Dnešní téma

Regulární výrazy, Dvousměrné FA

Marta Vomlelová

MS 303

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 2.1

- 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 2.1 (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	$L(\emptyset) = \{\} \equiv \emptyset$
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 2.1

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

Příklady reguláních výrazů, priorita

Definition 2.2 (priorita)

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

Example 2.2 (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 2.1 (!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).*

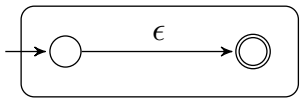
Převod RegE výrazu na ϵ–NFA automat

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:

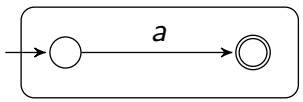
1. Právě jeden přijímající stav.
2. Žádné hrany do počátečního stavu.
3. Žádné hrany z koncového stavu.

Základ:



Prázdný řetězec ϵ

Prázdná množina $\emptyset$

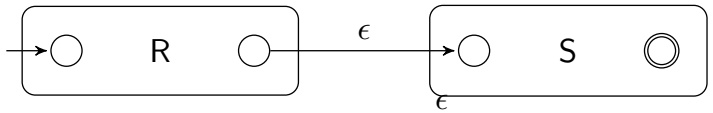


$a \in \Sigma$: výraz **a**

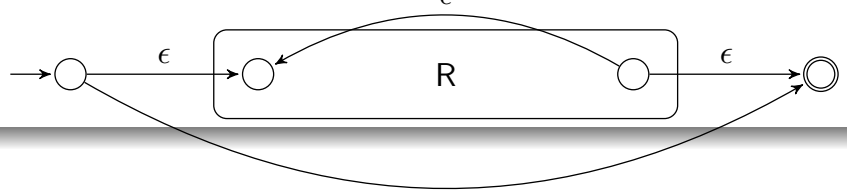
Součet $R + S$:

INDUKCE:

Zřetězení RS:



Iterace R^*:



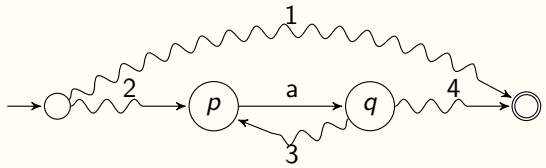
□

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 $\delta^|(p, a) = \delta(p, a) - \{q\}$)
    - $A_1 = (Q, \Sigma, \delta^|, q_0, F)$
    - $A_2 = (Q, \Sigma, \delta^|, q_0, \{p\})$
    - $A_3 = (Q, \Sigma, \delta^|, q, \{p\})$
    - $A_4 = (Q, \Sigma, \delta^|, q, F)$



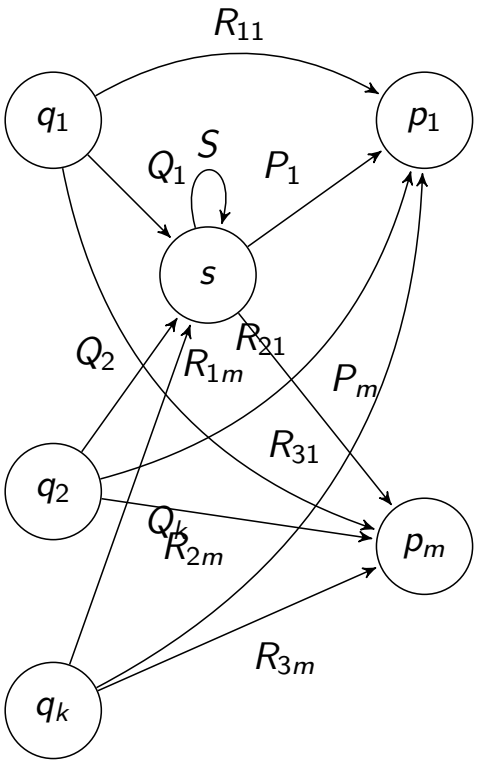
  - Potom $L(A) = L(A_1) \cup (L(A_2).a).(L(A_3).a)^* L(A_4)$,
  - jazyky $L(A_1), L(A_2), L(A_3), L(A_4) \in RJ(\Sigma)$ z indukčního předpokladu (n hran), označně α_i jejich regulární výrazy.
  - $L(A) = L(\alpha_1 + \alpha_2.a).(\alpha_3.a)^*.\alpha_4)$

