcal{L}_3$

   Řešení příkladu c): Tvrzení platí. $\Diamond L = L \cap L_1$ kde $L_1 = \{w \in \{a,b\}^* \mid |w| > 100\}$ Dá se jednoduše ukázat, že $L_1 \in \mathcal{L}_3$. Pak z uzavřenosti $\mathcal{L}_3$ na operaci $\cap$ plyne, že i $\Diamond L \in \mathcal{L}_3$.

2. Rozhodněte a dokažte, zda pro libovolný jazyk L a libovolný konečný jazyk K platí následující tvrzení:

   (a) $L \in \mathcal{L}_3 \iff L \setminus K \in \mathcal{L}_3$

   (b) $L \in \mathcal{L}_3 \iff L \cup K \in \mathcal{L}_3$

   Řešení příkladu a): $\Rightarrow$: Uvědomme si, že $L \setminus K = L \cap co - K$. K je konečný $\Rightarrow K \in \mathcal{L}_3 \Rightarrow co - K \in \mathcal{L}_3$. Z uzávěrových vlastností regulárních jazyků dostáváme, že $L \cap co - K \in \mathcal{L}_3$. $\Leftarrow$: Uvědomme si, že $L = (L \setminus K) \cup (K \cap L)$. $K \cap L$ je konečný tj. regulární a z uzávěrových vlastností dostáváme, že i $(L \setminus K) \cup (K \cap L) \in \mathcal{L}_3$.

8 Rozhodnutelné problémy pro regulární jazyky

1. Dokažte, že následující problémy jsou rozhodnutelné:

   (a) Pro daný konečný automat $A = (Q, \Sigma, \delta, q_0, F)$ je rozhodnutelné, zda jazyk $L(A)$ je nekonečný.

   (b) Pro daný konečný automat $A = (Q, \Sigma, \delta, q_0, F)$ je rozhodnutelné, zda platí toto tvrzení:

$$\forall n \in \mathbb{N} : \exists w \in L(A) : \#_a(w) > n$$

   Řešení příkladu a): Bez újmy na obecnosti předpokládejme, že A je deterministický. Zavedeme relaci $\rightsquigarrow \subseteq Q \times Q$:

$$p \rightsquigarrow q \iff \exists a \in \Sigma : \delta(p,a) = q$$

$$L(A) \text{ je nekonečný} \iff \exists q \in Q, q_f \in F : q_0 \rightsquigarrow^* q \ \wedge \ q \rightsquigarrow^+ q \ \wedge \ q \rightsquigarrow^* q_f$$

   Problém nekonečnosti L(A) jsme tedy převedli na hledání cyklu ($q \rightsquigarrow^+ q$), který je dosažitelný z iniciálního stavu ($q_0 \rightsquigarrow^* q$) a je z něj dosažitelný koncový stav ($q \rightsquigarrow^* q_f$).

9 Základní konstrukce pro bezkontextové jazyky

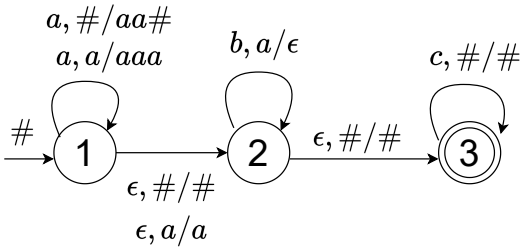
1. Zkonstruujte zásobníkové automaty a bezkontextové gramatiky pro tyto jazyky:

   (a) $L = \{a^n b^{2n} c^m \mid m, n \geq 0\}$
   (b) $L = \{ww'w^R \mid w \in \{a,b\}^* \wedge w' \in \{c,d\}^*\}$
   (c) $L = \{a^i b^j c^k \mid i \geq 3(k+j) \ \wedge \ i,j,k \geq 0\}$
   (d) $L = \{a^i b^j c^k \mid 3i \geq (k+j) \ \wedge \ i,j,k \geq 0\}$
   (e) $L = \{a^i b^j c^k \mid 3i \leq (k+j) \ \wedge \ i,j,k \geq 0\}$
   (f) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) = \#_b(w)\}$
   (g) $L = \{w \in \{a,b\}^* \mid \#_a(w) > \#_b(w) + 3\}$
   (h) $L = \{w \in \{a,b\}^* \mid \#_a(w) > 2 \cdot \#_b(w)\}$
   (i) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) > 2 \cdot (\#_b(w) + \#_c(w))\}$

   Řešení příkladu a):

   Automat na obrázku.

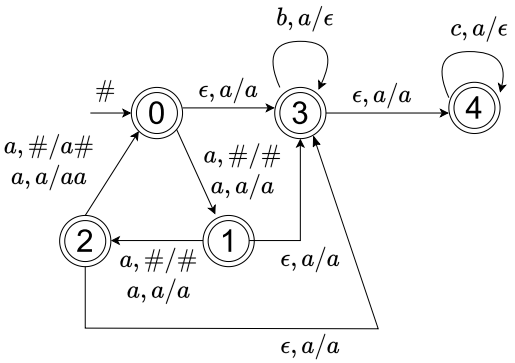
   $G = (\{S, S_1, C\}, \{a,b,c\}, P, S)$,
   kde P obsahuje pravidla:
   $S \ \rightarrow \ S_1 \mid C \mid S_1 C$
   $S_1 \ \rightarrow \ aS_1bb \mid \epsilon$
   $C \ \rightarrow \ cC \mid \epsilon$



   Řešení příkladu c):

   Automat na obrázku.

   $G = (\{S, S', A'\}, \{a,b,c\}, P, S)$,
   kde P obsahuje pravidla:
   $S \ \rightarrow \ aaaSc \mid aaaS'b \mid A$
   $S' \ \rightarrow \ aaaS'b \mid A$
   $A \ \rightarrow \ aA \mid \epsilon$



2. Pro následující gramatiky navrhněte (rozšířený) zásobníkový automat, který provádí syntaktickou analýzu

   - shora dolů,
   - zdola nahoru.

   V obou případech proveďte analýzu zadaného slova w.

   (a) $G = (\{S, A, B\}, \{a,b\}, P, S)$, kde P obsahuje pravidla:
   $S \ \rightarrow \ \epsilon \mid abSA$
   $A \ \rightarrow \ AaB \mid aB \mid a$
   $B \ \rightarrow \ aSS \mid bA$
   Slovo $w = ababa$.

(b) $G = (\{S, A, B\}, \{a, b, c\}, P, S)$, kde P obsahuje pravidla:
$S \rightarrow \epsilon \mid ASBB$
$A \rightarrow a \mid Aa$
$B \rightarrow b \mid c$
Slovo $w = aabc$.

Řešení příkladu a): Shora dolů. ZA $A = (\{q\}, \{a, b\}, \{S, A, B, a, b\}, \delta, q, S, \emptyset)$, kde
$\delta(q, \epsilon, S) = \{(q, \epsilon), (q, abSA)\}$
$\delta(q, \epsilon, A) = \{(q, AaB), (q, aB), (q, a)\}$
$\delta(q, \epsilon, B) = \{(q, aSS), (q, bA)\}$
$\delta(q, a, a) = \{(q, \epsilon)\}$
$\delta(q, b, b) = \{(q, \epsilon)\}$

$$(q, ababa, S) \vdash (q, ababa, abSA) \vdash^2 (q, aba, SA) \vdash (q, aba, A) \vdash (q, aba, aB) \vdash (q, ba, B) \vdash (q, ba, bA) \vdash$$

$$\vdash (q, a, A) \vdash (q, a, a) \vdash (q, \epsilon, \epsilon)$$

Zdola nahoru. ZA $A = (\{q, r\}, \{a, b\}, \{S, A, B, a, b, \#\}, \delta, q, \#, \{r\})$, kde
$\delta(q, \epsilon, \epsilon) = \{(q, S)\}$
$\delta(q, \epsilon, abSA) = \{(q, S)\}$
$\delta(q, \epsilon, AaB) = \{(q, A)\}$
$\delta(q, \epsilon, aB) = \{(q, A)\}$
$\delta(q, \epsilon, a) = \{(q, A)\}$
$\delta(q, \epsilon, aSS) = \{(q, B)\}$
$\delta(q, \epsilon, bA) = \{(q, B)\}$
$\delta(q, a, \epsilon) = \{(q, a)\}$
$\delta(q, b, \epsilon) = \{(q, b)\}$
$\delta(q, \epsilon, \#S) = \{(r, \epsilon)\}$
Vrchol zásobníku uvádíme, pro lepší čitelnost, vpravo.

$$(q, ababa, \#) \vdash^2 (q, aba, \#ab) \vdash (q, aba, \#abS) \vdash^3 (q, \epsilon, \#abSaba) \vdash (q, \epsilon, \#abSabA) \vdash (q, \epsilon, \#abSaB) \vdash$$

$$\vdash (q, \epsilon, \#abSA) \vdash (q, \epsilon, \#S) \vdash (r, \epsilon, \epsilon)$$

10 Transformace bezkontextových gramatik

1. Odstraňte ϵ-pravidla v následujících gramatikách:

(a) $G = (\{S, A, B, C, D\}, \{a, b, c\}, P, S)$, kde P obsahuje pravidla:
$S \rightarrow ABC$
$A \rightarrow AbA \mid BC \mid b$
$B \rightarrow bB \mid a \mid cBbAb \mid \epsilon$
$C \rightarrow cD \mid Aa \mid \epsilon$
$D \rightarrow SSD \mid b$

(b) $G = (\{S, A, B, C, D\}, \{a, b, c\}, P, S)$, kde P obsahuje pravidla:
$S \rightarrow ABC$
$A \rightarrow Ab \mid BC \mid b$
$B \rightarrow bB \mid a \mid Ab \mid \epsilon$
$C \rightarrow cD \mid cS \mid Aa \mid \epsilon$
$D \rightarrow SSS \mid bc$

