Dnešní téma

Nedeterministické ϵ–NFA, Operace zachovávající regularitu

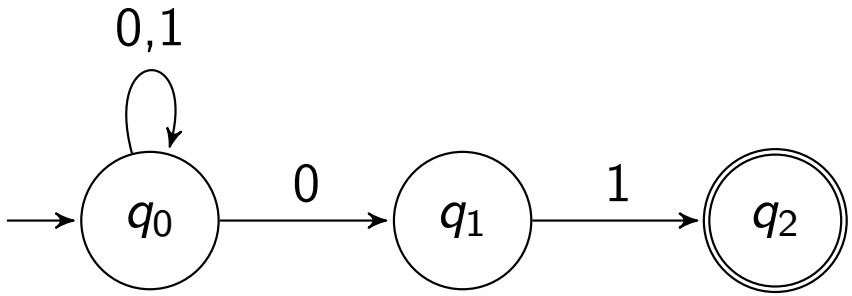
Marta Vomlelová

MS 303

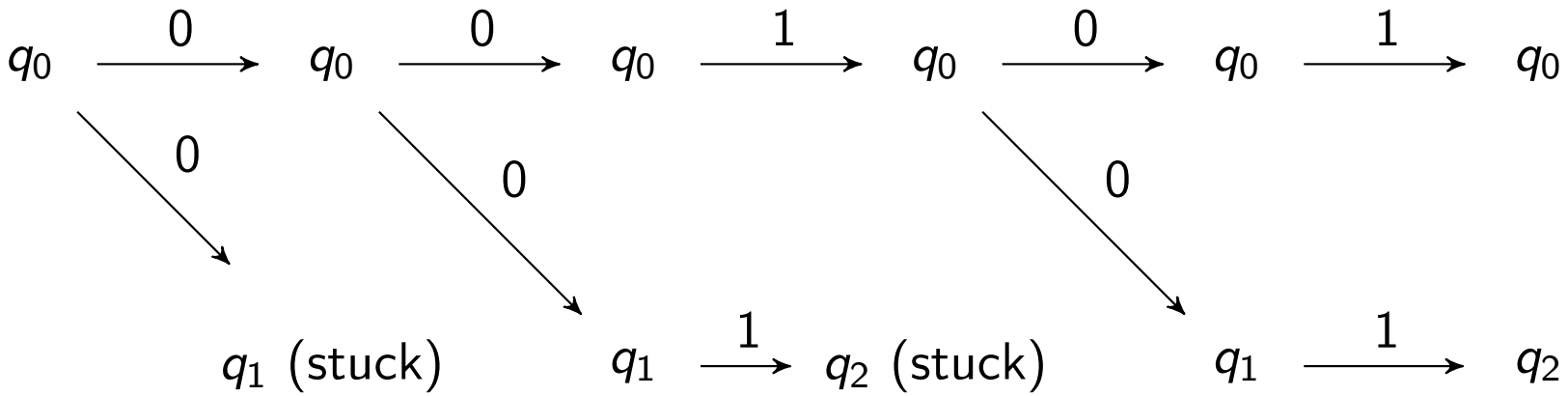
Nedeterministické konečné automaty s ϵ přechody (ϵ-NFA)

Nedeterministický automat může být ve více stavech paralelně. Má schopnost 'uhodnout' něco o vstupu.

NFA přijímající všechna slova končící 01.



NFA zpracovává vstup 00101.



Definition 2.1 (Nedeterministický konečný automat s ϵ přechody (ϵ-NFA))

Nedeterministický konečný automat s ϵ přechody (ϵ-NFA)

$A = (Q, \Sigma, \delta, q_0, F)$ sestává z:

1. konečné množiny **stavů**, zpravidla značíme Q
2. konečné množiny **vstupních symbolů**, značíme Σ
3. **přechodové funkce**, zobrazení $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ vracející podmnožinu Q.
4. **počáteční ho stavu**[a] $q_0 \in Q$,
5. a **množiny koncových (přijímajících) stavů** $F \subseteq Q$.

[a] alternativa: množiny počátečních stavů $S_0 \subseteq Q$

Example 2.1

Tabulka pro ϵ-NFA z předchozího slajdu $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$ je:

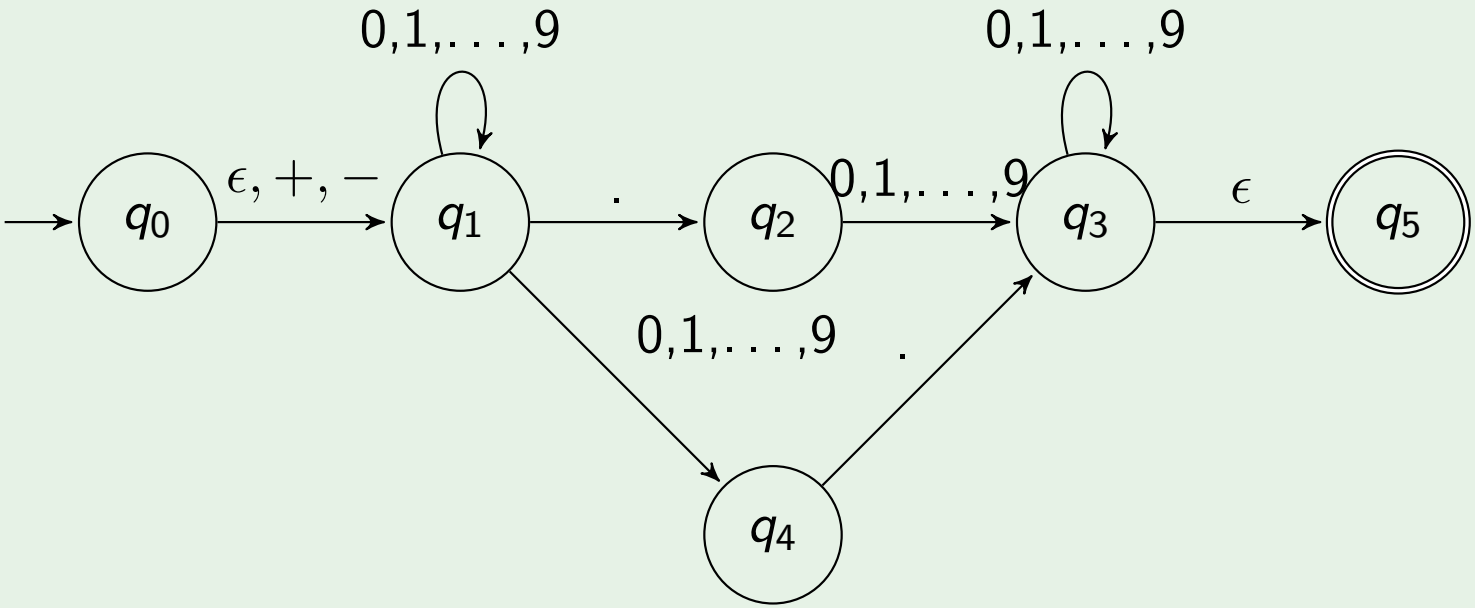
δ	ϵ	0	1
$\to q_0$	$\emptyset$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$	$\emptyset$

Konečné automaty s ϵ přechody

- ϵ-přechod znamená přechod bez přečtení vstupního symbolu.

Example 2.2 (ϵ-NFA, silně nedeterministická δ)

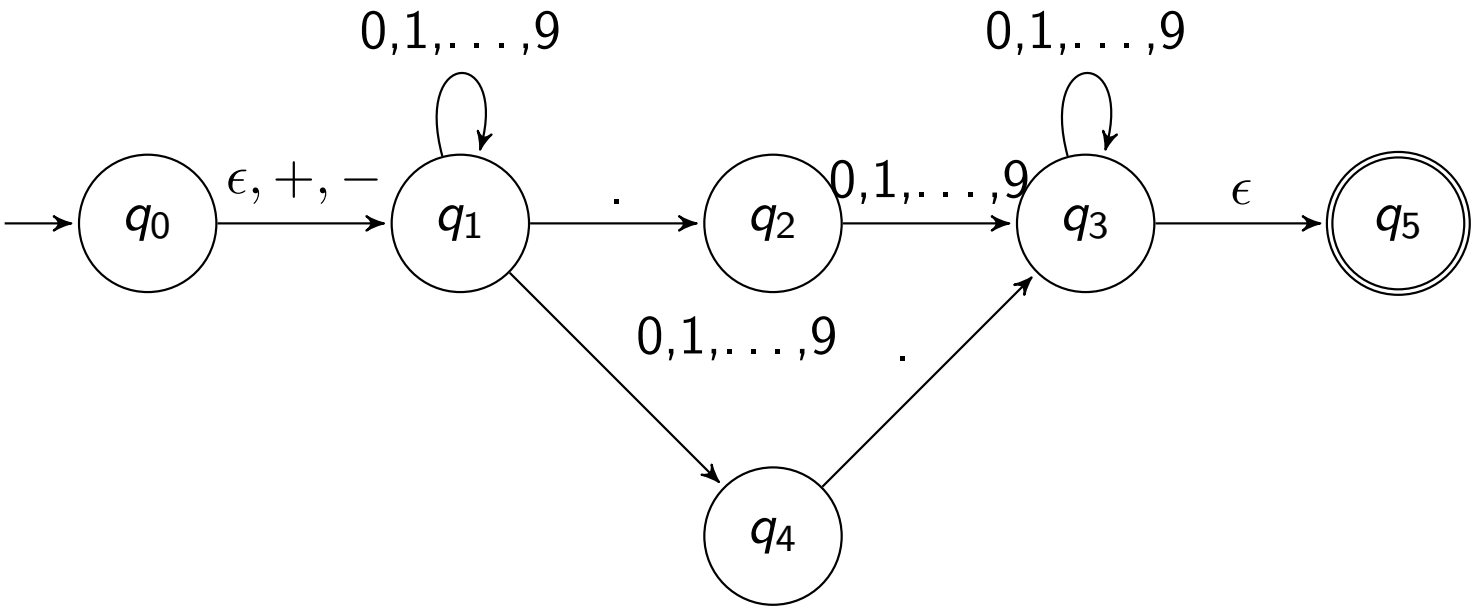
(1) Volitelně znaménko + nebo - ,
(2) řetězec číslic,
(3) desetinná tečka a
(4) další řetězec číslic. Nejméně jeden z řetězců (2) a (4) musí být neprázdný.



