460-4005/01: Teoretická informatika (TI)
přednáška 3

prof. RNDr Petr Jančar, CSc.

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

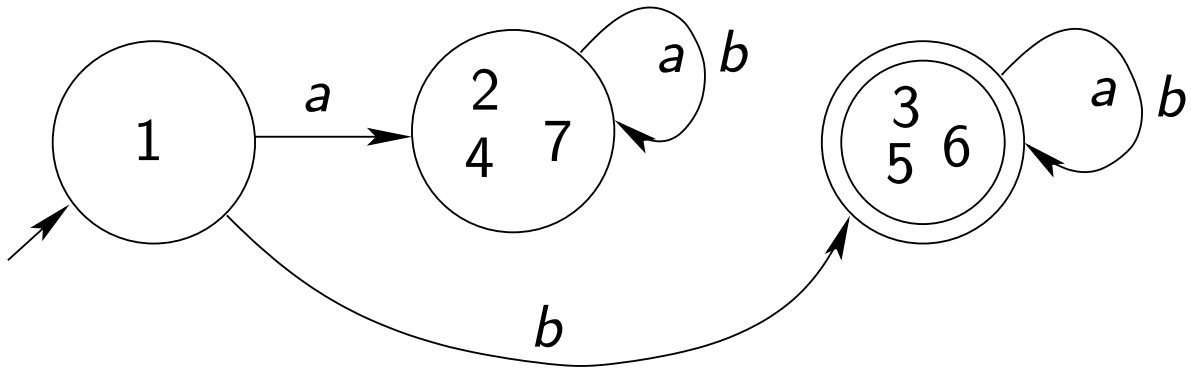
LS 2010/2011

Redukce konečných automatů

	a	b
$\rightarrow$ 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
$\rightarrow$ I	II	III
II	II	II
(III)	III	III



Podílový automat (faktor-automat) – připomenutí

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

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

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
$\big(p \sim q \Rightarrow (p \in F \Leftrightarrow q \in F)\big)$ a z $\big(p \sim q \Rightarrow \delta(p, a) \sim \delta(q, a)\big)$.
Jak ukážeme, že $L(A) = L(A_\sim)$?
Stačí (induktivně dle délky slova) dokázat

- $q \xrightarrow{w}_A q' \;\Rightarrow\; [q] \xrightarrow{w}_{A_\sim} [q']$
- $[q] \xrightarrow{w}_{A_\sim} [q'] \;\Rightarrow\; \exists r' : r' \sim q', q \xrightarrow{w}_A r'$

(Jazyková) ekvivalence stavů a automatů

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

	a	b
$\rightarrow 0$	0	1
$\leftarrow 1$	1	2
$\leftarrow 2$	3	1
3	2	4
4	2	3

	a	b
$\rightarrow 5$	5	6
6	7	5
$\leftarrow 7$	7	9
8	9	8
$\leftarrow 9$	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 $\mathcal{F} : 2^{Q \times Q} \longrightarrow 2^{Q \times Q}$
(pro $T_1 \subseteq T_2$ máme $\mathcal{F}(T_1) \subseteq \mathcal{F}(T_2)$)

dostáváme tak relace $R_0 \supseteq R_1 \supseteq R_2 \supseteq \cdots$
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?):

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

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.