Řešení příkladu a): $N_\epsilon^1 = \{B, C\}, N_\epsilon^2 = \{A, B, C\}, N_\epsilon^3 = N_\epsilon = \{S, A, B, C\}$

$G' = (\{S', S, A, B, C, D\}, \{a, b, c\}, P', S')$, kde P' obsahuje pravidla:

$S' \rightarrow S \mid \epsilon$

$S \rightarrow ABC \mid AB \mid A \mid AC \mid BC \mid B \mid C$

$A \rightarrow AbA \mid BC \mid b \mid Ab \mid bA \mid B \mid C$

$B \rightarrow bB \mid a \mid cBbAb \mid b \mid cbAb \mid cBbb \mid cbb$

$C \rightarrow cD \mid Aa \mid a$

$D \rightarrow SSD \mid b \mid SD \mid D$

2. Odstraňte jednoduchá pravidla v následujících gramatikách a pak je převeďte do Chomského normální formy.

   (a) $G = (\{S, A, B, C, D, E, F\}, \{a, b\}, P, S)$, kde P obsahuje pravidla:

   $S \rightarrow E \mid F$

   $A \rightarrow bS \mid D$

   $B \rightarrow Sa \mid a$

   $C \rightarrow aD \mid S$

   $D \rightarrow ba$

   $E \rightarrow aAS \mid C$

   $F \rightarrow SBb$

   (b) $G = (\{S, A, B, C, D, E, F\}, \{a, b\}, P, S)$, kde P obsahuje pravidla:

   $S \rightarrow A \mid F$

   $A \rightarrow bS$

   $B \rightarrow Sa \mid a \mid S$

   $C \rightarrow aD \mid S$

   $D \rightarrow Ba$

   $E \rightarrow aAS \mid E$

   $F \rightarrow SAb$

   Řešení příkladu a): $N_S = \{S, E, F, C\}, N_A = \{A, D\}, N_B = \{B\}, N_C = \{C, S, E, F\}, N_D = \{D\}, N_E = \{E, C, S, F\}, N_F = \{F\}$ $G' = (\{S, A, B, C, D, E, F\}, \{a, b\}, P', S)$, kde P' obsahuje pravidla:

   $S \rightarrow aAS \mid SBb \mid aD$

   $A \rightarrow bS \mid ba$

   $B \rightarrow Sa \mid a$

   $C \rightarrow aD \mid aAS \mid SBb$

   $D \rightarrow ba$

   $E \rightarrow aAS \mid SBb \mid aD$

   $F \rightarrow SBb$

   Nyní převedeme G' do Chomského normální formy.

   $G'' = (\{S, A, B, C, D, E, F, <AS>, <Bb>, <a>, <b>\}, \{a, b\}, P'', S)$, kde P'' obsahuje pravidla:

$$
\begin{aligned}
S &\rightarrow <a><AS> \mid S<Bb> \mid <a>D \\
A &\rightarrow <b>S \mid <b><a> \\
B &\rightarrow S<a> \mid a \\
C &\rightarrow <a>D \mid <a><AS> \mid S<Bb> \\
D &\rightarrow <b><a> \\
E &\rightarrow <a><AS> \mid S<Bb> \mid <a>D \\
F &\rightarrow S<Bb> \\
<AS> &\rightarrow AS \\
<Bb> &\rightarrow B<b> \\
<a> &\rightarrow a \\
<b> &\rightarrow b
\end{aligned}
$$

3. Odstraňte levou rekurzi v následujících gramatikách.

   (a) $G = (\{S, A\}, \{a, b\}, P, S)$, kde P obsahuje pravidla:
   $S \rightarrow AAa \mid Aa \mid AS \mid b$
   $A \rightarrow SbA \mid AbA \mid b$

   (b) $G = (\{S, A, B\}, \{a, b\}, P, S)$, kde P obsahuje pravidla:
   $S \rightarrow Ab \mid a \mid bB$
   $A \rightarrow BSa \mid ab \mid SBa$
   $B \rightarrow bB \mid BaB \mid Sa$

   Řešení příkladu a): Nejprve rozgenerujeme pravidlo $A \rightarrow SbA$, abychom získali přímou levou rekurzi.
   $A \rightarrow AAabA \mid AabA \mid ASbA \mid bbA \mid AbA \mid b$
   Poté odstraníme přímou levou rekurzi u neterminálu A.
   $A \rightarrow bbA \mid b \mid bbAA' \mid bA'$
   $A' \rightarrow AabA \mid abA \mid SbA \mid bA \mid AabAA' \mid abAA' \mid SbAA' \mid bAA'$
   $G = (\{S, A, A'\}, \{a, b\}, P', S)$, kde P' obsahuje pravidla:
   $S \rightarrow AAa \mid Aa \mid AS \mid b$
   $A \rightarrow bbA \mid b \mid bbAA' \mid bA'$
   $A' \rightarrow AabA \mid abA \mid SbA \mid bA \mid AabAA' \mid abAA' \mid SbAA' \mid bAA'$

11 Jednoznačnost a determinismus bezkontextových gramatik

1. Navrhněte jednoznačnou gramatiku pro jazyk

   (a) $L = \{ww^R \mid w \in \{a, b\}^*\} \cup \{a^k \mid k > 2\} \cup \{b^k \mid k > 3\}$

   (b) $L = \{a^n b^n \mid n > 0\} \cup \{a^n b^{2m} \mid m, n > 0\}$

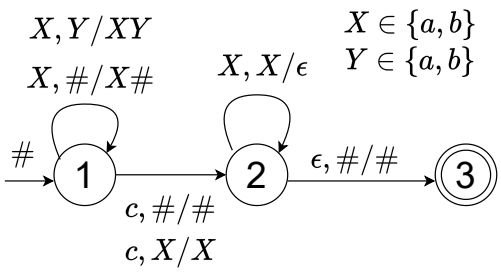
   Řešení příkladu a): $G = (\{S, S_1, S_2, S_3\}, \{a, b\}, P, S)$ kde P obsahuje pravidla
   $S \rightarrow S_1 \mid S_2 \mid S_3$
   $S_1 \rightarrow aS_1a \mid bS_1b \mid \epsilon$
   $S_2 \rightarrow aS_2a \mid aaa$
   $S_3 \rightarrow bS_3b \mid bbbbb$

2. Navrhněte deterministický zásobníkový automat pro jazyk:

   (a) $L = \{wcw^R \mid w \in \{a, b\}^*\}$

   (b) $L = 0L_1 \cup L_2$, kde $L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$ a $L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$

Řešení příkladu a): DZA na obrázku



12 Fix-point algoritmy pro bezkontextové gramatiky

1. Zkonstruujte algoritmus, který pro danou BG $G = (N, \Sigma, S, P)$ spočítá následující množiny:

   (a) $N_F = \{A \in N \mid \exists k \in \mathbb{N} : \texttt{card}(\{w \in \Sigma^* \mid A \Rightarrow^+ w\}) \leq k\}$

   (b) $N_a = \{A \in N \mid \forall k \in \mathbb{N} : \exists w \in \Sigma^* : A \Rightarrow^+ w \wedge \#_a(w) > k\}$

   Řešení příkladu a): Množina N_F obsahuje neterminály, z kterých se dá derivovat jen konečný počet řetězců nad Σ^*. Zkonstruujeme algoritmus, který spočítá množinu $N_N = N \setminus N_F$, která obsahuje neterminály, z kterých se dá derivovat nekonečný počet řetězců. Požadovanou množinu pak dostaneme jako $N_F = N \setminus N_N$. Nejdříve si spočítáme množinu neterminálů

   $$N_t = \{A \in N \mid A \Rightarrow^+ w \wedge w \in \Sigma^*\},$$

   tj. neterminálů, které generují alespoň 1 řetězec terminálů (i prázdný). N_t získáme pomocí následující fix-point algoritmu.

   $N_t^0 = \emptyset$
   $N_t^{i+1} = \{A \in N \mid \exists (A \to \alpha) \in P \wedge \alpha \in (\Sigma \cup N_t^i)^*\}$
   jestliže $N_t^{i+1} = N_t^i$, tak $N_t = N_t^i$

   Dále si spočítáme množinu neterminálů

   $$N_{t+} = \{A \in N \mid A \Rightarrow^+ w \wedge w \in \Sigma^+\},$$

   tj. neterminálů, které generují alespoň 1 neprázdný řetězec terminálů. N_{t+} získáme pomocí následující fix-point algoritmu.

   $N_{t+}^0 = \emptyset$
   $N_{t+}^{i+1} = \{A \in N \mid \exists (A \to \alpha) \in P \wedge \alpha \in (\Sigma \cup N_t)^*(\Sigma \cup N_{t+}^i)(\Sigma \cup N_t)^*\}$
   jestliže $N_{t+}^{i+1} = N_{t+}^i$, tak $N_{t+} = N_{t+}^i$

   Dále definujeme relace $\rho, \rho_\Sigma \subseteq N \times N$, takové že

   $$A \rho B \iff \exists (A \to \alpha B \beta) \in P, \text{ kde } \alpha, \beta \in (\Sigma \cup N_t)^*,$$

   $A \rho_\Sigma B \iff \exists (A \to \alpha B \beta) \in P \vee (A \to \beta B \alpha) \in P$, kde $\alpha \in (\Sigma \cup N_t)^* \wedge \beta \in (\Sigma \cup N_t)^*(\Sigma \cup N_{t+})(\Sigma \cup N_t)^*$.