Example 2.3 (Přechodová funkce v tabulce)

Předešlý ϵ-NFA je: $E = (\{q_0, q_1, \ldots, q_5\}, \{., +, -, 0, 1, \ldots, 9\}, \delta, q_0, \{q_5\})$ s δ:

δ	ϵ	+,-	.	0,1,...,9
q_0	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$*q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$



ϵ–uzávěr

Definition 2.2 (ϵ–uzávěr)

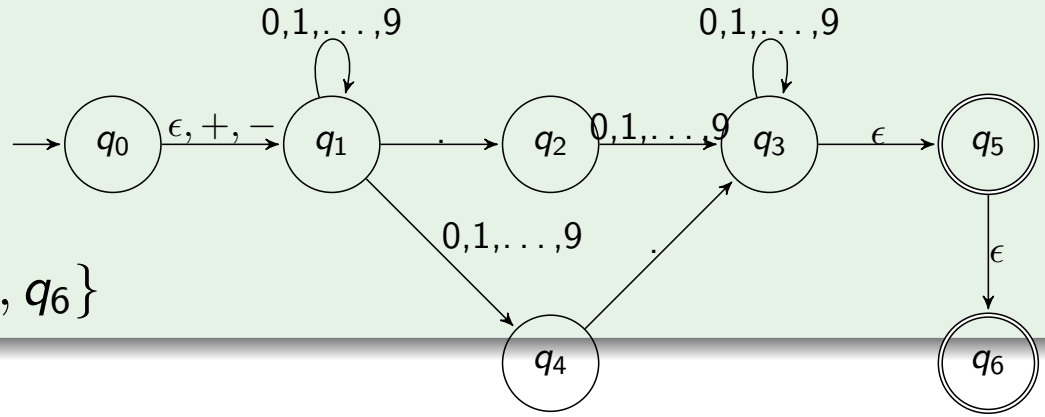
Pro $q \in Q$ definujeme **ϵ–uzávěr** $\epsilon closure(q)$ rekurzivně:

- Stav q je v $\epsilon closure(q)$.
- Je–li $p \in \epsilon closure(q)$ a $r \in \delta(p, \epsilon)$ pak i $r \in \epsilon closure(q)$.

Pro množinu stavů $S \subseteq Q$ definujeme $\epsilon closure(S) = \bigcup_{q \in S} \epsilon closure(q)$.

Example 2.4 (ϵ uzávěr)

- $\epsilon closure(q_0) = \{q_0, q_1\}$
- $\epsilon closure(q_1) = \{q_1\}$
- $\epsilon closure(q_3) = \{q_3, q_5, q_6\}$
- $\epsilon closure(\{q_3, q_4\}) = \{q_3, q_4, q_5, q_6\}$



Rozšířená přechodová funkce a jazyk přijímaný ϵ-NFA

Definition 2.3 (δ^* pro ϵNFA)

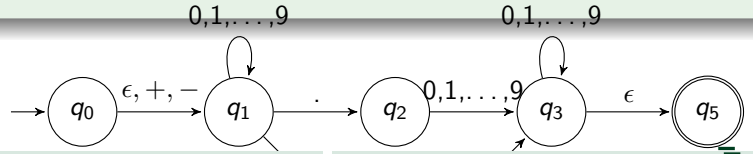
Nechť $E = (Q, \Sigma, \delta, q_0, F)$ je ϵ–NFA. Rozšířenou přechodovou funkci δ^* definujeme následovně:

- $\delta^*(q, \epsilon) = \epsilon closure(q)$.
- Indukční krok: $v = wa$, kde $w \in \Sigma^*, a \in \Sigma$.

$$\delta^*(q, wa) = \epsilon closure\left(\bigcup_{p \in \delta^*(q,w)} \delta(p, a)\right).$$

Example 2.5

$$\begin{array}{rcl}
\delta^*(q_0, \epsilon) = & \epsilon closure(q_0) & = \{q_0, q_1\} \\
\delta^*(q_0, 5) = \epsilon closure(\bigcup_{q \in \delta^*(q_0,\epsilon)} \delta(q, 5)) & = \epsilon closure(\delta(q_0, 5) \cup \delta(q_1, 5)) & = \{q_1, q_4\} \\
\delta^*(q_0, 5.) = & \epsilon closure(\delta(q_1, .) \cup \delta(q_4, .)) & = \{q_2, q_3, q_5, q_6\} \\
\delta^*(q_0, 5.6) = & \epsilon closure(\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6)) & = \{q_3, q_5, q_6\}
\end{array}$$



Jazyk přijímaný ϵNFA

Definition 2.4 (Jazyk přijímaný ϵNFA)

Mějme nedeterministický konečný automat ϵNFA $A = (Q, \Sigma, \delta, q_0, F)$, pak

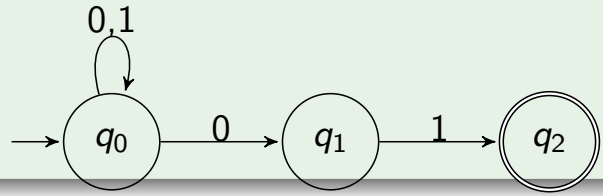
$$L(A) = \{w|\ \delta^*(q_0, w) \cap F \neq \emptyset\}$$

je jazyk přijímaný automatem A.
Tedy $L(A)$ je množina slov $w \in \Sigma^*$ takových, že $\delta^*(q_0, w)$ obsahuje alespoň jeden přijímající stav.

Example 2.6

Automat z předchozího slajdu přijímá jazyk $L = \{w|w \text{ končí na } 01, w \in \{0,1\}^*\}$.
Důkaz indukcí konjunkce tvrzení:

- $\delta^*(q_0, w)$ obsahuje q_0 pro každé slovo w.
- $\delta^*(q_0, w)$ obsahuje q_1 iff w končí 0.
- $\delta^*(q_0, w)$ obsahuje q_2 iff w končí 01.



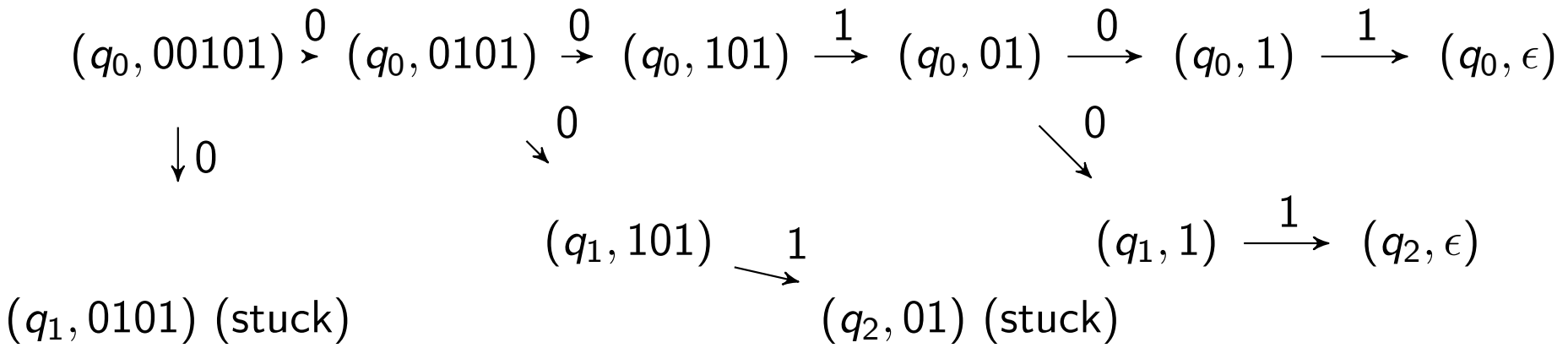
Konfigurace automatu, Výpočetní graf

Definition 2.5 (Konfigurace *DFA*, ϵ*NFA*)

Mějme ϵNFA $A = (Q, \Sigma, \delta, q_0, F)$, $q \in Q$, $v \in \Sigma^*$, pak dvojice (q, v) označuje **konfiguraci** konečného automatu, nacházejícího se ve stavu q s nepřečteným vstupem v.

