

456-330/1: Teoretická informatika (TI) přednáška 3

prof. RNDr Petr Jančar, CSc.

katedra informatiky FEI VŠB-TUO
www.cs.vsb.cz/jancar

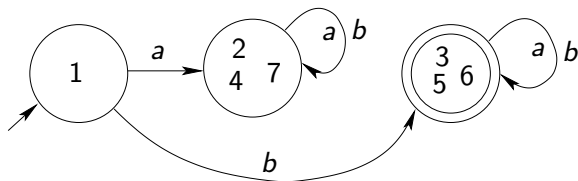
LS 2009/2010

Redukce konečných automatů

| | a | b |
|-----|---|---|
| → 1 | 2 | 3 |
| 2 | 2 | 4 |
| 3 | 3 | 5 |
| 4 | 2 | 7 |
| 5 | 6 | 3 |
| 6 | 6 | 6 |
| 7 | 7 | 4 |

$$I = \{1\}$$
$$II = \{2, 4, 7\}$$
$$III = \{3, 5, 6\}$$

| | a | b |
|-----|-----|-----|
| → I | II | III |
| II | II | II |
| III | III | III |



Podílový automat (faktor-automat) – připomenutí

K $A = (Q, \Sigma, \delta, q_0, F)$ definujeme ekvivalenci \sim na množině Q takto:

$$q \sim q' \iff df \quad L_q^{toAcc} = L_{q'}^{toAcc}$$

Podílový automat podle ekvivalence \sim , označený A_\sim , definujeme takto (píšeme stručněji $[q]$ místo $[q]_\sim$):

$A_\sim = (Q_\sim, \Sigma, \delta_\sim, [q_0], F_\sim)$, kde

$Q_\sim = \{ [q] \mid q \in Q \}$,

$F_\sim = \{ [q] \mid q \in F \}$

$\delta_\sim([q], a) = [\delta(q, a)]$

Korektnost definice (co to je?) plyne z

$(p \sim q \Rightarrow (p \in F \Leftrightarrow q \in F))$ a z $(p \sim q \Rightarrow \delta(p, a) \sim \delta(q, a))$.

Jak ukážeme, že $L(A) = L(A_\sim)$?

Stačí (induktivně dle délky slova) dokázat

$$q \xrightarrow{w}_A q' \Leftrightarrow [q] \xrightarrow{w}_{A_\sim} [q']$$

....

(Jazyková) ekvivalence stavů a automatů

Nalezněme všechny dvojice stavů q, q' , pro něž platí $L_q^{toAcc} = L_{q'}^{toAcc}$.

| | a | b |
|----|---|---|
| →0 | 0 | 1 |
| ←1 | 1 | 2 |
| ←2 | 3 | 1 |
| 3 | 2 | 4 |
| 4 | 2 | 3 |

| | a | b |
|----|---|---|
| →5 | 5 | 6 |
| 6 | 7 | 5 |
| ←7 | 7 | 9 |
| 8 | 9 | 8 |
| ←9 | 8 | 7 |

Poznámka (pro hloubavější): koinduktivní přístup

Na naši konstrukci relace $\sim \subseteq Q \times Q$
definované vztahem

$$q \sim q' \Leftrightarrow L_q^{toAcc} = L_{q'}^{toAcc}$$

(pro automat $A = (Q, \Sigma, \delta, q_0, F)$), se lze dívat takto:

vezmeme (největší) relaci $R_0 = Q \times Q$,
postupně aplikujeme jistý **monotónní funkcionál** $\mathcal{F} : 2^{Q \times Q} \rightarrow 2^{Q \times Q}$
(pro $T_1 \subseteq T_2$ máme $\mathcal{F}(T_1) \subseteq \mathcal{F}(T_2)$)

dostáváme tak relace $R_0 \supseteq R_1 \supseteq R_2 \supseteq \dots$
kde $R_1 = \mathcal{F}(R_0)$, $R_2 = \mathcal{F}(R_1)$, ...

až se dostaneme k (největšímu) **pevnému bodu** R (pro nějž $R = \mathcal{F}(R)$).

Rozhodování ekvivalence automatů

Věta. Existuje algoritmus, který pro zadané konečné automaty A_1, A_2 rozhodne, zda $L(A_1) = L(A_2)$.

Jedna idea algoritmu (umíte ji dotáhnout?):

$$L(A_1) = L(A_2) \iff (L(A_1) - L(A_2)) \cup (L(A_2) - L(A_1)) = \emptyset$$

Máte jiný (lepší) nápad?

K A_1, A_2 zkonstruujeme ekvivalentní redukované automaty *v normovaném tvaru* a ty porovnáme.

(Když jsou stejné, tvrdíme $L(A_1) = L(A_2)$, jinak $L(A_1) \neq L(A_2)$.)

(Je to korektní??)

Definice. Konečný automat A je *minimální*, jestliže neexistuje automat A' , který má méně stavů než A a přitom $L(A) = L(A')$.

Věta. Je-li automat redukovaný, pak je minimální.

Věta. Dva minimální automaty, které přijímají tentýž jazyk, jsou izomorfní to také znamená, že mají stejný normovaný tvar.