   Požadovanou množinu pak dostaneme jako:

   $$N_N = \{A \in N \mid \exists B, C, D \in N_t : A \rho^* B \wedge B \rho^* C \wedge C \rho_\Sigma D \wedge D \rho^* B\},$$

   tj. z A je dosažitelný cyklus $B \rightsquigarrow B$, který generuje aspoň jeden terminál (přechod $C \rho_\Sigma D$).

13 Pumping lemma pro bezkontextové jazyky

1. Pomocí Pumping lemmatu dokažte, že následující jazyky nejsou bezkontextové:

   (a) $L = \{a^i w \mid i \geq 2 \wedge w \in \{b,c,d\}^* \ \wedge \ \#_b(w) > \#_c(w) > \#_d(w)\}$,

   (b) $L = \{uww^R u \mid u, w \in \{a,b\}^*\}$,

   (c) $L = \{a^i b^{2i} c^{3i} \mid i \geq 0\}$.

   Řešení příkladu a): Nechť k je konstanta z Pumping lemmatu. Zvolme slovo $z = a^2 b^{k+2} c^{k+1} d^k$. Evidentně $z \in L$ a $|z| = 3k + 5 \geq k$. Pro každé rozdělení slova z takové, že $z = uvwxy \wedge \ vx \neq \epsilon \ \wedge \ |vwx| \leq k$, nastane alespoň jedna z následujících možností:

   - Podslovo vx obsahuje znak a. Pak pro volbu $i = 0$ porušíme u řetězce $uv^i wx^i y$ definiční podmínku jazyka L na počet znaků a a tudíž slovo $uv^i wx^i y \notin L$.
   - Podslovo vx obsahuje znak b, ale neobsahuje znak c. Pak pro volbu $i = 0$ porušíme podmínku $\#_b(uv^i wx^i y) > \#_c(uv^i wx^i y)$ a tudíž slovo $uv^i wx^i y \notin L$.
   - Podslovo vx obsahuje znak c, ale neobsahuje znak b. Pak pro volbu $i = 2$ porušíme podmínku $\#_b(uv^i wx^i y) > \#_c(uv^i wx^i y)$ a tudíž slovo $uv^i wx^i y \notin L$.
   - Podslovo vx obsahuje znak c, ale neobsahuje znak d. Pak pro volbu $i = 0$ porušíme podmínku $\#_c(uv^i wx^i y) > \#_d(uv^i wx^i y)$ a tudíž slovo $uv^i wx^i y \notin L$.
   - Podslovo vx obsahuje znak d, ale neobsahuje znak c. Pak pro volbu $i = 2$ porušíme podmínku $\#_c(uv^i wx^i y) > \#_d(uv^i wx^i y)$ a tudíž slovo $uv^i wx^i y \notin L$.

   Zdůrazněme, že díky podmínce $|vwx| \leq k$ podslovo vx nemůže zároveň obsahovat znaky b a d (tedy ani znaky b, c a d) a tudíž jsme skutečně ukázali, že pro libovolné validní rozdělení jsme schopni výsledné slovo "vypumpovat"ven z jazyka. Tudíž jsme dokázali, že L není bezkontextový jazyk.

2. Dokažte, že následující jazyky splňují tuto formuli (tj. pravou stranu Pumping lemmatu)

   $$\exists k > 0 : \forall z \in \Sigma^* : z \in L \ \wedge \ |z| \geq k \Rightarrow \\ (\exists uvwxy \in \Sigma^* : z = uvwxy \wedge \ vx \neq \epsilon \ \wedge \ |vwx| \leq k \ \wedge \ \forall i \geq 0 : uv^i wx^i y \in L)$$

   (a) $L = \{ww^R \mid w \in \{a,b\}^*\}$,

   (b) $L = \{w \in \{a,b\}^* | \ \#_a(w) = \#_b(w)\}$.

   Řešení příkladu a): Zvolme $k = 2$. Pro libovolné slovo $z = a_1 a_2 \ldots a_{n-1} a_n$ z jazyka L ($n \geq 2$ a je sudé) platí, že

   $$a_1 = a_n \wedge a_2 = a_{n-1} \wedge \ldots \wedge a_{n/2} = a_{n/2+1}.$$

   Potom zvolme rozdělení, kde $v = a_{n/2} \wedge w = \epsilon \wedge x = a_{n/2+1}$ (tudíž $|vwx| = 2$). Pak pro libovolné $i \geq 0$ dostáváme slovo $uv^i wx^i y = a_1 a_2 \ldots a_{m-1} a_m$, kde m je sudé a platí

   $$a_1 = a_m \wedge a_2 = a_{m-1} \wedge \ldots \wedge a_{m/2} = a_{m/2+1}.$$

   a tudíž $uv^i wx^i y \in L$. Poznamenejme, že pokud $n = 2$ a $i = 0$, tak $uv^i wx^i y = \epsilon \in L$.

14 Uzávěrové vlastnosti bezkontextových gramatik

1. Rozhodněte a dokažte, zda platí následující tvrzení:

   (a) $L_1 \in \mathcal{L}_3 \vee L_2 \in \mathcal{L}_3 \implies L_1 \cup L_2 \in \mathcal{L}_2$,

   (b) $L_1 \in \mathcal{L}_2 \setminus \mathcal{L}_3 \wedge L_2$ je konečný $\implies (L_1 \setminus L_2) \in \mathcal{L}_2 \setminus \mathcal{L}_3$,

   (c) $\exists L_1, L_2 \in \mathcal{L}_2 \setminus \mathcal{L}_3 : L_1 \cdot L_2 \in \mathcal{L}_3$,

   (d) $\exists L_1, L_2 \in \mathcal{L}_1 \setminus \mathcal{L}_2 : L_1 \cap L_2 \in \mathcal{L}_2 \setminus \mathcal{L}_3$,

   (e) $\exists L_1, L_2 \in \mathcal{L}_1 \setminus \mathcal{L}_2 : L_1 \cup L_2 \in \mathcal{L}_2 \setminus \mathcal{L}_3$.

Řešení příkladu a): Tvrzení neplatí. Stačí zvolit $L_1 = \emptyset$ a $L_2 \notin \mathcal{L}_2$ (například $L_2 = \{a^n b^n c^n \mid n \geq 0\}$). Pak $L_1 \cup L_2 = L_2 \notin \mathcal{L}_2$.

Řešení příkladu b): Tvrzení platí. Předpokládejme, že tvrzení neplatí, tj. že existuje $L_1 \in \mathcal{L}_2 \setminus \mathcal{L}_3 \wedge$ L_2 je konečný a zároveň $(L_1 \setminus L_2) \notin \mathcal{L}_2 \setminus \mathcal{L}_3$. Jelikož $L_1 \setminus L_2 = L_1 \cap co - L_2$ a $co - L_2$ je regulární (L_2 je konečný a tudíž regulární), pak i $L_1 \setminus L_2 \in \mathcal{L}_2$ ($\mathcal{L}_2$ je uzavřena na průnik s regulárními jazyky). Z $L_1 \setminus L_2 \in \mathcal{L}_2$ a v kombinaci s naším předpokladem, že $(L_1 \setminus L_2) \notin \mathcal{L}_2 \setminus \mathcal{L}_3$ dostáváme, že $L_1 \setminus L_2 \in \mathcal{L}_3$. To je ovšem spor, jelikož $L_1 = L_1 \setminus L_2 \cup (L_1 \cap L_2)$ a $L_1 \cap L_2 \in \mathcal{L}_3$ (L_2 je konečný). Tudíž dostáváme, že i $L_1 \in \mathcal{L}_3$, což je spor s tím, že $L_1 \in \mathcal{L}_2 \setminus \mathcal{L}_3$.

15 Základní konstrukce Turingových strojů

1. Zkonstruujte (a popište přechodovým diagramem) TS, který:

   (a) převádí vstupní konfiguraci pásky $\underline{\Delta} a^{2n} \Delta^\omega$, kde $n > 0$, na $\underline{\Delta} a^n \Delta^\omega$,

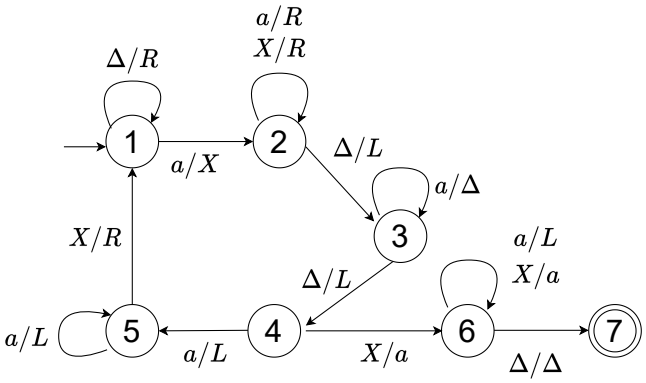
   (b) převádí vstupní konfiguraci pásky $\underline{\Delta} x \Delta^\omega$, kde $x \in \{0\} \cdot \{0,1\}^*$, na $\underline{\Delta} y \Delta^\omega$, kde x a y reprezentují čísla u_x a u_y v binárním kódu taková, že $u_y = 2 \cdot u_x$,

   (c) převádí vstupní konfiguraci pásky $\underline{\Delta} a^n \Delta^\omega$, kde $n > 0$, na $\underline{\Delta} a^n b^{2n} c^{3n} \Delta^\omega$,

   (d) akceptuje jazyk $L = \{w \in \{a,b,c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$,

   (e) akceptuje jazyk $L = \{w \in \{a,b,c\}^* \mid \#_a(w) > 2 * \#_b(w)\}$.

   Řešení příkladu a): TS na obrázku.