Definition 2.6 (Výpočetní strom, graf ϵ*NFA*)

Mějme NFA $A = (Q, \Sigma, \delta, q_0, F)$ a vstupní slovo $w \in \Sigma^*$. Uzly **výpočetního grafu** jsou konfigurace A nad slovem w, orientované hrany značí možný přechod mezi konfiguracemi, tj. z (p, au) vede hrana do (q, u) právě když $q \in \delta(p, a)$.



Podmnožinová konstrukce (s ϵ–přechody)

Theorem 2.1 (Podmnožinová konstrukce (s ϵ–přechody))

Jazyk L je rozpoznatelný ϵ-NFA právě když je L regulární.

Algorithm: !Podmnožinová konstrukce (s ϵ–přechody)

Pro libovolný ϵ-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ zkonstruujeme DFA $D = (Q_D, \Sigma, \delta_D, q_{D_0}, F_D)$ přijímající stejný jazyk jako N.

- Nové stavy jsou ϵ-uzavřené podmnožiny Q_N.

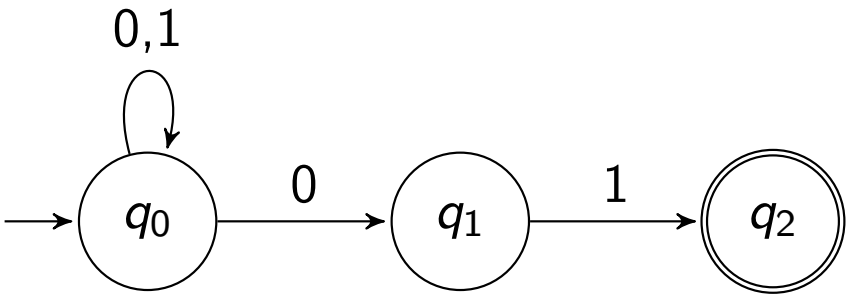
  $Q_D \subseteq \mathcal{P}(Q_N)$, $\forall S \subseteq Q_N : \epsilon closure(S) \in Q_D$. V Q_D může být i $\emptyset$.
- Počáteční stav je ϵ-uzávěr q_0.

  $q_D = \epsilon closure(q_0)$.
- Přijímací stavy jsou všechny množiny obsahující nějaký přijímající stav.

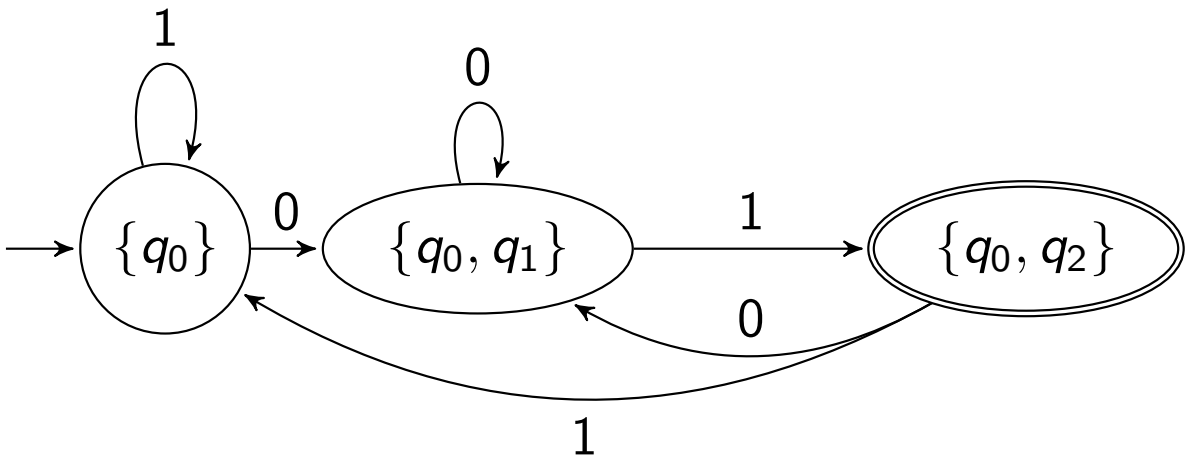
  $F_D = \{S | S \in Q_D \ \& \ S \cap F_N \neq \emptyset\}$.
- Přechodová funkce sjednotí předchody z jednotlivých stavů a uzavře ϵclosure.

  Pro $S \in Q_D, a \in \Sigma$ definujeme $\delta_D(S, a) = \epsilon closure(\bigcup_{p \in S} \delta_N(p, a))$.

Příklad podmnožinové konstrukce pro $\{w.01 | w \in \{0,1\}\}$



	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



Theorem 2.2 (Převod ϵ-NFA na DFA)

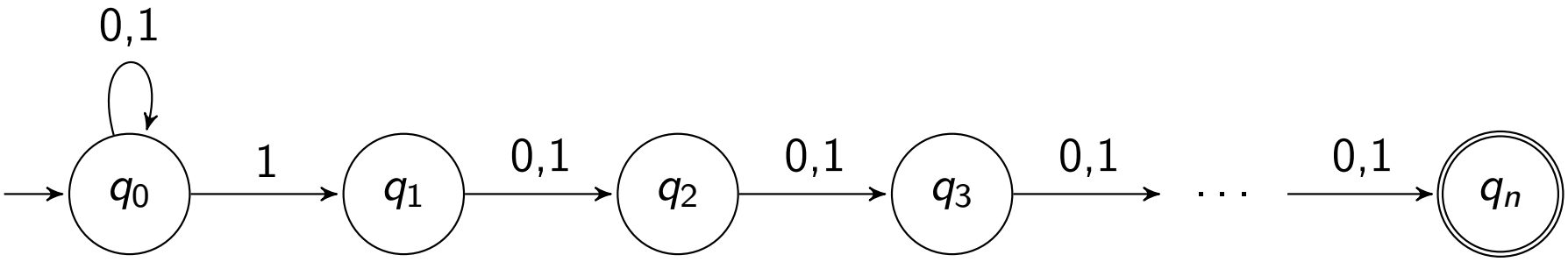
Pro DFA $D = (Q_D, \Sigma, \delta_D, q_{D_0}, F_D)$ *vytvořený podmnožinovou konstrukcí z NFA* $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ *platí* $L(N) = L(D)$.

Proof.

Indukcí dokážeme: $\delta_D^*(q_0, w) = \delta_N^*(q_{D_0}, w)$. □

Example 2.7 ('Těžký' případ pro podmnožinovou konstrukci)

Jazyk $L(N)$ slov 0's a 1's takových, že n–tý symbol od konce je 1. Intuitivně si DFA musí pamatovat n posledních přečtených symbolů.



- Aplikace hledání v textu

Množinové operace nad jazyky

Definition 2.7 (Množinové operace nad jazyky)

Mějme dva jazyky L, M. Definujme následující operace:

(1) binární **sjednocení** $L \cup M = \{w | w \in L \vee w \in M\}$
- Příklad: jazyk obsahuje slova začínající a^i nebo tvaru $b^j c^j$.

(2) **průnik** $L \cap M = \{w | w \in L \ \& \ w \in M\}$
- Příklad: jazyk obsahuje slova sudé délky končící na 'baa'.

(3) **rozdíl** $L - M = \{w | w \in L \ \& \ w \notin M\}$
- Příklad: jazyk obsahuje slova sudé délky nekončící na 'baa'.

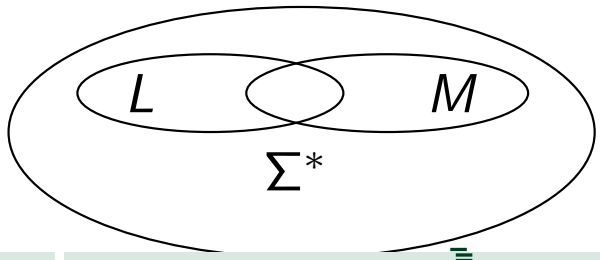
(4) **doplněk (komplement)** $\overline{L} = -L = \{w | w \notin L\} = \Sigma^* - L$
- Příklad: jazyk obsahuje slova nekončící na 'a'.

Theorem (de Morganova pravidla)

Platí:

$$L \cap M = \overline{\overline{L} \cup \overline{M}}$$
$$L \cup M = \overline{\overline{L} \cap \overline{M}}$$
$$L - M = L \cap \overline{M}.$$

Důkaz ze vztahů $\&, \vee, \neg$.