- *zřetězení*

$$L_1 \cdot L_2 = \{uv \mid u \in L_1, v \in L_2\}$$

- *iterace*

$$L^* = \bigcup_{n=0}^{\infty} L^n,$$

kde L^n je definováno induktivně:

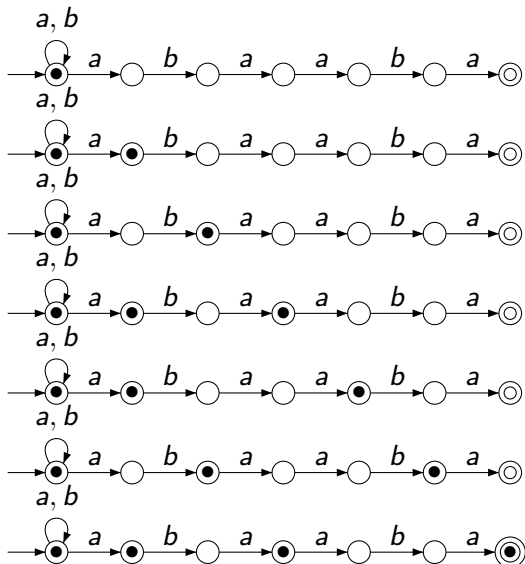
$$L^0 = \{\varepsilon\}, L^{n+1} = L \cdot L^n.$$

Pozn. L^* lze definovat také např. takto:

$$L^* = \left\{ w \mid \text{ex. } n \geq 0 \text{ a slova } u_1, u_2, \dots, u_n \in L \text{ tak, že} \right. \\ \left. w = u_1 u_2 \dots u_n \right\}.$$

Otázka. Je třída reg. jazyků uzavřena na tyto operace ?

Nedeterministické konečné automaty (NKA)



Zobecněné NKA (s ε -přechody)

Příklad: pro regulární výraz $(aa)^*(bb)^*(cc)^* \dots$

$A = (Q, \Sigma, \delta, I, F)$, kde ...

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q).$$

Induktivní definice (ternární relace) $q \xrightarrow{w} q'$:

- 1 $q \xrightarrow{\varepsilon} q$,
- 2 $(\delta(q, a) \ni q', \text{ pro } a \in \Sigma \cup \{\varepsilon\}, a q' \xrightarrow{v} q'') \implies q \xrightarrow{av} q''$.

$$L(A) = \{w \in \Sigma^* \mid \exists q \in I : q \xrightarrow{w} F\}.$$

Věta. Existuje algoritmus, který ke každému ZNKA A sestrojí ekvivalentní (deterministický) konečný automat A' (tedy $L(A) = L(A')$).

(Některé) uzávěrové vlastnosti třídy REG

Věta.

Jestliže $L \subseteq \Sigma^*$ je regulární jazyk, pak

- doplněk $\bar{L} = \Sigma^* - L$ je regulární jazyk.

Jestliže L_1, L_2 jsou regulární jazyky, pak

- $(L_1 \cup L_2)$ je regulární jazyk,
- $(L_1 \cap L_2)$ je regulární jazyk,
- $(L_1 - L_2)$ je regulární jazyk.

Jinými slovy

Věta.

Třída regulárních jazyků, označme ji REG, je uzavřena vůči booleovským množinovým operacím (... a sice **konstruktivně** ...).

... a také vůči **zřetězení**, **iteraci**, **zrcadlovému obrazu**, ...

příklad: $(01)^*111 + (00 + 1)^*$

$RV(\Sigma)$... množina *regulárních výrazů* nad Σ ; je to jistá množina řetězců v abecedě $\Sigma \cup \{ \emptyset, \epsilon, +, \cdot, *, (,) \}$.

Induktivní definice:

- $\emptyset \in RV(\Sigma)$, $\epsilon \in RV(\Sigma)$, $a \in RV(\Sigma)$ (pro vš. $a \in \Sigma$);
- když $\alpha, \beta \in RV(\Sigma)$, pak také
 - $(\alpha + \beta) \in RV(\Sigma)$,
 - $(\alpha \cdot \beta) \in RV(\Sigma)$,
 - $(\alpha^*) \in RV(\Sigma)$.

Reg. výraz α reprezentuje jazyk $[\alpha]$:

$$[\emptyset] = \emptyset, \quad [\epsilon] = \{\epsilon\}, \quad [a] = \{a\},$$

$$[(\alpha + \beta)] = [\alpha] \cup [\beta],$$

$$[(\alpha \cdot \beta)] = [\alpha] \cdot [\beta],$$

$$[(\alpha^*)] = [\alpha]^*.$$

Příklad regulárního výrazu (podle definice)

$$((((0 \cdot 1)^* \cdot 1) \cdot (1 \cdot 1)) + ((0 \cdot 0) + 1)^*)$$

Zjednodušíme:

- vynecháváme vnější závorky
- vynecháváme znak \cdot pro zřetězení
- využíváme asociativitu operace zřetězení pro vynechání závorek
- další vynechání závorek umožňuje dohodnutá priorita operátorů:
 - * má vyšší prioritu než \cdot
 - \cdot má vyšší prioritu než $+$

Uvedený výraz lze tedy psát

$$(01)^*111 + (00 + 1)^*$$

$$(1^*0 + 10)^*1$$

syntaktický strom

(jeden možný) lineární zápis

$$\text{Conc}(\text{Iter}(\text{Union}(\text{Conc}(\text{Iter}(1), 0), \text{Conc}(1, 0))), 1)$$