16 Konstrukce vícepáskových Turingových strojů

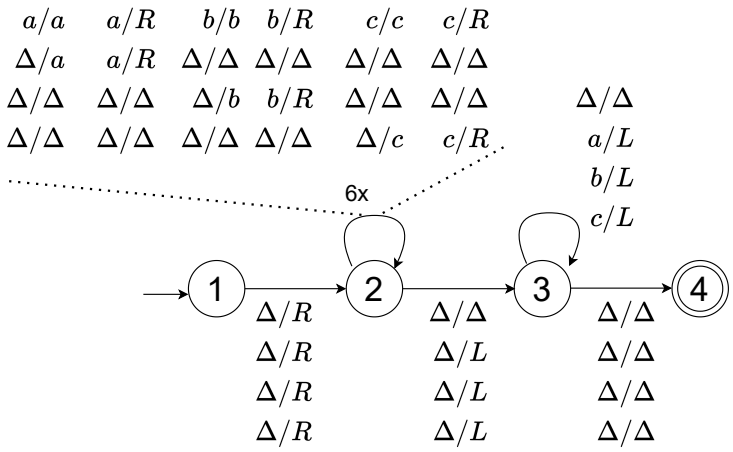
1. Zkonstruujte vícepáskový TS (a popište přechodovým diagramem), který akceptuje jazyk:

   (a) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$,

   (b) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\}$,

   (c) $L = \{w \in \{a,b,c\}^* \mid \#_a(w) = 2 * \#_b(w) = \#_c(w)\}$.

   Řešení příkladu a): 4-páskový TS na obrázku.



17 Důkazy (částečné) rozhodnutelnosti

1. Pro každý níže uvedený jazyk rozhodněte, zda je rozhodnutelný či pouze částečně rozhodnutelný. Uveďte hlavní myšlenku konstrukce TS, který tuto (částečnou) rozhodnutelnost dokazuje.

   (a) $L = \{<M> \# <n> \mid \texttt{card}(L(M)) > n\}$,

   (b) $L = \{<M_1> \# <M_2> \mid L(M_1) \cap L(M_2) \neq \emptyset\}$,

   (c) $L = \{<M> \# <w> \# <n> \mid M$ neakceptuje w do n kroků$\}$,

   (d) $L = \{<M> \# <n> \mid \exists w \in \Sigma^* : M$ akceptuje w do n kroků$\}$,

   (e) $L = \{<M> \# <n> \mid \forall w \in \Sigma^* : \texttt{steps}(M, w) \leq n\}$,

   (f) $L = \{<M> \# <n> \mid \forall w \in \Sigma^* : \texttt{steps}(M, w) \geq n\}$.

   Řešení příkladu a): Jazyk je částečně rozhodnutelný. Sestrojíme nedeterministický TS T, pro který $L(T) = L$. TS T nedeterministicky zvolí $n + 1$ různých slov. Pro tyto slova postupně pouští simulace stroje M. Pokud všechny simulace skončí v akceptujícím stavu tak T akceptuje vstup $< M > \# < n >$. Pokud nějaká simulace skončí v zamítajícím stavu (či skončí abnormálně), T zamítá tento vstup. Pokud nějaká simulace cyklí, tak T cyklí. Zřejmě platí, že pokud $\texttt{card}(L(M)) > n$, tak v T existuje akceptující výpočet (pro správnou volbu slov). V opačném případě T na všech cestách zamítá či cyklí. Tudíž skutečně platí $L(T) = L$. Alternativní řešení bez použití "neohraničeného"nedeterminismu využívá paralelní simulaci (tzv. step-counting): Sestrojíme deterministický TS T pro který $L(T) = L$. TS T postupně lexikograficky generuje slova $w_1, w_2, w_3 \ldots$ a pouští na nich paralelně simulaci. V každé iteraci $i > 1$ udělá jeden další krok simulace na všech slovech w_j, kde $0 < j \leq i$. Po iteraci i tedy T již provedl i kroků na slově w_1, $i-1$ kroků na slově w_2, $i-2$ kroků na slově w_3 atd. Pokud simulace nějakého slova skončila v akceptujícím stavu, T inkrementuje čítač akceptovaných slov. Pokud čítač dosáhne hodnotu $n+1$, T akceptuje. Zřejmě platí, že pokud $\texttt{card}(L(M)) > n$, tak existuje nějaké číslo $c \geq n+1$, že ze slov $w_1, w_2, \ldots w_c$ M akceptuje $n + 1$ slov do c kroků a tudíž T akceptuje $<M> \# <n>$. Pokud $\texttt{card}(L(M)) \leq n$, tak T bude cyklit.

18 Rozhodnutelnost a redukce, diagonalizace

1. Rozhodněte, zda jsou následující jazyky rozhodnutelné, nerozhodnutelné ale částečně rozhodnutelné nebo nejsou ani částečně rozhodnutelné. Nerozhodnutelnost případně skutečnost, že jazyky není ani částečně rozhodnutelný dokažte pomocí redukce. Rozhodnutelnost a částečnou rozhodnutelnost nemusíte dokazovat (podobné důkazy byly procvičeny v minulém příkladu).

   (a) $L = \{<M> \mid M$ je TS t.ž. $\texttt{card}(L(M) \cap \{a, b, c, d\}) > 2\}$

   (b) $L = \{<M> \mid M$ je TS t.ž. $\texttt{card}(L(M) \cap \{a, b, c, d\}) = 2\}$

(c) $L = \{<M> \# <n> \mid M \text{ je TS t.ž. } \texttt{card}(L(M)) > n\}$

(d) $L = \{<M> \# <n_1> \# <n_2> \mid M \text{ je TS t.ž. } n_1 \leq \texttt{card}(L(M)) \leq n_2\}$

(e) $L = \{<M> \# <w_1> \# <w_2> \mid M \text{ je TS t.ž. } w_1, w_2 \in L(M) \wedge \texttt{steps}(M, w_1) > \texttt{steps}(M, w_2)\}$

(f) $L = \{<M_1> \# <M_2> \mid M_1, M_2 \text{ jsou TS t.ž. } L(M_1) \cup L(M_2) \neq \emptyset\}$

(g) $^*L = \{<M> \mid M \text{ je TS t.ž. } L(M) = \Sigma^*\}$

Řešení a): Jazyk je částečně rozhodnutelný. Jeho nerozhodnutelnost dokážeme redukcí z HP, tj ukážeme, že $HP \leq L_1$. Požadovaná redukce je funkce $\sigma : \{0, 1, \#\}^* \to \{0, 1\}^*$ definovaná takto:

$$\sigma(<M> \# <w>) = <M'> .$$

Pokud vstup $<M> \# <w>$ není korektní instance HP, tak funkce σ vrací kód TS M' takového, že $L(M') = \emptyset$ (tj. $<M'> \notin L$). Jinak σ vrací kód TS M', který pracuje následovně.

- M' nejdříve spustí simulaci stroje M na vstupu w (tj. ignoruje svůj vstup w'). Poznamenejme, že kódy M a w jsou přímo uloženy v kódu M'.
- Pokud simulace cyklí, tak M' cyklí pro každý svůj vstup w'.
- Pokud simulace skončí, tak M' akceptuje a tudíž akceptuje každý svůj vstup w'.

Tudíž pro M' platí:

$$<M> \# <w> \notin HP \;\Rightarrow\; L(M') = \emptyset \;\Rightarrow <M'> \notin L \text{ a}$$

$$<M> \# <w> \in HP \;\Rightarrow\; L(M') = \Sigma^* \;\Rightarrow <M'> \in L,$$

$$\text{neboli } <M> \# <w> \in HP \iff \sigma(<M> \# <w>) \in L.$$

Je vidět, že výše popsanou konstrukci stroje M' lze implementovat pomocí úplného TS a tudíž funkce σ je totální, rekurzivně vyčíslitelná funkce.

Řešení b): Jazyk není ani částečně rozhodnutelný. Toto dokážeme redukcí z $co-HP$, tj ukážeme, že $co-HP \leq L$. Požadovaná redukce je funkce $\sigma\{0, 1, \#\}^* \to \{0, 1\}^*$ definovaná takto:

$$\sigma(<M> \# <w>) = <M'> .$$

Pokud vstup $<M> \# <w>$ není korektní instance $co-HP$, tak funkce σ vrací kód TS M' takového, že $L(M') = \emptyset$ (tj. $<M'> \notin L$). Jinak σ vrací kód TS M', který pracuje následovně.

- M' nejdříve zkontroluje svůj vstup w'.
- Pokud $w' \in \{a, b\}$ tak M' akceptuje.
- Pokud $w' \notin \{a, b\}$ tak M' spustí simulaci stroje M na vstupu w. Poznamenejme, že kódy M a w jsou přímo uloženy v kódu M'.
- Pokud simulace cyklí, tak M' cyklí pro každý vstup $w' \in \Sigma^* \setminus \{a, b\}$.
- Pokud simulace skončí, tak M' akceptuje a tudíž akceptuje každý svůj vstup w'.

Tudíž pro M' platí:

$$<M> \# <w> \in co-HP \;\Rightarrow\; L(M') = \{a, b\} \;\Rightarrow <M'> \in L \text{ a}$$

$$<M> \# <w> \notin co-HP \;\Rightarrow\; L(M') = \Sigma^* \;\Rightarrow <M'> \notin L,$$

$$\text{neboli } <M> \# <w> \in co-HP \iff \sigma(<M> \# <w>) \in L.$$

Je vidět, že výše popsanou konstrukci stroje M' lze implementovat pomocí úplného TS a tudíž funkce σ je totální, rekurzivně vyčíslitelná funkce.