Uzávěrové vlastnosti regulárních jazyků

Theorem 2.3 (Uzavřenost na množinové operace)

Mějme regulární jazyky L, M. Pak jsou následující jazyky také regulární:

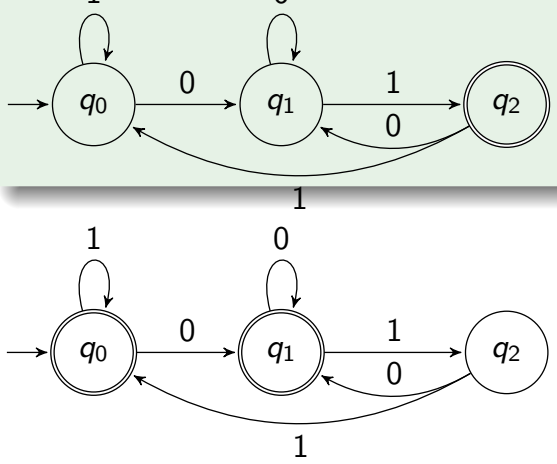
(1) *sjednocení* $L \cup M$,
(2) *průnik* $L \cap M$,
(3) *rozdíl* $L - M$,
(4) *doplněk* $\bar{L} = \Sigma^* - L$.

Proof: Uzavřenost RJ na doplněk

- Pokud δ není pro některé dvojice q, a definovaná, přidáme nový nepřijímající stav q_{fail} a do něj přechod pro vše dříve nedefinované plus $\forall a \in \Sigma$: $\delta(q_{fail}, a) = q_{fail}$.
- Pak stačí prohodit koncové a nekoncové stavy přijímajícího deterministického FA, tj. $F_{complement} = Q_A - F_A$. □

Example 2.8

Jazyk $\{w; w \in \{0,1\}^*01\}$



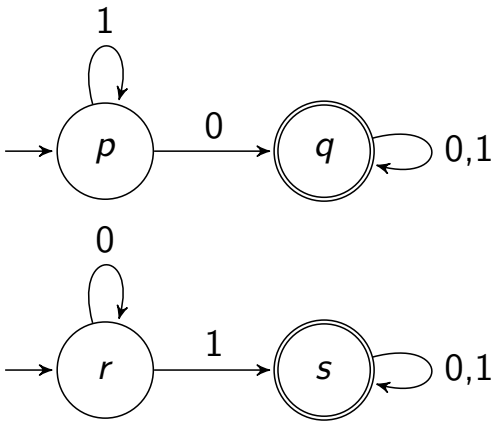
Konstrukce součinu automatů

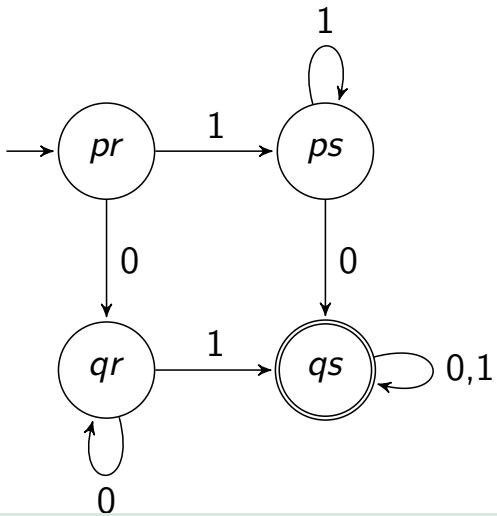
Proof: Průnik, sjednocení, rozdíl

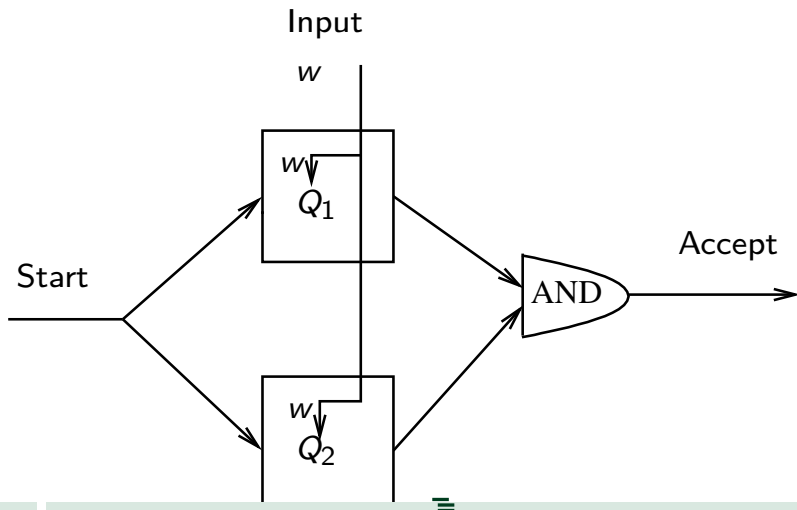
- Pro sjednocení a rozdíl doplníme funkci δ na totální.
- Zkonstruujeme součinový automat,
  $Q = (Q_1 \times Q_2, \Sigma, \delta([p_1, p_2], x) = [\delta_1(p_1, x), \delta_2(p_2, x)], [q_{0_1}, q_{0_2}], F)$
- průnik: $F = F_1 \times F_2$
- sjednocení: $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- rozdíl: $F = F_1 \times (Q_2 - F_2)$.

□

Příklad součinu automatů (průnik jazyků). Slova obsahující 0,1, oboje.







Příklady na uzávěrové vlastnosti

Example 2.9

Konstruujeme konečný automat přijímající slova, která obsahují $3k + 2$ symbolů 1 a neobsahují posloupnost 11.

- Přímá konstrukce je komplikovaná.
- $L_1 = \{w | w \in \{0,1\}^* \& |w|_1 = 3k + 2\}$
- $L_2 = \{w | u, v \in \{0,1\}^* \& w = u11v\}$
- $L = L_1 - L_2$.

Example 2.10

Jazyk M slov s různým počtem 0 a 1 není regulární.

- Je–li M regulární, $\overline{M} = \Sigma^* - M$ je také regulární.
- O $\overline{M}$ víme, že regulární není (pumping lemma).

Ještě jeden příklad

Example 2.11

Jazyk $L_{0\neq 1} = \{0^i 1^j : i \neq j, i, j \in \mathbb{N}_0\}$ není regulární.

- Jazyk $L_{01} = \{0^i 1^j; i, j \in \mathbb{N}_0\}$ je regulární, umíme sestrojit konečný automat.
- $L_{01} - L_{0\neq 1} = \{0^i 1^i : i \in \mathbb{N}_0\}$
- Z uzávěrových vlastností víme, že rozdíl regulárních jazyků je regulární.
- Jazyk L_{01} regulární je.
- Předpokládejme, že $L_{0\neq 1}$ je regulární. Pak by i $\{0^i 1^i : i \in \mathbb{N}_0\}$ musel být regulární, což není - SPOR.

Řetězcové operace nad jazyky

Definition 2.8 (Řetězcové operace nad jazyky)

Nad jazyky L, M definujeme následující operace:

zřetězení jazyků	$L.M = \{uv \mid u \in L \ \& \ v \in M\}$
	$L.x = L.\{x\}$ a $x.L = \{x\}.L$ pro $x \in \Sigma$
mocniny jazyka	$L^0 = \{\epsilon\}$
	$L^{i+1} = L^i.L$
pozitivní iterace	$L^+ = L^1 \cup L^2 \ldots = \bigcup_{i \geq 1} L^i$
obecná iterace	$L^* = L^0 \cup L^1 \cup L^2 \ldots = \bigcup_{i \geq 0} L^i$
	tedy $L^* = L^+ \cup \{\epsilon\}$
otočení jazyka	$L^R = \{u^R \mid u \in L\}$
(=zrcadlový obraz,reverze)	$(x_1 x_2 \ldots x_n)^R = x_n x \ldots x_2 x_1$
levý kvocient L podle M	$M \setminus L = \{v \mid uv \in L \ \& \ u \in M\}$
levá derivace L podle w	$\partial_w L = \{w\} \setminus L$ (pozn. derivace bude i v jiném významu
pravý kvocient L podle M	$L/M = \{u \mid uv \in L \ \& \ v \in M\}$
pravá derivace L podle w	$\partial_w^R L = L/\{w\}.$

Theorem 2.4 (Uzavřenost reg. jazyků na řetězcové operace)

Jsou–li L, M regulární jazyky, je regulární i $L.M$, L^*, L^+, L^R, $M \setminus L$ a L/M.

Lemma ($L.M$)

Jsou–li L, M regulární jazyky, je regulární i $L.M$.

Proof:

Vezmeme DFA $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, pak $A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ tak že $L = L(A_1)$ a $M = L(A_2)$.