□

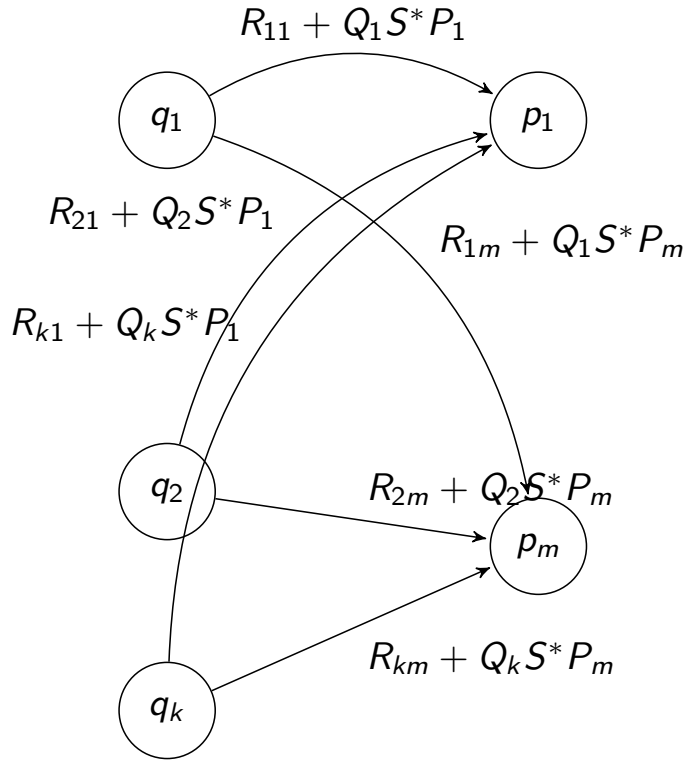
Konverze NFA na RegE eliminací stavů

- Následující algoritmus se občas vyhne duplicitě.
- Dovolíme regulární výrazy jako popisky na hranách grafu (transformovaného automatu).
- Iterativně eliminujeme uzly. Při eliminaci s, spojíme každého rodiče s každým dítětem s a anotujeme hranu.

Stav s vybrán k eliminaci



Výsledek eliminace s z předchozího grafu.

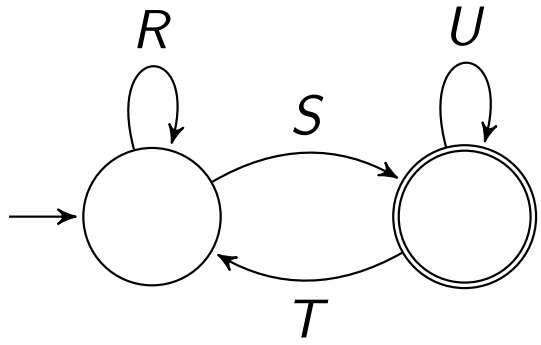


Konstrukce regulárního výrazu RegE z NFA

1. Pro každý cílový stav $q \in F$ aplikujeme předchozí redukci na všechny $p \in Q \setminus \{q, q_0\}$.

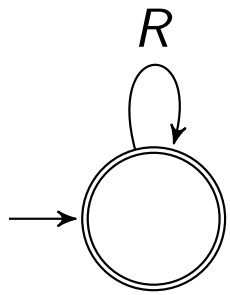
2. Pro $q \neq q_0$ vezmeme

$$RegE(q) = (R + SU^* T)^* SU^*.$$



3. Pro $q = q_0$ vezmeme

$$RegE(q) = R^*.$$

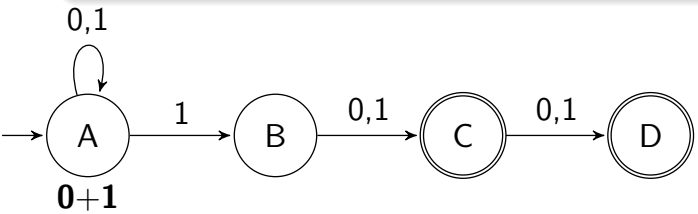


4. Sečteme výrazy (jazyky sjednotíme) přes všechny přijímající stavy; $RegE(NFA) = \oplus_{q \in F} RegE(q)$.

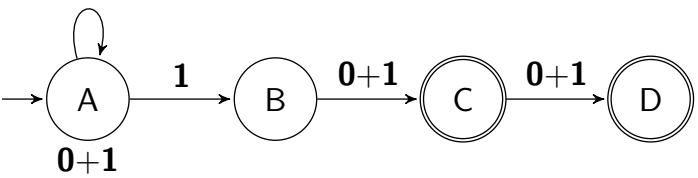
Příklad

Example 2.3

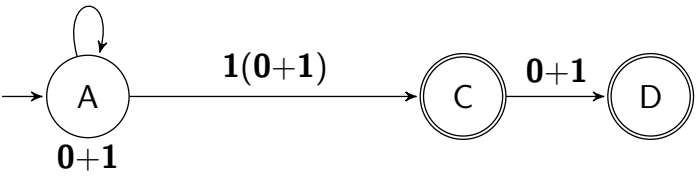
NFA přijímající slova s 1 na 2. nebo 3. pozici od konce.