2. Rozhodněte a dokažte, zda platí následující tvrzení

(a) $\forall L_1, L_2 \in \mathcal{L}_2 \setminus \mathcal{L}_3 : L_1 \leq L_2$,

(b) $\forall L_1, L_2 \in 2^{\Sigma^*} : L_1 \leq L_2$,

(c) $\forall L_1, L_2 \in \mathcal{L}_{REC} : L_1 \leq L_2$,

(d) $\forall L_1, L_2 \in \mathcal{L}_{REC} \setminus \{\emptyset, \Sigma^*\} : L_1 \leq L_2$.

Řešení a): Toto tvrzení platí. Dokážeme, že existuje požadovaná redukce δ. Jelikož $L_2 \in \mathcal{L}_2 \setminus \mathcal{L}_3$, pak existuje $w_1, w_2 \in \Sigma^*$, pro které platí $w_1 \in L_2 \wedge w_2 \notin L_2$. Jelikož $L_1 \in \mathcal{L}_2 \setminus \mathcal{L}_3$, tak existuje úplný TS M, pro který platí $L(M) = L_1$. Úplný TS T, který implementuje δ pracuje následovně. Pro vstup w TS T simuluje TS M. Pokud M akceptuje, tj. $w \in L_1$, pak T vrací $w_1 \in L_2$. Pokud M zamítá w, tj. $w \notin L_1$, pak T vrací $w_2 \notin L_2$. Je zřejmé, že $w \in L_1 \iff \delta(w) \in L_2$.

3. Diagonalizací ukažte, že:

   (a) existuje úplný TS, který se liší od všech lineárně ohraničených automatů,

   (b) existuje úplný TS, který se liší od všech kontextových gramatik.

   Poznámka: V důkazu nevyužívejte tvrzení o ekvivalentní vyjadřovací síle kontextových gramatik a lineárně ohraničených automatů.

   Řešení a): Uvažme nekonečnou matici:

$$\begin{array}{cccc} & w_0 & w_1 & \dots \\ LOA_0 & a_{00} & a_{01} & \dots \\ LOA_1 & a_{10} & a_{11} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

   V prvním řádku máme lexikograficky seřazená všechna slova nad abecedou Σ^* a v prvním sloupci máme všechny lineárně ohraničené automaty (LOA) nad abecedou Σ^* seřazené dle jejich kódu. LOA můžeme takto seřadit, jelikož existuje funkce, která pro číslo i vrací LOA_i, tj. LOA jehož kód je dán binární reprezentací čísla i. Pokud i není validní kód LOA, pak LOA_i je triviální stroj, který každý vstup přijme v jednom kroku. Dále platí, že

$$a_{ij} = \begin{cases} 1 & \text{if } w_j \in L(LOA_i) \\ 0 & \text{if } w_j \notin L(LOA_i). \end{cases}$$

   Nyní uvážíme jazyk $L = \{w_i \mid a_{ii} = 0\}$. L není možné přijmout žádným LOA. Pokud by existoval LOA, který akceptuje L, tak by se musel vyskytovat na nějakém řádku naší matice. L se však liší od všech LOA uvedených v matici právě v řetězcích na diagonále. Nyní ukážeme, že L lze akceptovat úplným TS. Uvažme úplný TS T, který pracuje následovně. Pro vstup w_i, T simuluje LOA_i. Pokud LOA_i akceptuje T zamítá, pokud LOA_i zamítá T akceptuje a pokud LOA_i cyklí, T je schopen toto detekovat (viz skripta) a akceptovat w_i. Tímto jsme ukázali, že T je skutečně úplný TS, který akceptuje L.

19 Logika a rozhodnutelnost, neúplnost

Verze 0 této kapitoly, o problémech, chybách, nejasnostech prosím dejte vědět.

1. Navrhněte efektivní teorii s jazykem, jehož signatura obsahuje pouze symbol rovnosti (t.j., $\langle \emptyset, \{=_{/2}\} \rangle$), jejíž modely mají pouze nekonečné domény.

   Rešení: Hint: Teorie musí být pouze efektivní, nic jiného. Nemusí být konečná.

2. Navrhněte konečnou teorii (množina T spec. axiomů je konečná) s jakýmkoliv jazykem, která má jen nekonečné modely.

3. Rozhodněte a zdůvodněte, zda jsou varianty teorie T (se signaturou $\langle \{0_{/0}, f_{/1}\}, \{=_{/2}\} \rangle$) s následujícími speciálními axiomy bezesporné, úplné. Co můžete navíc říci o jejich rozhodnutelnosti?

(a)
1. $\forall x \exists y (x = f(y))$
2. $\forall x (f(x) = 0)$

(b)
1. $\forall x \exists y (f(x) = f(f(y)))$
2. $\forall x (f(x) = x)$

(c)
1. $\exists x \forall y (f(x) = f(y))$
2. $\forall x (f(x) = x)$

(d)
1. $\exists x (f(x) \neq f(f(x)))$
2. $\forall x (f(x) = x)$

Rešení: (3a) Není sporná, má model M, kde $D_M = \{a\}$, $M(0) = \{a\}$ a $M(f) = \{(a, a)\}$. Je úplná, je to její jediný model (1. říká, že na každý prvek se něco musí zobrazí a 2. říká, že všechny prvky se zobrazí na 0). Rozhodnutelná bude: Protože má konečně mnoho modelů konečné velikosti, bude možné pro všechny modely zkusit všechny možnosti dosazení hodnot z domény za proměnné.

(3b) Není sporná, má model M, kde $D_M = \{a\}$, $M(0) = \{a\}$ a $M(f) = \{(a, a)\}$. Není úplná, má ještě model M', kde $D'_M = \{a, b\}$, $M(0) = \{a\}$ a $M'(f) = \{(a, a), (b, b)\}$, a jsou rozlišitelné formulí $\phi : \forall x \forall y\, x = y$, kde $M \models \phi$ a $M' \models \neg\phi$. Rozhodnutelná bude, protože teorie rovnosti je rozhodnutelná, ale nebude to úplně jednoduché...

(3d) Je sporná, nemá model. Rozhodnutelná je, důsledkem sporné teorie jsou všechny věty jazyka.

4. Navrhněte, jak eliminovat kvantifikátory pro teorii T se signaturou $\langle \emptyset, \{ \mathit{True}_{/1} \} \rangle$ a prázdnou množinou axiomů.

   Rešení: Uvědomme si, že jde v podstatě o výrokovou logiku s kvantifikátory. Jiné atomické formule než $\mathit{True}(x)$ v jazyce nejsou a pro každou proměnnou x, $\mathit{True}(x)$ je vyhodnoceno buď na 0 nebo na 1. Z formule $\exists x \varphi$ tedy eliminujeme kvantifikátor tak, že z φ vytvoříme formuli ψ, která bude disjunkcí těchto dvou případů, tedy $\psi : \varphi[1(x)/1] \vee \varphi[1(x)/0]$. (například z $\exists x(\mathit{True}(x) \vee \mathit{True}(y))$ vyrobíme $(1 \vee \mathit{True}(y)) \vee (0 \vee \mathit{True}(y))$)

5. Navrhněte, jak eliminovat kvantifikátory pro teorii se signaturou $\langle \emptyset, \{ \mathit{positive}_{/1}, \mathit{negative}_{/1} \} \rangle$, která definuje *positive* a *negative* jako kladnost a zápornost celých čísel.

   Rešení: x má buď nulovou, pozitivní, nebo negativní hodnotu. Stačí enumerovat tyto tři případy. Formuli $\exists x \varphi$ převedeme na ekvivalentní formuli $\varphi_- \vee \varphi_0 \vee \varphi_+$, kde: φ_- vznikne z φ nahrazením každého výskytu $\mathit{positive}(x)$ za 0 a $\mathit{negative}(x)$ za 1, φ_+ vznikne z φ nahrazením každého výskytu $\mathit{positive}(x)$ za 1 a $\mathit{negative}(x)$ za 0, a φ_0 vznikne z φ nahrazením každého výskytu $\mathit{positive}(x)$ nebo $\mathit{negative}(x)$ za 0. (například $\exists x((\mathit{positive}(x) \vee \mathit{negative}(y)) \wedge \mathit{negative}(x))$ bude $((1 \vee \mathit{negative}(y)) \wedge 0) \vee ((0 \vee \mathit{negative}(y)) \wedge 0) \vee ((0 \vee \mathit{negative}(y)) \wedge 1))$

6. Navrhněte, jak eliminovat kvantifikátory pro teorii se signaturou $\langle \{ \geq_{/2} \}, \emptyset \rangle$, která definuje predikát $\geq$ jako jakoukoliv relaci částečného uspořádání.

   Rešení: 1) Z formule $\exists x \varphi$ eliminujeme kvantifikátor tak, že z φ vytvoříme formuli ψ následovně. Převedeme do DNF. 2) Každou klauzuli *saturujeme tranzitivním uzávěrem*, t.j., pro každou dvojici (pozitivních) literálů $a \leq b$ a $b \leq c$ přidáváme literály $a \leq c$, dokud je co přidávat. 3) Sporné klauzule nahradíme za 0 (obsahují $a \leq b$ a $\neg a \leq b$). 4) Smažeme všechny predikáty s výskytem x.

   Všimněte si, že je opravdu nejdříve třeba spočítat tranzitivní uzávěr, až potom odstranit sporné klauzule, a až potom odstranit výskyty x. Musíme zaručit, aby smazání negativních literálů s výskytem x nezměnilo význam formule. (například $\exists y(y \leq z \wedge z \leq u \wedge \neg y \leq u)$ bude po spočítání tranz. uz. $(y \leq z \wedge z \leq u \wedge \neg y \leq u \wedge y \leq u)$, kde se vyskytuje spor, a proto to nakonec bude 0),

7. Navrhněte, jak eliminovat kvantifikátory pro teorii T se signaturou obsahující pouze rovnost, kde

   (a) $T = \{\neg\exists x \exists y (x \neq y)\}$

   (b) $T = \{\neg\exists x \exists y \exists z (x \neq y \wedge x \neq z \wedge y \neq z)\}$

   (c) $T = \{\neg\exists x_1 \ldots \exists x_4 \bigwedge_{1 \leq i < j \leq 4} (x_1 \neq x_j)\}$

   (d) Předchozí bod, ale libovolné $n \in \mathbb{N}$ na místo 4.