Definujeme nedeterministický automat $B = (Q \cup \{q_0\}, \Sigma, \delta, q_0, F_2)$ kde:

$Q = Q_1 \cup Q_2$ předpokládáme různá jména stavů, jinak přejmenujeme

končíme až po přečtení slova z L_2

$\delta(q_0, \epsilon)$	$= \{q_1, q_2\}$	pro $q_1 \in F_1$	tj. $\epsilon \in L(A_1)$
$\delta(q_0, \epsilon)$	$= \{q_1\}$	pro $q_1 \notin F_1$	tj. $\epsilon \notin L(A_1)$
$\delta(q_0, x)$	$= \emptyset$	pro $x \in \Sigma$	
$\delta(q, x)$	$= \{\delta_1(q, x)\}$	pro $q \in Q_1$ & $\delta_1(q, x) \notin F_1$	počítáme v A_1
	$= \{\delta_1(q, x), q_2\}$	pro $q \in Q_1$ & $\delta_1(q, x) \in F_1$	nedet. přechod z A_1
	$= \{\delta_2(q, x)\}$	pro $q \in Q_2$	počítáme v A_2.

Pak $L(B) = L(A_1).L(A_2)$. □

Uzavřenost iterace

Lemma (L^*, L^+)

Je–li L regulární jazyk, je regulární i L^*, L^+.

- Idea: opakovaný výpočet automatu $A = (Q, \Sigma, \delta, q_0, F)$
- realizace: nedeterministické rozhodnutí, zda pokračovat nebo restart
- speciální stav pro příjem $\epsilon \in L^0$ (pro L^+ vynecháme či $\notin F$).

Proof: Důkaz pro L^*

Vezmeme DFA $A = (Q, \Sigma, \delta, q_0, F)$, tak že $L = L(A)$.
Definujeme NFA automat $B = (Q \cup \{q_B\}, \Sigma, \delta_B, q_B, F \cup \{q_B\})$ kde:

$\delta_B(q_B, \epsilon) = \{q_0\}$ nový stav q_B pro příjem ϵ, přejdeme do q_0
$\delta_B(q_B, x) = \emptyset$ pro $x \in \Sigma$
$\delta_B(q, x) = \{\delta(q, x)\}$ pokud $q \in Q$ & $\delta(q, x) \notin F$ uvnitř A
$= \{\delta(q, x), q_0\}$ pokud $q \in Q$ & $\delta(q, x) \in F$ možný restart

Pak $L(B) = L(A)^*$ ($q_B \in F_B$), $L(B) = L(A)^+$ ($q_B \notin F_B$). □

Uzavřenost reverze

Lemma (L^R)

Je–li L regulární jazyk, je regulární i L^R.

- Zřejmě $(L^R)^R = L$ a tedy stačí ukázat jeden směr.
- idea: obrátíme šipky ve stavovém diagramu; nederministický FA

Proof:

Vezmeme DFA $A = (Q, \Sigma, \delta, q_0, F)$, tak že $L = L(A)$.
Definujeme nedeterministický automat $B = (Q \cup \{q_B\}, \Sigma, \delta_B, q_B, \{q_0\})$ kde:

- $\delta_B(q, x) = \{p | \delta(p, x) = q\}$ pro $q \in Q$
- $\delta_B(q_B, \epsilon) = F$, $\delta_B(q_B, x) = \emptyset$.
- Pro libovolné slovo $w = x_1 x_2 \dots x_n$
  - $q_0, q_1, q_2, \dots, q_n$ je přijímající výpočet pro w v A

  $\Leftrightarrow$

  - $q_B, q_n, q_{n-1}, \dots, , q_2, q_1, q_0$ je přijímající výpočet pro w^R v B. □

Pozn. Někdy L nebo L^R má výrazně jednodušší přijímající automat

Uzavřenost kvocientu

Lemma ($M \setminus L$ a L/M)

Jsou–li L, M regulární jazyky, je regulární i $M \setminus L$ a L/M.

- Idea: A_L, budeme startovat ve stavech, do kterých se lze dostat slovem z M.

Proof:

- Vezmeme DFA $A = (Q, \Sigma, \delta, q_0, F)$, tak že $L = L(A)$.
  Definujeme nedeterministický NFA $B = (Q \cup \{q_\epsilon\}, \Sigma, \delta, q_\epsilon, F)$ kde:
- a přidáme přechod $\delta(q_\epsilon, \epsilon) = \{q | q \in Q \ \& \ (\exists u \in M)\ q = \delta^*(q_0, u)\}$ do stavů, kam lze dojit slovem z M.
  - lze nalézt algoritmicky
    $(\{q;\ L(A_q) \cap M \neq \emptyset$ kde $A_q = (Q, \Sigma, \delta, q_0, \{q\})\})$
- $v \in M \setminus L$
  - $\Leftrightarrow$ $(\exists u \in M)\ uv \in L$
  - $\Leftrightarrow$ $(\exists u \in M, \exists q \in Q)\ \delta(q_0, u) = q \ \& \ \delta(q, v) \in F$
  - $\Leftrightarrow$ $\exists q \in \delta(q_\epsilon, \epsilon) \ \& \ \delta(q, v) \in F$
  - $\Leftrightarrow$ $v \in L(B)$.

$L/M = (M^R \setminus L^R)^R$. □

Regulární výrazy

Regulární výrazy (RV) jsou

- algebraickým popisem jazyků
- deklarativním způsobem, jak vyjádřit slova, která chceme přijímat.
- Schopné definovat všechny a pouze regulární jazyky.
- Můžeme je brát jako programovací jazyk, uživatelsky přívětivý popis konečného automatu.

Example 3.1

- Základní UNIX grep příkaz.
- Lexikální analyzátory jako Lex a Flex (popis pomocí 'token'ů je vzásadě regulární výraz).
- Python knihovna re .

- Syntaktická analýza potřebuje silnější nástroj, bezkontextové gramatiky, budou následovat.

Definition 3.1 (Regulární výrazy (Regular Expression) (RegE), hodnota RegE $L(\alpha)$)

Regulární výrazy $\alpha, \beta \in RegE(\Sigma)$ nad konečnou neprázdnou abecedou $\Sigma = \{a_1, a_2, \ldots, a_n\}$ a jejich hodnota $L(\alpha)$ jsou definovány induktivně:

- Základ:

výraz α	pro	hodnota $L(\alpha) \equiv [\alpha]$
ϵ	prázdný řetězec	$L(\epsilon) = \{\epsilon\}$
$\emptyset$	prázdný výraz	$L(\emptyset) = \{\} \equiv \emptyset$
$\mathbf{a}$	$a \in \Sigma$	$L(\mathbf{a}) = \{a\}$.

- Indukce:

výraz	hodnota	poznámka
$\alpha + \beta$	$L(\alpha + \beta) = L(\alpha) \cup L(\beta)$	v grep, re |
$\alpha\beta$	$L(\alpha\beta) = L(\alpha)L(\beta)$	můžeme značit ., ale plete se s UNIX grep.
α^*	$L(\alpha^*) = L(\alpha)^*$	
(α)	$L((\alpha)) = L(\alpha)$	závorky nemění hodnotu.

Každý regulární výraz dostaneme indukcí výše, tj. třída $RegE(\Sigma)$ je nejmenší třída uzavřená na uvedené operace.

Lemma 3.1

Jazyk $L(\epsilon) = \{\epsilon\} = \emptyset^*$, v definici jen pro význam symbolu $L(\epsilon)$.

Příklady reguláních výrazů, priorita

Definition 3.2 (priorita)

Nejvyšší prioritu má iterace $*$, nižší konkatenace (zřetězení), nejnižší sjednocení $+$.

Example 3.2 (Regulární výrazy)

Jazyk střídajících se nul a jedniček lze zapsat:

- $(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{1}(\mathbf{01})^* + \mathbf{0}(\mathbf{10})^*$
- $(\epsilon + \mathbf{1})(\mathbf{01})^*(\epsilon + \mathbf{0})$.

Jazyk $L((\mathbf{0}^*\mathbf{10}^*\mathbf{10}^*\mathbf{1})^*\mathbf{0}^*) = \{w | w \in \{0,1\}^*, |w|_1 = 3k, k \geq 0\}$.

Theorem 3.1 (!varianta Kleeneho věty)

1. *Každý jazyk reprezentovaný konečným automatem lze zapsat jako regulární výraz.*
2. *Každý jazyk popsaný regulárním výrazem můžeme zapsat jako ϵ-NFA (a tedy i DFA).*

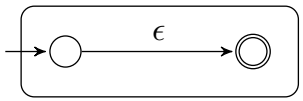
Převod RegE výrazu na ϵ–NFA automat

Převod RegE výrazu na ϵ–NFA automat.

Důkaz indukcí dle struktury R. V každém kroku zkonstruujeme ϵ-NFA E rozpoznávající stejný jazyk $L(R) = L(E)$ se třemi dalšími vlastnostmi:

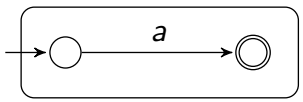
1. Právě jeden přijímající stav.
2. Žádné hrany do počátečního stavu.
3. Žádné hrany z koncového stavu.