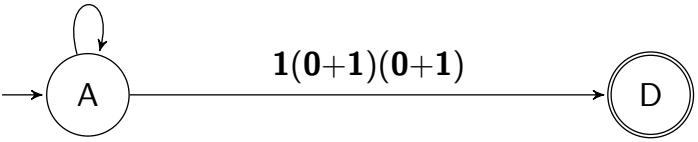
původní automat



Řetězce nahradíme RegE.



Eliminujeme B.



Eliminujeme C.

A máme RegE výraz: $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})+(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$.

[Pořadí eliminace]

1. Nejdřív eliminujeme uzly ne–cílové a ne–startovní $q \notin F, q \neq q_0$.

Zjednodušení regulárních výrazů (netřeba znát)

Lemma (Další vlastnosti bez důkazu)

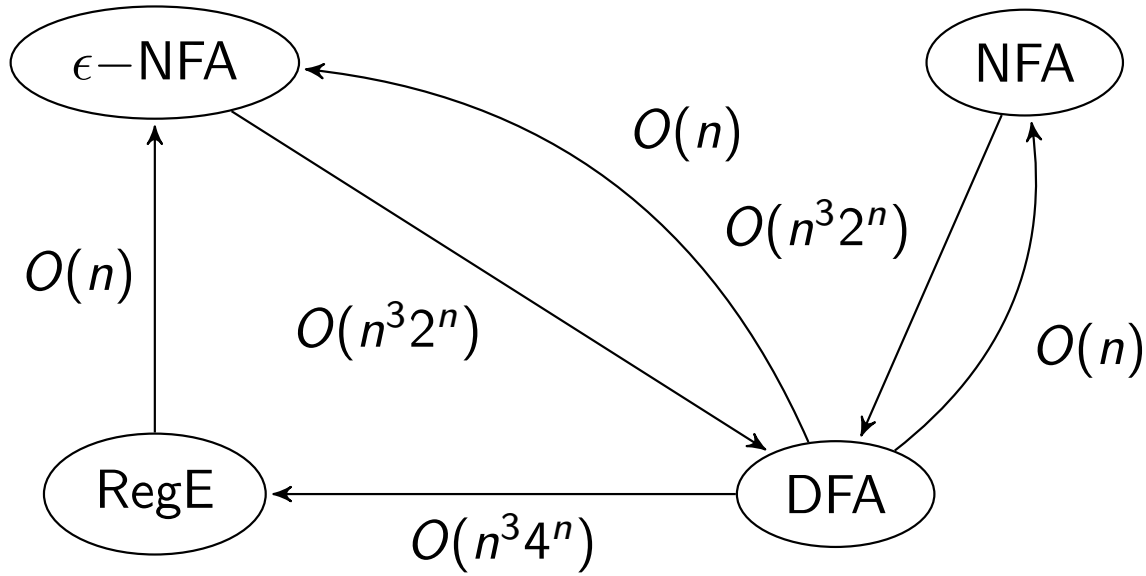
- Zjednodušení návrhu automatů

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

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 ϵ-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 2.3 (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 2.4 (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 2.4 (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 $h^{-1}(L) = \{w | h(w) \in L\}$.

Pozn. Inverzní homomorfizmus obecně není homomorfizmus, jedno slovo $h(w)$ může mít více 'rodičů' w.

Example 2.5 (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 2.2 (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.

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

- Podtržení $\underline{\sigma}$ je technický detail pro ty, kteří rozliší σ aplikované na jazyk od $\underline{\sigma}$ aplikovaného na regulární výraz.
- Pro iteraci rozložíme na nekonečné sjednocení, pro každý konkrétní počet iterací σ aplikované na konečné zřetězení.