   (e) $T = \{\exists x_0 \ldots \exists x_n \bigwedge_{0 \leq i < j \leq n} x_i \neq x_j \mid n \in \mathbb{N}\}$ Těžké, ale dá se vymyslet, dá se i najít.

Rešení: (7a) Doména má jen jeden prvek. Proto se každý predikát $x = y$ vyhodnotí na 1. Z formule $\exists x\varphi$ tedy eliminujeme kvantifikátor tak, že z φ vytvoříme formuli ψ nahrazením každého atomického predikátu formy $x = y$ za 1. (např. z formule $\exists x(x = y \wedge \neg u = v) \vee \neg x = v$ bude $(1 \wedge \neg u = v) \vee \neg 1$)

(7b) Doména má maximálně dva prvky, nazvěme je a, b (mohou být jeden a ten samý prvek). Z formule $\exists x\varphi$ eliminujeme kvantifikátor tak, že z φ vytvoříme formu ψ následovně. Nechť $P = \{\{A, B\} \mid A \cup B = X \wedge A \cap B = \emptyset\}$ je množina všech rozkladů množiny X proměnných formule φ na dvě části (množinu A chápeme jako proměnné s hodnotou a, B jako proměnné s hodnotou B). Potom ψ je formule $\bigvee_{\{A,B\}\in P} \varphi_{\{A,B\}}$, kde $\varphi_{\{A,B\}}$ vznikne z φ (1) přidáním konjunktů $\bigwedge_{y,z\in A \vee y,z\in B} y = z \wedge \bigwedge_{y\in A\wedge z\in B} y \neq z$, a potom (2) nahrazením každého predikátu $x = y$ za 1 pokud $x, y \in A$ nebo pokud $x, y \in B$, a za 0 jinak. (část (1) explicitně zafixuje volbu rozkladu, aby ji později nebylo možno změnit nekonzistentně se zvolenými rovnostmi. Například $\exists x(x = y \wedge \neg y = z) \vee \neg x = z$ vygeneruje 4 disjunkty konkrétně pro rozklady $\{\{x, y, z\}, \emptyset\}$, $\{\{x, y\}, \{z\}\}$, $\{\{x, z\}, \{y\}\}$, $\{\{y, z\}, \{x\}\}$, kde třeba pro případ $\{\{x, y\}, \{z\}\}$ dostaneme (1) $((x = y \wedge \neg y = z) \vee \neg x = z) \wedge (x = y) \wedge (x \neq z \wedge y \neq z)$ a z toho (2) $((1 \wedge \neg y = z) \vee \neg 0)) \wedge (1) \wedge (1 \wedge y \neq z))$.

8. Navrhněte, jak eliminovat kvantifikátory pro teorii $T_n, n \in \mathbb{N}$ se signaturou $\langle\{P^1_{/1}, \ldots, P^n_{/1}\}, \emptyset\rangle$ a prázdnou množinou axiomů. (teorie v podstatě mluví o náležitosti v n libovolných podmnožinách domény).

   Rešení: Podobně jako příklad 7b. Je třeba uvažovat každou možnost, jak může vypadat Vennův diagram n-tice množin.

9. Mějme teorii T reálných čísel se sčítáním, násobením, rovností, a operátorem $\lfloor n \rfloor$, který vrací dolní celou část čísla n. Dokažte, že je tato teorie nerozhodnutelná.

   Rešení: Redukujeme na T Peanovu aritmetiku (T_{PA}). Všechny operátory máme (+,*,=), potřebujeme jen definovat predikát, který identifikuje celá čísla. Potom budeme snadno schopni jakoukoliv formuli Peanovy aritmetiky zakódovat/redukovat na formuli T. Redukce bude vyčíslitelná surjekce, která zachovává platnost v teorii. Protože je Peanova aritmetika nerozhodnutelná, musí být i T nerozhodnutelná. Konkrétně, redukce bude funkce f, která z formule φ jazyka teorie T_{PA} vytvoří formuli $f(\varphi)$ tak, že každou atomickou formuli ψ s proměnnými $x_1, \ldots, x_n$ nahradí za formuli $(\psi \wedge x_1 = \lfloor x_1 \rfloor \wedge \ldots \wedge x_n = \lfloor x_n \rfloor)$. (Reálné číslo je celé, právě když se rovná své dolní celé části. Omezili jsme hodnoty proměnných pouze na celá čísla, tedy právě na hodnoty přípustné v teorii T_{PA}.)

10. Mějme teorii T reálných čísel se sčítáním, rovností, a binární funkcí $mod(m, n)$, která vrátí zbytek z $m \in \mathbb{R}$ po „celočíselném“ dělení $n \in \mathbb{R}$, tedy, vrátí číslo $k \in \mathbb{R}$ takové, že $0 \leq k < n$ a pro nějaké $l \in \mathbb{N}$, $m = nl + k$. Dokažte, že je tato teorie nerozhodnutelná.

11. Dokažte, že pokud je množina speciálních axiomů úplné prvořádové teorie T částečně rozhodnutelná, pak je T rozhodnutelná teorie (t.j. množina důsledků T je rozhodnutelná).

    Rešení: Hint: Víme, že důkaz každé věty nebo její negace existuje. Stále jej časem můžeme vygenerovat, jen nemůžeme ověřovat platnost důkazů jeden po druhém, protože díky pouze částečné rozhodnutelnosti by některé tyto výpočty, kdy ověřujeme řetězec, který není důkazem, nemusely terminovat. Proto je třeba paralelně ověřovat všechny řetězce, podobně jako v důkazu částečné rozhodnutelnosti problému neprázdnosti TS (viz. cvičení, skeny poznámek TV).)

12. Uvažujme systém podobný abstraktnímu formálnímu systému z přednášky o Tarského a Gödelě větě (tento bude jednodušší)[1]: *Predikáty* P jsou podmnožinou množiny *výrazů* $V \supseteq P$. Pro predikát H a výraz X, výraz HX je *věta*, která je interpretována jako „X je prvkem množiny *identifikované* predikátem H“, a tato věta je buď *pravdivá* nebo *nepravdivá*. Podmnožina V je *identifikovatelná*, pokud je identifikovaná nějakým predikátem. Předpokládáme, (1) že pro každý predikát H existuje predikát H' takový, že pro každý výraz X, HX je pravda, právě když $H'X$ není pravda, a (2) že existuje jeho *diagonalizace*, predikát $H^\sharp$ takový, že $H^\sharp X$ je pravda, právě když HXX je pravda (HXX je H aplikované na XX). Věta X je *pevným bodem* predikátu H, pokud HX je pravda, právě když X je pravda (X a HX jsou *ekvivalentní*).

    - Ukažte, že každý predikát má pevný bod.

[1] příklad z knihy The Gödelian Puzzle Book Raymonda M. Smullyana. Je tam i řešení

- Ukažte, že množina pravdivých vět není identifikovatelná. (Všimněte si, že každá věta by byla pevným bodem predikátu identifikujícího množinu pravdivých vět. Je třeba zkonstruovat predikát, který jeho pevným nemůže být. Viz Tarského věta a její důkaz ve slajdech.)
- Vaše řešení z předchozího bodu bude pravděpodobně pevným bodem H', délky 6 znaků z $\{H, ', \sharp\}$, kde H je hypotetický predikát identifikující pravdivé věty. Najděte další pevné body H' stejné délky. Bylo by je také možné použít v předchozím bodě?

Mějme množinu dokazatelných vět identifikovanou predikátem D a množinu vyvratitelných vět identifikovanou predikátem R. Dokazatelné věty jsou pravdivé, vyvratitelné jsou nepravdivé (systém je korektní). Dokažte, že

- Existuje pravdivá nedokazatelné věta. (Viz. abstraktní Gödelova věta ve slajdech. Potřebujete najít větu, která je pravdivá, ale nedokazatelná. Bude to pevný bod něčeho, čeho? Bude se skládat ze symbolů $D, ', \sharp$)
- Existuje nepravdivá nevyvratitelná věta.

20 Asymptotická složitost

1. Rozhodněte a dokažte, zda platí následující:
   (a) $300n^4 + 900n^2 + 1 \in \mathcal{O}(n^4 - 100n^3)$,
   (b) $300n^4 - 900n^2 + 1 \in \mathcal{O}(300n^3 + 100n^2)$.

   Řešení a): Toto tvrzení platí. Stačí najít hodnoty $c \in \mathbb{R}^+$ a $n_0 \in \mathbb{N}$, takové že

   $$\forall n \geq n_0 : 300n^4 + 900n^2 + 1 \leq c \cdot (n^4 - 100n^3)$$

   Zvolme například $c = 301$ a požadované n_0 dopočítáme (omezme se na $n_0 > 0$, abychom mohli níže krátit).

   $$\forall n \geq n_0 : 300n^4 + 900n^2 + 1 \leq 301n^4 - 30100n^3$$
   $$\forall n \geq n_0 : 30100n^3 + 900n^2 + 1 \leq n^4$$
   $$\forall n \geq n_0 : 30100 + 900n^{-1} + 1n^{-3} \leq n$$

   Vidíme, že pro $n \geq 30101$ požadovaná nerovnost očividně platí.