Základ:



Prázdný řetězec ϵ

Prázdná množina $\emptyset$

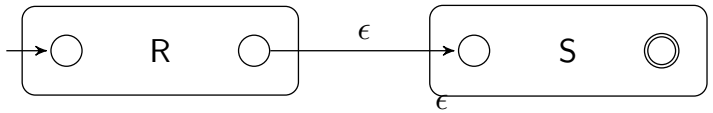


$a \in \Sigma$: výraz **a**

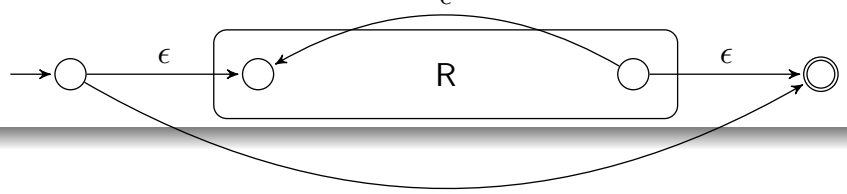
Součet $R + S$:

INDUKCE:

Zřetězení RS:



Iterace R^*:



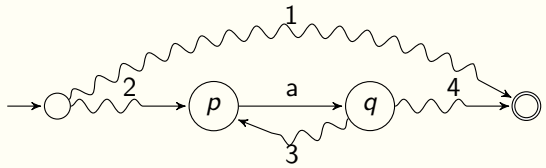
□

Alternativní (neefektivní) důkaz Kleeneovy věty

Proof: Rozpoznatelný FA ⇒ RJ

Máme automat DFA $A = (Q, \Sigma, \delta, q_0, F)$ který přijímá jazyk $L(A)$. Indukcí podle počtu hran A dokážeme $L(A) \in RJ(\Sigma)$.

- žádná hrana – pouze jazyky $\emptyset$ a $\{\epsilon\}$, z definice a $\emptyset^*$.
- $(n+1)$ hran
  - vybereme jednu hranu $p \rightarrow^a q$, tj. $q \in \delta(p, a)$
  - sestrojíme čtyři automaty bez tého hrany ($\delta^| = \delta$, jen $\delta^|(p, a) = \delta(p, a) - \{q\}$)
    - $A_1 = (Q, \Sigma, \delta^|, q_0, F)$
    - $A_2 = (Q, \Sigma, \delta^|, q_0, \{p\})$
    - $A_3 = (Q, \Sigma, \delta^|, q, \{p\})$
    - $A_4 = (Q, \Sigma, \delta^|, q, F)$



  - Potom $L(A) = L(A_1) \cup (L(A_2).a).(L(A_3).a)^* L(A_4)$,
  - jazyky $L(A_1), L(A_2), L(A_3), L(A_4) \in RJ(\Sigma)$ z indukčního předpokladu (n hran).

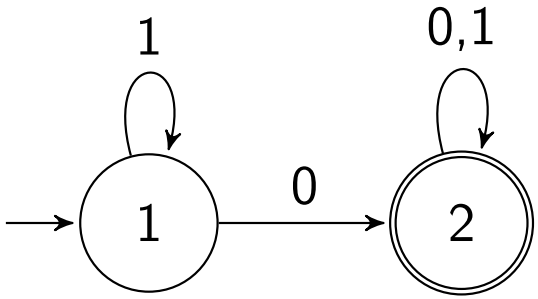
□

Od DFA k regulárním výrazům

Regulární výraz z DFA

- Mějme DFA A, $Q_A = \{1, \ldots, n\}$ o n stavech, stavy očíslujeme od 1.
- Nechť $R_{ij}^{(k)}$ je regulární výraz, $L(R_{ij}^{(k)}) = \{w | \delta^*_{\leq k}(i, w) = j\}$ množina slov převádějících stav i do stavu j v A cestou, která neobsahuje stav s vyšším indexem než k.
- Budeme rekruzivně konstruovat $R_{ij}^{(k)}$ pro $k = 0, \ldots, n$.
- $k = 0$, $i \neq j$: $R_{ij}^{(0)} = \mathbf{a_1} + \mathbf{a_2} + \ldots + \mathbf{a_m}$ kde $a_1, a_2, \ldots, a_m$ jsou symboly označující hrany i do j (nebo $R_{ij}^{(0)} = \emptyset$ nebo $R_{ij}^{(0)} = \mathbf{a}$ pro $m = 0, 1$).
- $k = 0$, $i = j$: smyčky, $R_{ii}^{(0)} = \epsilon + \mathbf{a_1} + \mathbf{a_2} + \ldots + \mathbf{a_m}$ kde $a_1, a_2, \ldots, a_m$ jsou symboly na smyčkách v i.

Příklad: Od konečného automatu k RegE



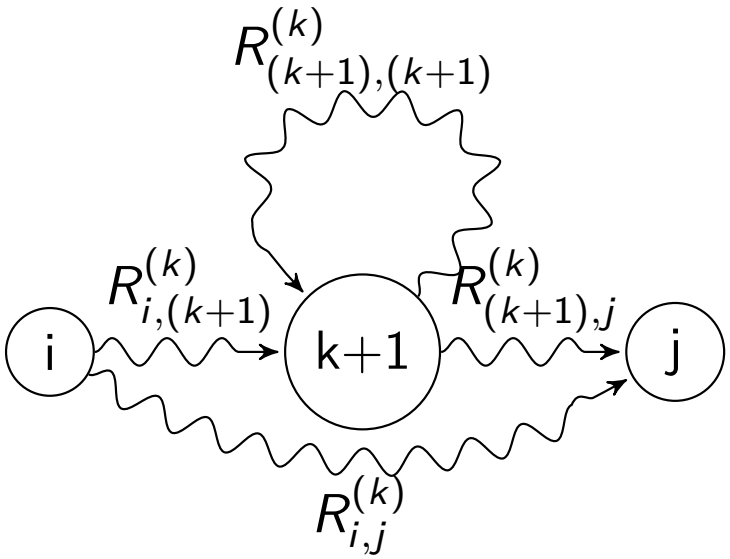
$$R_{12}^{(2)} = \mathbf{1}^*\mathbf{0}(\mathbf{0}+\mathbf{1})^*$$

$$R_{ij}^{(k+1)} = R_{ij}^{(k)} + R_{i(k+1)}^{(k)}(R_{(k+1)(k+1)}^{(k)})^* R_{(k+1)j}^{(k)}$$

$R_{11}^{(0)}$	$\epsilon + \mathbf{1}$	$=$
$R_{12}^{(0)}$	$\mathbf{0}$	$=$
$R_{21}^{(0)}$	$\emptyset$	$=$
$R_{22}^{(0)}$	$(\epsilon + \mathbf{0} + \mathbf{1})$	$=$
$R_{11}^{(1)}$	$(\epsilon + \mathbf{1}) + (\epsilon + \mathbf{1})(\epsilon + \mathbf{1})^*(\epsilon + \mathbf{1})$	$=\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0} + (\epsilon + \mathbf{1})(\epsilon + \mathbf{1})^*\mathbf{0}$	$=\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\emptyset + \emptyset(\epsilon + \mathbf{1})^*(\epsilon + \mathbf{1})$	$=\emptyset$
$R_{22}^{(1)}$	$(\epsilon + \mathbf{0} + \mathbf{1}) + \emptyset(\epsilon + \mathbf{1})^*\mathbf{0}$	$=\epsilon + \mathbf{0} + \mathbf{1}$
$R_{11}^{(2)}$	$\mathbf{1}^* + \mathbf{1}^*\mathbf{0}(\epsilon + \mathbf{0} + \mathbf{1})^*\emptyset$	$=\mathbf{1}^*$
$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0} + \mathbf{1}^*\mathbf{0}(\epsilon + \mathbf{0} + \mathbf{1})^*(\epsilon + \mathbf{0} + \mathbf{1})$	$=\mathbf{1}^*\mathbf{0}(\mathbf{0}+\mathbf{1})^*$
$R_{21}^{(2)}$	$\emptyset + (\epsilon + \mathbf{0} + \mathbf{1})(\epsilon + \mathbf{0} + \mathbf{1})^*\emptyset$	$=\emptyset$
$R_{22}^{(2)}$	$(\epsilon + \mathbf{0} + \mathbf{1}) + (\epsilon + \mathbf{0} + \mathbf{1})(\epsilon + \mathbf{0} + \mathbf{1})^*(\epsilon + \mathbf{0} + \mathbf{1})$	$=(\mathbf{0}+\mathbf{1})^*$

Proof.

INDUKCE. Mějme $\forall i, j \in Q\ R_{ij}^{(k)}$. Konstruujeme $R_{ij}^{(k+1)}$.



1. $R_{ij}^{(k+1)} = R_{ij}^{(k)} + R_{i(k+1)}^{(k)}(R_{(k+1)(k+1)}^{(k)})^* R_{(k+1)j}^{(k)}$
   - Cesty z i do j neprocházející uzlem $(k+1)$ jsou již v $R_{ij}^{(k)}$.
   - Cesty z i do j přes $(k+1)$ s případnými smyčkami můžeme zapsat $R_{i(k+1)}^{(k)}(R_{(k+1)(k+1)}^{(k)})^* R_{(k+1)j}^{(k)}$.
   - regulární výrazy jsou uzavřené na sčítání (sjednocení), zřetězení i iteraci, tj. $R_{ij}^{k+1} \in RegE(\Sigma)$
2. Nakonec, $RegE = \oplus_{j \in F_A} R_{1j}^{(n)}$ sjedncení přes přijímající stavy j.