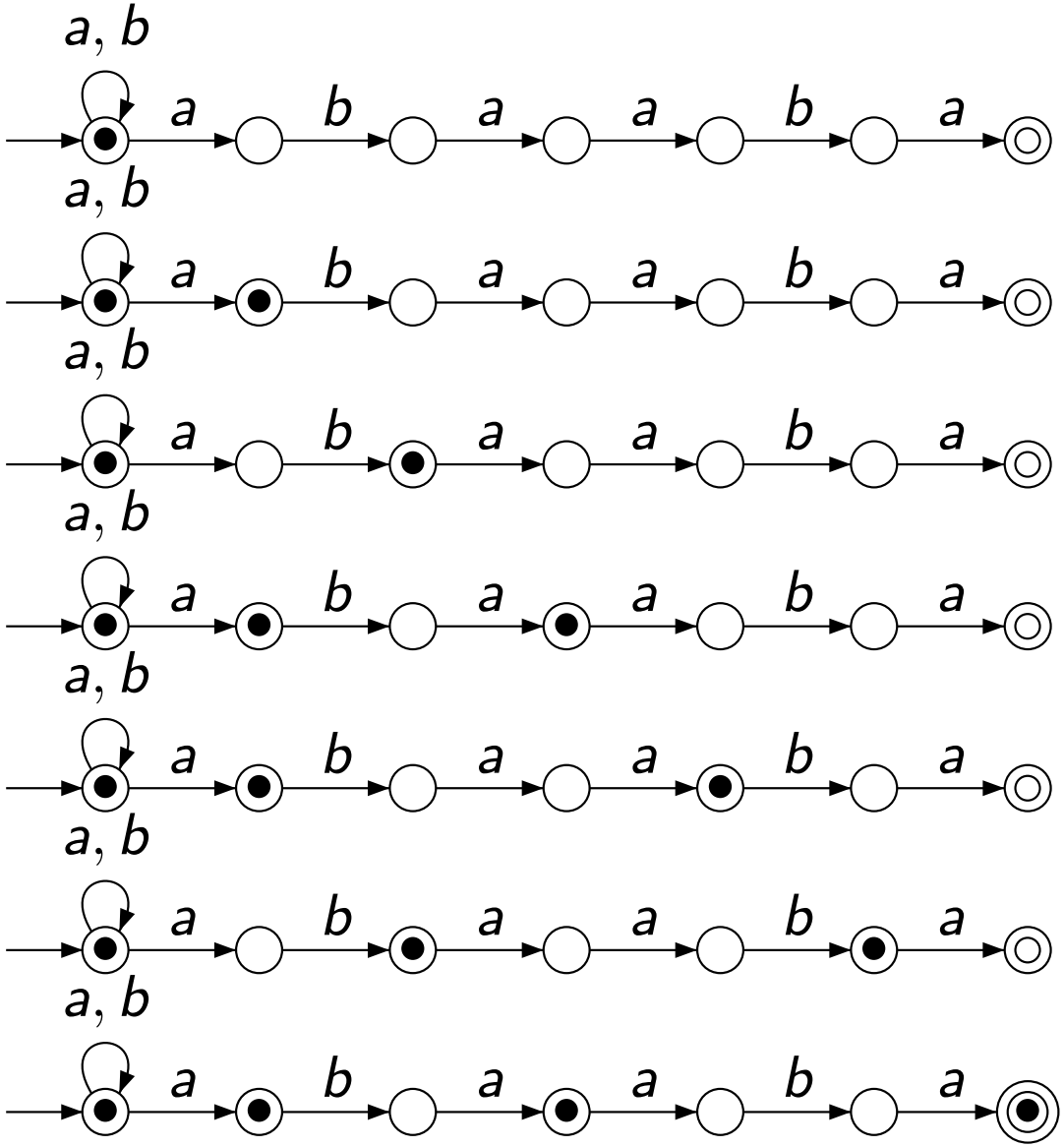
Operace zřetězení jazyků; operace iterace (hvězdička)

- *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:
$L^* = \{\, w \mid$ ex. $n \geq 0$ a slova $u_1, u_2, \ldots, u_n \in L$ tak, že
$w = u_1 u_2 \ldots u_n \,\}$.

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

Nedeterministické konečné automaty (NKA)



Zobecněné NKA (s ε-přechody)

Příklad: pro regulární výraz $(aa)^*(bb)^*(cc)^*$ ...
$A = (Q, \Sigma, \delta, I, F)$, kde ...

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

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

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

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

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 $\overline{L} = \Sigma^* - L$ 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, ...

Regulární výrazy

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 $[\alpha]$:

$$[\emptyset] = \emptyset, \quad [\epsilon] = \{\varepsilon\}, \quad [a] = \{a\},$$
$$[(\alpha + \beta)] = [\alpha] \cup [\beta],$$
$$[(\alpha \cdot \beta)] = [\alpha] \cdot [\beta],$$
$$[(\alpha^*)] = [\alpha]^*.$$

Konvence při psaní regulárních výrazů

Příklad regulárního výrazu (podle definice)

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

Zjednodušujeme:

- vynecháváme vnější závorky
- vynecháváme znak $\cdot$ pro zřetězení
- využíváme asociativitu oprace 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
$(01)^*111 + (00 + 1)^*$

Kostrukce ZNKA k regulárnímu výrazu

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

syntaktický strom ....

(jeden možný) lineární zápis

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