   Řešení b): Toto tvrzení neplatí. Z definice $\mathcal{O}()$ stačí ukázat, že

   $$\lim_{n\to\infty} \frac{c \cdot (300n^3 + 100n^2)}{300n^4 - 900n^2 + 1} = 0$$

   Opakovanou aplikací L'Hospitalova pravidla (čitatel i jmenovatel jdou v limitě k ∞) dostáváme

   $$\lim_{n\to\infty} \frac{c}{4n} = 0,$$

   jelikož c je konstanta.

2. Rozhodněte a dokažte, jaký vztah platí mezi následujícími třídami funkcí:
   (a) $\mathcal{O}(n^3)$ a $\mathcal{O}(n^4)$,
   (b) $\mathcal{O}(n \cdot \log n)$ a $\mathcal{O}(n^2)$,
   (c) $\mathcal{O}(n \cdot \sin n)$ a $\mathcal{O}(n^2)$,
   (d) $\mathcal{O}(2^n)$ a $\mathcal{O}(2^{n^2})$,
   (e) * $\mathcal{O}(2^n)$ a $\mathcal{O}(n!)$,
   (f) $\mathcal{O}(n^2)$ a $\mathcal{O}(2^{\ln n})$,
   (g) $\mathcal{O}(2^{2^n})$ a $\mathcal{O}(2^{n^2})$,

(h) $\mathcal{O}(\ln n)$ a $\mathcal{O}(n \cdot \ln \ln n)$,

(i) $\mathcal{O}(\ln n)$ a $\mathcal{O}(\log n)$.

Řešení a): Platí, že $\mathcal{O}(n^3) \subset \mathcal{O}(n^4)$. Stačí ukázat (podobně jako výše), že např. $n^4 \in \mathcal{O}(n^4)$, ale $n^4 \notin \mathcal{O}(n^3)$, a rovněž, že $\forall f \in \mathcal{O}(n^3) : f \in \mathcal{O}(n^4)$. K tomu využijeme fakt, že pro $\forall f \in \mathcal{O}(n^3)$ existují hodnoty $c \in \mathbb{R}^+$ a $n_0 \in \mathbb{N}$ takové, že

$$\forall n \geq n_0 : f(n) \leq c \cdot n^3,$$

a zároveň, že existují hodnoty $c' \in \mathbb{R}^+$ a $n_0' \in \mathbb{N}$, takové že

$$\forall n \geq n_0' : c \cdot n^3 \leq c' \cdot n^4,$$

z čehož získáváme, že $f \in \mathcal{O}(n^4)$ (uvážíme maximum z n_0 a n_0').

21 Konstrukce TS s danou složitostí

1. Snažte se najít co nejmenší k takové, aby platily následující vztahy. Dokažte, že vztah platí, ale nedokazujte, že vaše k je opravdu nejmenší možné.

   (a) $L' \in DTIME[n^2] \Rightarrow L \in DTIME[n^k]$, kde
   $L = \{w \in \Sigma^* \mid$ pro všechny prefixy w' slova w platí, že $w' \in L'\}$.

   (b) $L' \in DTIME[n^2] \Rightarrow L \in NTIME[n^k]$, kde
   $L = \{w \in \Sigma^* \mid$ pro všechny prefixy w' slova w platí, že $w' \in L'\}$.

   (c) $L' \in DTIME[n^3] \Rightarrow L \in DTIME[n^k]$, kde
   $L = \{w \in \Sigma^* \mid w$ obsahuje dvě různá podslova w_1, w_2 takové, že $w_1, w_2 \in L'\}$.

   Řešení a): Ukážeme, jak deterministický TS T akceptující jazyk L může efektivně fungovat a podle toho určíme hodnotu k. T pro vstup w (kde $|w| = n$) musí zkontrolovat všechny prefixy w' – těch je $\mathcal{O}(n)$ a jejich velikost je taktéž v $\mathcal{O}(n)$. Kontrola každého w' je realizována simulací deterministického T' akceptující jazyk L', který dle zadání pracuje v čase $\mathcal{O}(n^2)$. Celkově složitost TS T je $\mathcal{O}(n) \cdot \mathcal{O}(n^2) = \mathcal{O}(n^3)$. Tudíž $k = 3$.

2. Rozhodněte a dokažte, zda třída **P** je uzavřena na následující operace:

   (a) $\diamond L = \{w \in L \mid w = a^n b^n c^n \wedge n \geq 0\}$,

   (b) $\diamond L = \{w \in L \mid \ |w| = 2^n \wedge n \geq 0\}$,

   (c) $\diamond L = \{w \in L \mid w$ je kód TS M, pro který platí, že $w \in L(M)\}$.

   Řešení a): Třída **P** je uzavřena na operaci $\diamond$. $\diamond L = L \cap L'$, kde $L' = \{a^n b^n c^n \mid n \geq 0\}$. Připomeňme, že **P** je uzavřena na průnik (TS T akceptující $L_1 \cap L_2$ bude simulovat stroje pro L_1 a L_2. Složitost T je dána součtem složitostí strojů pro L_1 a L_2 – součet dvou polynomů je zase polynom a tudíž T bude pracovat též v polynomiálním čase). Tudíž nám stačí ukázat, že $L' \in \mathbf{P}$. Požadovaný stroj M bude mít 4 pásky. Nejdříve projde vstupní pásku, zkontroluje správné pořadí písmen a překopíruje jednotlivá písmena na zvláštní pásky (např. všechny znaky a budou na 2. pásce, všechny znaky b budou na 3. pásce a všechny znaky c budou na 4. pásce). Toto je možné udělat pomocí jedno lineárního průchodu vstupem tj. v $\mathcal{O}(n)$. Druhým lineárním průchodem zkontroluje, jestli je na páskách 2–4 stejný počet symbolů a podle toho akceptuje či zamítá. Je zřejmé, ze stroj pracuje v $\mathcal{O}(n)$.

22 Složitost algoritmů a amortizovaná složitost

1. Uvažme k-bitový čítač implementovaný pomocí binárního pole A o velikosti k (pole je indexované od 1 s nejvýznamnějším bitem na pozici 1). Čítač je inicializovaný na hodnotu 0. Dále uvažme (jedinou) operaci $inc(A, k)$, která inkrementuje k-bitový čítač A o hodnotu 1 (viz pseudokód níže).

(a) Analyzujte a zdůvodněte asymptotickou časovou složitost operace $inc(A, k)$ v nejhorším případě.

(b) Uvažme posloupnost n operací $inc(A, k)$, kde $n < 2^k$. Analyzujte a zdůvodněte amortizovanou časovou složitost této posloupnosti operací.

```
Function inc(A, k)
    while k ≥ 1 ∧ A[k] = 1 do
        A[k] := 0
        k := k − 1
    if k ≥ 1 then A[k] := 1
```

Nápověda: Pro amortizovanou složitost doporučujeme využít metodu účtů. Zkuste si napsat delší sekvenci operací, určit reálnou cenu každé operace a zamyslete se, jak tuto cenu pokrýt pomocí vhodně zvolených kreditů získaných z předchozích operací.

Řešení a): V nejhorším případě jsou v poli A samé jedničky. V tom případě je cena operace $inc(A, k)$ v $\mathcal{O}(k)$, jelikož while cyklus proběhne k-krát. b) Hlavní myšlenka analýzy amortizované složitosti spočívá v předplacení operace "překlopení" $1 \to 0$, která se musí v drahých instancích $inc(A, k)$ volat vícekrát. Počet operací $1 \to 0$ přímo souvisí předchozím voláním operací "překlopení" $0 \to 1$ a tudíž při provedení této operace si budeme budovat potřebný kredit.

operace	kredit	cena
$1 \to 0$	0	1
$0 \to 1$	2	1

Pokud $inc(A, k)$ inkrementuje ("překlopí" $0 \to 1$) nejvyšší bit na pozici i, operace $1 \to 0$ musela být předtím volána $(k-i)$-krát. Na účtě ale muselo být v této situaci aspoň $(k-i)$ kreditů, jelikož na posledních $(k-i)$ bitech (pozicích) byly jedničky. Také si uvědomme, že po dokončení volání $inc(A, k)$ bude na účtě aspoň 1 kredit za inkrementaci i-tého bitu. Na účtě může být samozřejmě i více kreditů pokud jsou jedničky na nějakých vyšších bitech. Jednoduše nahlédneme, že během celého výpočtu platí invariant, že na účtě je tolik kreditů, kolik je v poli jedniček a tudíž kredity na účtě nikdy nemůže klesnout pod 0.

Celková amortizovaná složitost sekvence volání $inc(A, k)$ je dána maximálním počtem kreditů nutných na zaplacení jedné operace a počtem volání. Tudíž pro n volání $inc(A, k)$ dostáváme složitost $\max\{0, 2\} \cdot n = 2n \in \mathcal{O}(n)$. Připomeňme, že při použití analýzy v nejhorším případě, by složitost n volání byla $\mathcal{O}(n \cdot k)$.

2. Nechť A, B jsou pole o velikosti n (indexované od 1), která obsahují přirozená čísla. Uvažme následující rozhodovací problém:

$$P(A, B, n) = true \Leftrightarrow \exists i \in \{1, \ldots, n\} : \forall j \in \{1, \ldots, n\} : A[i] \geq B[j]$$

Níže uvedený algoritmus $dominate(A, B, n)$ řeší tento problém.

(a) Analyzujte asymptotickou časovou složitost algoritmu *dominate* v nejhorším případě.

(b) Analyzujte asymptotickou časovou složitost algoritmu *dominate* v nejlepším případě.