Zjednodušení regulárních výrazů (netřeba znát)

Lemma (Další vlastnosti bez důkazu)

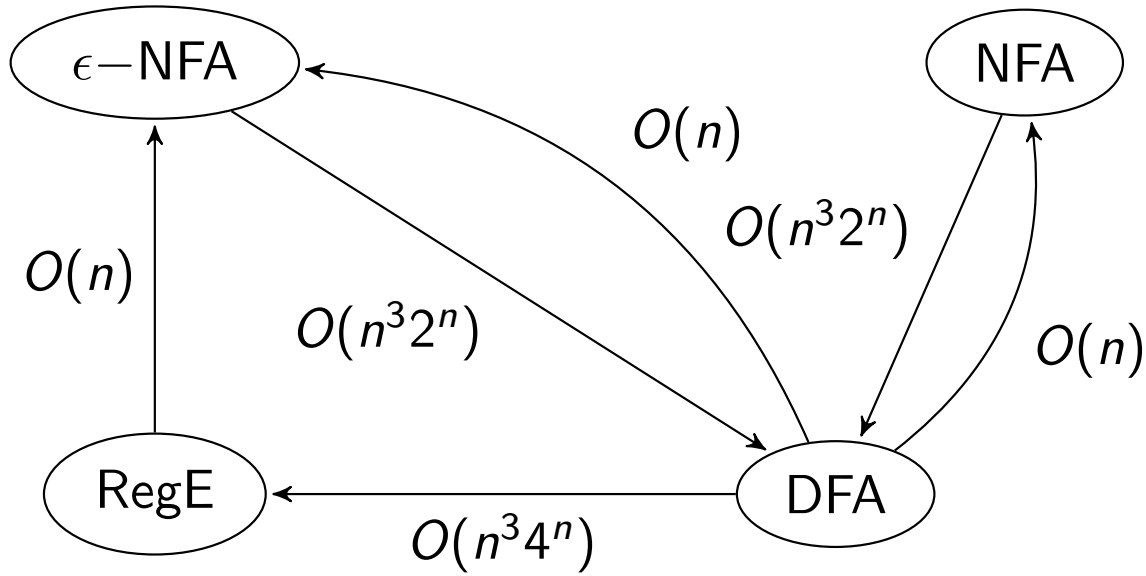
- Zjednodušení návrhu automatů

$$
\begin{aligned}
L.\emptyset = \emptyset.L &= \emptyset \\
\{\epsilon\}.L = L.\{\epsilon\} &= L \\
(L^*)^* &= L^* \\
(L_1 \cup L_2)^* &= L_1^*(L_2.L_1^*)^* = L_2^*(L_1.L_2^*)^* \\
(L_1.L_2)^R &= L_2^R.L_1^R \\
\partial_w(L_1 \cup L_2) &= \partial_w(L_1) \cup \partial_w(L_2) \\
\partial_w(\Sigma^* - L) &= \Sigma^* - \partial_w L.
\end{aligned}
$$

Shrnutí převodů mezi reprezentacemi regulárních jazyků

Převod NFA na DFA

- ϵ uzávěr v $O(n^3)$ – prohledává n stavů násobeno n^2 hran pro ϵ přechody.
- Podmnožinová konstrukce, DFA s až 2^n stavy. Pro každý stav, $O(n^3)$ času na výpočet přechodové funkce.



Převod DFA na NFA

- Přidat množinové závorky k přechodové funkci a přechody pro ϵ u $\epsilon-$NFA.

Převod automatu DFA an RegE regulární výraz

- $O(n^3 4^n)$

RegE výraz na automat

- V čase $O(n)$ vytvoříme $\epsilon-$NFA.

Substituce jazyků

Definition 3.3 (Substituce jazyků)

Mějme konečnou abecedu Σ. Pro každé $x \in \Sigma$ budiž $\sigma(x)$ jazyk v nějaké abecedě Y_x. Dále položme

$$\sigma(\epsilon) = \{\epsilon\}$$
$$\sigma(u.v) = \sigma(u).\sigma(v)$$

- Zobrazení $\sigma : \Sigma^* \to P(Y^*)$, kde $Y = \bigcup_{x \in \Sigma} Y_x$ se nazývá **substituce**.
- Pro jazyk L definujeme: $\sigma(L) = \bigcup_{w \in L} \sigma(w)$, podobně sjednocení.
- **nevypouštějící substituce** je substituce, kde žádné $\sigma(x)$ neobstahuje ϵ.

Example 3.3 (substituce)

1) $\Sigma = \{k, p, m, c, t\}$, $L = (kmp)(ckmp)^* t$,
   k slovník křestních jmen, p slovník příjmení, m mezera, c čárka, t tečka.
2) Pokud $\sigma(0) = \{a^i b^j, i, j \geq 0\}, \sigma(1) = \{cd\}$
   tak $\sigma(010) = \{a^i b^j cda^k b^l, i, j, k, l \geq 0\}$.

Homomorfizmus a inverzní homomorfizmus jazyků

Definition 3.4 (homomorfizmus (jazyků), inverzní homomorfizmus)

Homomorfizmus h je speciální případ substituce, kde obraz je vždy jen jednoslovný jazyk (vynecháváme u něj závorky), tj. $(\forall x \in \Sigma)\ h(x) = w_x$.
Pokud $\forall x : w_x \neq \epsilon$, jde o **nevypouštějící homomorfizmus**.
Inverzní homomorfizmus $h^{-1}(L) = \{w | h(w) \in L\}$.

Example 3.4 (homomorfizmus)

- Znaky nahradíme TEXzápisem, $h(\mu) = \backslash mu$ a podobně.
- Homomorfizmus h definujeme: $h(0) = ab$, a $h(1) = \epsilon$. Pak $h(0011) = abab$. Pro $L = \mathbf{10}^*\mathbf{1}$ je $h(L) = (ab)^*$.

Theorem 3.2 (uzavřenost na homomorfizmus)

Je–li jazyk L i $\forall x \in \Sigma$ jazyk $\sigma(x), h(x)$ regulární, pak je regulární i $\sigma(L), h(L)$.

Uzavřenost na substituci, homomorfizmus.

Strukturální indukcí 'probubláváním' algebraickým popisem jazyka základních, sjednocení, zřetězení a iterace. Pro sjednocení a zřetězení z definice substituce a uzavřenosti regulárních jazyků na sjednocení a zřetězení.

$$\begin{aligned} \underline{\sigma}(\alpha+\beta) &= \sigma(L(\alpha)) \cup \sigma(L(\beta)) \\ \underline{\sigma}(\alpha\beta) &= \{w | \exists u \in L(\alpha) \exists v \in L(\beta) : \sigma(u)\sigma(v) = w\} \end{aligned}$$

Pro iteraci rozložíme na nekonečné sjednocení, pro každý konkrétní počet iterací σ aplikované na konečné zřetězení.

$$\begin{aligned} \sigma(L(\alpha)^*) &= \sigma(L(\alpha)^0) \cup \sigma(L(\alpha)^1) \cup \ldots \cup \sigma(L(\alpha)^n) \cup \ldots \\ &= \underline{\sigma}(\alpha)^0 \cup \underline{\sigma}(\alpha)^1 \cup \ldots \cup \underline{\sigma}(\alpha)^n \cup \ldots \\ &= L(\underline{\sigma}(\alpha)^*). \end{aligned}$$

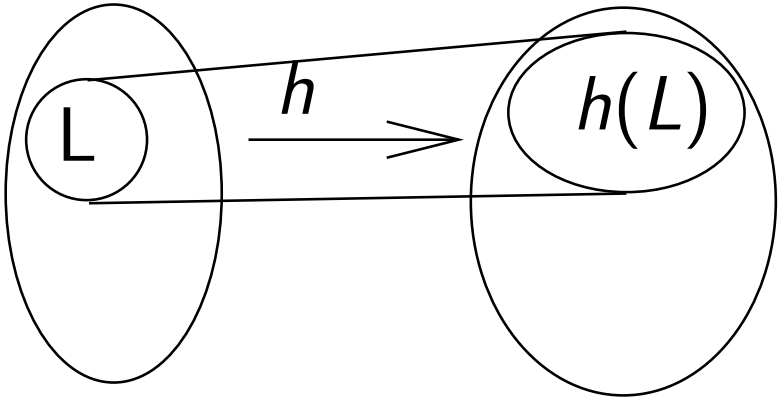
□

Inverzní homomorfizmus

Definition ((3.4) Inverzní homomorfizmus)

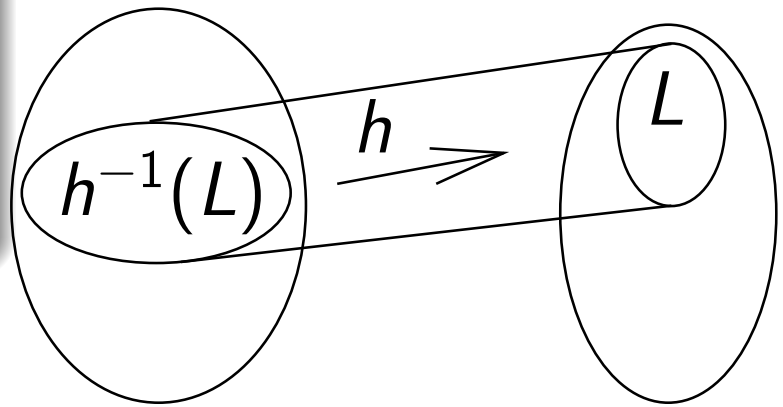
Nechť h je homomorfizmus abecedy T do slov nad abecedou Σ. Pak $h^{-1}(L)$ *'h inverze L'* je množina řetězců
$h^{-1}(L) = \{w | w \in T^*;\ h(w) \in L\}$.