(c) Navrhněte algoritmus $dominate^+(A, B, n)$, který řeší tento problém a má lepší asymptotickou časovou složitost v nejhorším případě než algoritmu *dominate*. Analyzujte asymptotickou časovou složitost algoritmu $dominate^+$ v nejhorším případě.

```
Function dominate(A, B, n)
    found := false
    i := 1
    while i ≤ n ∧ found = false do
        found := true
        j := 1
        while j ≤ n ∧ found = true do
            if A[i] < B[j] then
                found := false
            j := j + 1
        i := i + 1
    return found
```

Řešení: Algoritmus *dominate* zjišťuje, jestli v poli A existuje prvek, který je větší nebo roven než všechny prvky v poli B. a) Uvažme vstup, kde poslední prvek v poli B je větší než všechny prvky v poli A a ostatní prvky v B jsou menší než všechny prvky v A. V tomto nejhorším případě se vnější cyklus algoritmu *dominate* provede n-krát (algoritmus musí projít všechny prvky v poli A). Navíc v každé iteraci vnějšího cyklu se vnitřní cyklus rovněž provede n-krát (algoritmus musí vždy projít všechny prvky v B, než ukončí vnitřní cyklus). Celkově tedy dostáváme složitost $\mathcal{O}(n^2)$. b) Uvažme vstup, kde první prvek v poli A je větší než všechny prvky v poli B. V tomto nejlepším případě se vnější cyklus algoritmu *dominate* provede pouze jednou a vnitřní cyklus se provede n-krát (algoritmus musí ověřit, že první prvek v A je

skutečně větší než všechny prvky v B. Celkově tedy dostáváme složitost $\mathcal{O}(n)$. Uvažme algoritmus, který nejdříve najde nevetší prvek v poli A (označme ho $A_{\max}$) a největší prvek v poli B (označme ho $B_{\max}$) a pak vrací *true* pokud $A_{\max} \geq B_{\max}$. Jelikož hledání maxima má složitost $\mathcal{O}(n)$, navržený algoritmus má i v nejhorším případě složitost $\mathcal{O}(n)$.

3. Uvažte následující implementaci FIFO fronty Q pomocí dvou zásobníku S_1 a S_2:

   - Q.ENQUEUE(x) : S_1.PUSH(x)
   - Q.DEQUEUE() :

   ```
   if S2.isEmpty() then
   |  while S1.isNonEmpty() do
   |  |  S2.push(S1.top())
   |  |  S1.pop()
   if S2.isNonEmpty() then
   |  S2.pop()
   ```

   (a) Analyzujte a zdůvodněte asymptotickou časovou složitost operace Q.DEQUEUE() v nejhorším případě.

   (b) Uvažme libovolnou posloupnost n operací Q.ENQUEUE(x) a Q.DEQUEUE(). Analyzujte a zdůvodněte amortizovanou časovou složitost této posloupnosti operací.

4. Uvažme algoritmus *middle*(A, n), který má na vstupu pole A (indexované od 1), které obsahuje n různých přirozených čísel. Algoritmus vrací k-tý nejvyšší prvek v A, pro $k = \lfloor n/2 \rfloor$ ($\lfloor n/2 \rfloor$ je celá část podílu $n/2$).

   (a) Analyzujte asymptotickou časovou složitost algoritmu *middle* v nejhorším případě.

   (b) Analyzujte asymptotickou časovou složitost algoritmu *middle* v nejlepším případě.

   (c) Navrhněte algoritmus *middle*$^+(A, n)$, který řeší stejný problém a má lepší asymptotickou časovou složitost v nejhorším případě. Analyzujte tuto složitost.

```
Function middle(A, n)
|  res := ∞
|  i := 1
|  while i ≤ ⌊n/2⌋ do
|  |  tmp := −1
|  |  j := 1
|  |  while j ≤ n do
|  |  |  if tmp < A[j] < res then
|  |  |  |  tmp := A[j]
|  |  |  j := j + 1
|  |  res := tmp
|  |  i := i + 1
|  return res
```

23 Třídy složitosti a NP-úplnost

1. Pro následující jazyky/problémy dokažte, že

   - patří do **NP**,
   - patří do **EXP** (v důkaze nevyužívejte fakt, že **NP** $\subseteq$ **EXP**),
   - jsou **NP**-těžké:

   (a) $SAT_{16} = \{\Phi \mid \Phi$ je výroková formule v konjunktivní normální formě, pro kterou existuje alespoň 16 různých splnitelných valuací $\}$.

   (b) *Nezávislá množina vrcholů* grafu $G = (V, E)$ je množina vrcholů $U \subseteq V$ taková, že žádné dva vrcholy z U nejsou spojeny hranou, tj., $\forall\{v_1, v_2\} \in E : \{v_1, v_2\} \not\subseteq U$. *Problém nezávislé množiny* (INDEPENDENT-SET):

   Má graf G nezávislou množinu velikosti alespoň m?

   Pro důkaz **NP**-těžkosti použijte redukci z 3-SAT.

   (c) Mějme konečnou množinu R, váhovou funkci $w : R \to \mathbb{Z}$ a cenovou funkci $v : R \to \mathbb{Z}$. Dále mějme maximální váhu $W \in \mathbb{Z}$ a minimální cenu $V \in \mathbb{Z}$. *Problém batohu* (KNAPSACK) je definovaný následovně:

Existuje $R' \subseteq R$ taková, že $\sum_{r \in R'} w(r) \leq W$ a zároveň $\sum_{r \in R'} v(r) \geq V$?

Pro důkaz **NP**-těžkosti použijte redukci z problému SUBSET-SUM (probíráno na cvičení).

(d) * Graf $G = (V, E)$ má *chromatické číslo* c, pokud existuje funkce *obarvení* $f : V \to \{1, \ldots, c\}$ taková, že pro každé dva sousední uzly platí, že *nejsou* obarveny stejnou barvou, tj. $\forall \{v_1, v_2\} \in E : f(v_1) \neq f(v_2)$. *Problém chromatického čísla grafu* (CHROMATIC-NUMBER) je definovaný následovně:

Má graf G chromatické číslo c?

Pro důkaz **NP**-těžkosti použijte redukci z problému 3-SAT.

(e) *Klikové pokrytí* grafu $G = (V, E)$ je množina klik K taková, že G je sjednocení klik z K. *Problém klikového pokrytí* (CLIQUE-COVER) je definovaný následovně:

Existuje množina klik K o velikosti maximálně m taková, že K je klikové pokrytí G?

Pro důkaz **NP**-těžkosti použijte redukci z problému CHROMATIC-NUMBER.

Řešení pro příklad a): $SAT_{16} \in \mathbf{NP}$: Nedeterministický vícepáskový TS T, který rozhoduje SAT_{16} v polynomiálním čase, pracuje následovně. T nejdříve nedeterministicky vygeneruje 16 valuací pro proměnné ve formuli Φ. Každou valuaci zapíše na jednu pásku (pro jednoduchost uvažujeme, že jsou valuace zapsané na pásce dle uspořádaní proměnných). Potom T ověří, že valuace jsou různé a že jsou všechny splnitelné. Toto se zajisté dá realizovat v polynomiálním čase (tj. existuje polynomiální verifikátor). Zjištění pravdivosti formule pro danou valuaci vyžaduje lineární průchod formulí. Zjištění, že se daná valuace liší od ostatních 15 valuací je taktéž lineární. Pokud T ověří, že valuace jsou různé a všechny jsou splnitelné, tak akceptuje Φ, jinak zamítá.

$SAT_{16} \in \mathbf{EXP}$: Deterministický vícepáskový TS T, který rozhoduje SAT_{16} v exponenciálním čase, pracuje následovně. T postupně generuje (např. v lexikografickém pořadí) jednotlivé valuace proměnných z formule Φ. Pokud Φ obsahuje n proměnných, pak existuje 2^n různých valuací. Pro každou valuaci T zkontroluje, jestli Φ je splnitelné (lineární průchod formulí). Pokud ano, T zvýší čítač splnitelných valuací. Pokud čítač dosáhne hodnotu 16, T akceptuje. Pokud prošel všechny valuace a neakceptoval, tak zamítne. Složitost T je v $\mathcal{O}(2^n \cdot n) \subseteq \mathcal{O}(2^{n^2})$. Připomeňme, že $SAT_{16} \in \mathbf{EXP}$ plyne taktéž přímo z $SAT_{16} \in \mathbf{NP}$.

SAT_{16} je **NP**-těžký: Sestrojíme polynomiální redukci δ, která ukáže, že $SAT \leq_P SAT_{16}$. Jelikož SAT je **NP**-úplný, tak pak dostaneme požadované. Pro správně zformované formule Φ definujme

$$\delta(\Phi) = \Phi \wedge (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)$$

kde x_1, x_2, x_3, x_4 a x_5 jsou čerstvé proměnné, které se ve formuli Φ nevyskytují. Pro nesprávně zformované vstupy redukce stačí vrátit $x_1 \wedge \neg x_1$.

Konstrukce formule $\delta(\Phi)$ je lineární k Φ (včetně správnosti správného zformování stačí jeden průchod formulí). Její korektnost dokážeme takto. Pokud $\Phi \in SAT$, tak existuje aspoň jedna splnitelná valuace v. Klauzule $K = (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)$ má $2^5 - 1 = 31 \geq 16$ splnitelných valuací. Tudíž $\Phi \wedge (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)$ má aspoň 31 splnitelných valuací (zkombinujeme valuaci v s valuacemi pro klauzuli K) a tedy $\Phi \wedge K \in SAT_{16}$. Naopak, pokud $\Phi \notin SAT$, tak a) pro nesprávně zformovaný vstup $x_1 \wedge \neg x_1 \notin SAT_{16}$ a b) pro správně zformovaný vstup $\Phi \wedge (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \notin SAT_{16}$, jelikož podformule Φ není splnitelná. Celkově dostáváme, že

$$\Phi \in SAT \iff \delta(\Phi) \in SAT_{16}.$$