Homomorfizmus aplikovaný dopředně a zpětně.



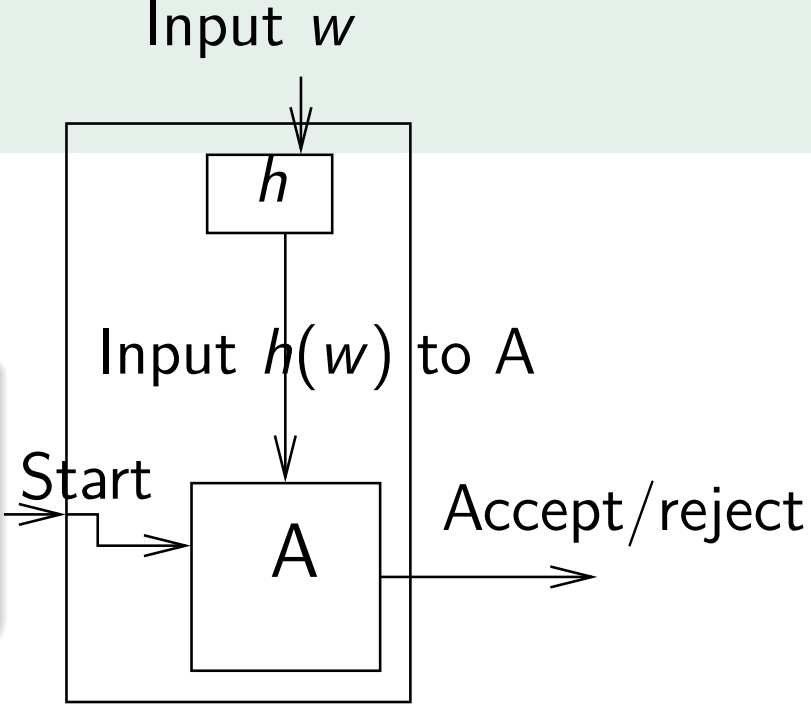
Example 3.5

Nechť $L = (\{00\} \cup \{1\})^*$, $h(a) = 01$ a $h(b) = 10$.
Pak $h^{-1}(L) = (\{ba\})^*$.



Důkaz: $h((\{ba\})^*) \in L$ snadno.
Ostatní w generují izolované 0 (rozbor případů).

Inverzní homomorfizmus DFA



Theorem 3.3

Je–li h homomorfizmus abecedy T do abecedy Σ a L je regulární jazyk abecedy Σ, pak $h^{-1}(L)$ je také regulární jazyk.

Proof:

- pro L máme DFA $A = (Q, \Sigma, \delta, q_0, F)$
- $h : T \to \Sigma^*$
- definujeme ϵ-NFA $B = (Q', T, \delta', [q_0, \epsilon], F \times \{\epsilon\})$ kde

$$Q' = \{[q, u] \,|\, q \in Q, u \in \Sigma^*, \exists(a \in T)\exists(v \in \Sigma^*)h(a) = vu\} \quad u \text{ je buffer}$$
$$\delta'([q, \epsilon], a) = [q, h(a)] \quad \text{naplňuje buffer}$$
$$\delta'([q, bv], \epsilon) = [p, v] \text{ kde } \delta(q, b) = p \quad \text{čte buffer.}$$

Příklad: Navštiv všechny stavy

Example 3.6

Nechť $A = (Q, \Sigma, \delta, q_0, F)$ je DFA. Definujme jazyk $L = \{w \in \Sigma^*;\ \delta^*(q_0, w) \in F$ a pro každý stav $q \in Q$ existuje prefix x_q slova w tak, že $\delta^*(q_0, x_q) = q\}$.
Tento jazyk L je regulání.

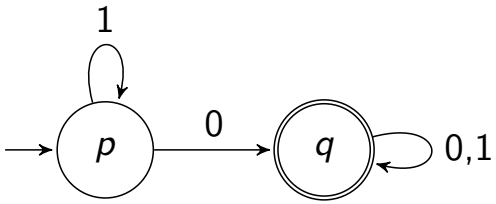
M Označme $M = L(A)$.

T Definujeme novou abecedu T trojic $\{[paq]; p, q \in Q, a \in \Sigma, \delta(p, a) = q\}$.

h Definujeme homomorfizmus $(\forall\ p, q, a)\ h([paq]) = a$.

L_1 Jazyk $L_1 = h^{-1}(M)$ je regulární, protože M je regulární (DFA inverzní homomorfizmus).

- $h^{-1}(101)$ obsahuje $2^3 = 8$ řetězců, např. $[p1p][q0q][p1p] \in \{[p1p], [q1q]\}\{[p0q], [q0q]\}\{[p1p], [q1q]\}$.
- Dále zkonstruujeme L z L_1 (další slide).



L_2 Vynutíme začátek q_0. Definujeme $E_1 = \bigcup_{a\in\Sigma, q\in Q}\{[q_0 a q]\} =$ $E_1 = \{[q_0 a_1 q_0], [q_0 a_2 q_1], \ldots, [q_0 a_m q_n]\}$. Pak $L_2 = L_1 \cap L(E_1.T^*)$.

L_3 Vynutíme stejné sousedící stavy. Definujeme ne–odpovídající dvojice $E_2 = \bigcup_{q\neq r, p,q,r,s\in Q, a,b\in\Sigma}\{[paq][rbs]\}$. Definujeme $L_3 = L_2 - L(T^*.E_2.T^*)$,

- Končí v přijímajícím stavu, protože jsme začali z jazyku M přijímaném DFA A.

L_4 Všechny stavy. $\forall\ q \in Q$ definujeme E_q jako regulární výraz sjednocení všech symbolů T takových, že q není ani na první, ani na poslední pozici. Odečteme $L(E_q^*)$ od L_3. $L_4 = L_3 - \bigcup_{q\in Q}\{E_q^*\}$.

L Odstraníme stavy, necháme symboly. $L = h(L_4)$. Tedy L je regulární.

Přehled:

M $= L(A)$
Inverzní homom.

L_1 $h^{-1}(M) \subseteq \{[qap]\}^*$
průnik RJ

L_2 $+\ q_0$
rozdíl RJ

L_3 + sousední stavy rovny
rozdíl RJ

L_4 + všechny stavy
homomorfismus

L $h([qap]) = a$

Rozhodovací problémy pro regulární jazyky!

Lemma (Test ne–prázdnosti regulárního jazyka)

Lze algoritmicky rozhodnout, zda jazyk přijímaný DFA, NFA, ϵ-NFA je prázdný.

Jazyk je prázdný právě když žádný z koncových stavů není dosažitelný. Dosažitelnost lze testovat $O(|Q|^2)$.

Lemma (Test náležení do regulárního jazyka)

Pro daný řetězec w; $|w| = n$ a regulární jazyk L. Lze algoritmicky rozhodnout, zda je $w \in L$.

- DFA: Spusť automat; pokud $|w| = n$, při dobré reprezentaci a konstatním čase přechodu $O(n)$.
- NFA o s stavech: čas $O(ns^2)$. Každý vstupní symbol aplikujeme na všechny stavy předchozího kroku, kterých je nejvýš s.
- $\epsilon-$NFA - nejdříve určíme ϵ–uzávěr. Pak aplikujeme přechodovou funkci a ϵ–uzávěr na výsledek.

Shrnutí minulé přednášky (+dva řádky a sloupce navíc)

- Podmnožinová konstrukce DFA z ϵNFA
- Regulární výrazy
- Kleeneho věta
  - Jazyk je přijímaný konečným automatem právě když lze napsat jako regulární výraz,
  - tj. z $\emptyset$ a $\{\mathbf{a}\}$ pro $a \in \Sigma$
  - a konečného počtu aplikací iterace, zřetězení a sjednocení.
- Uzávěrové vlastnosti
  - dnes jen 'regulární' sloupec.

jazyk	regulární (RL)	bezkontextové	deterministické CFL
sjednocení	ANO	ANO	NE
průnik	ANO	NE	NE
$\cap$ s RL	ANO	ANO	ANO
doplněk	ANO	NE	ANO
homomorfizmus	ANO	ANO	NE
inverzní hom.	ANO	ANO	ANO