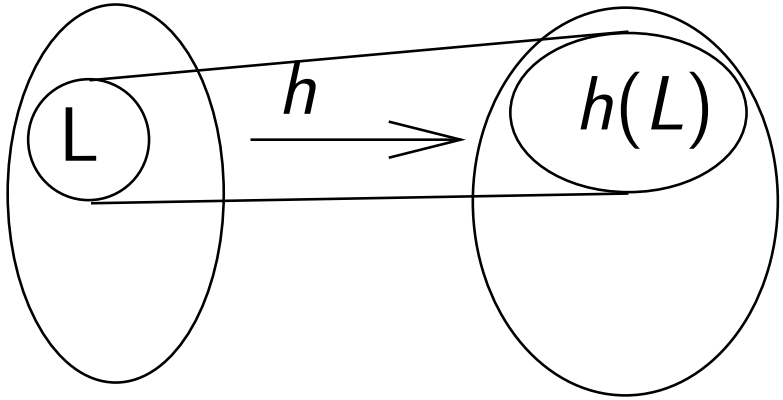
$$\begin{aligned} \sigma(L(\alpha)^*) &= \sigma(L(\alpha)^0) \cup \sigma(L(\alpha)^1) \cup \ldots \cup \sigma(L(\alpha)^n) \cup \ldots \\ &= \underline{\sigma}(\alpha)^0 \cup \underline{\sigma}(\alpha)^1 \cup \ldots \cup \underline{\sigma}(\alpha)^n \cup \ldots \\ &= L(\underline{\sigma}(\alpha)^*).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

Inverzní homomorfizmus

Definition ((2.4) Inverzní homomorfizmus)

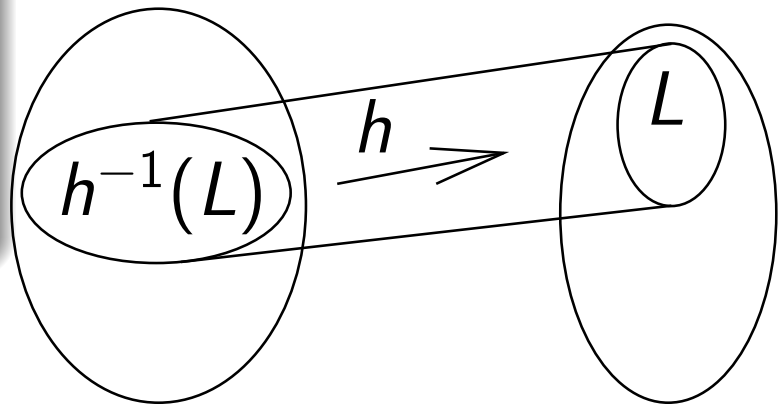
Nechť h je homomorfizmus abecedy T do slov nad abecedou Σ. Pak $h^{-1}(L)$ *'h inverze L'* je množina řetězců
$h^{-1}(L) = \{w | w \in T^*;\ h(w) \in L\}$.

Homomorfizmus aplikovaný dopředně a zpětně.



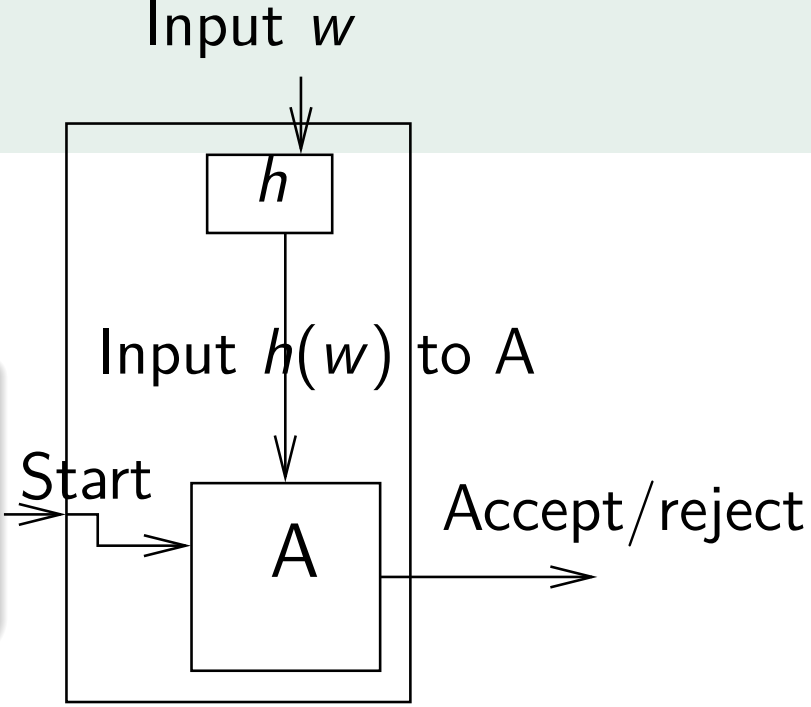
Example 2.6

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 2.3

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

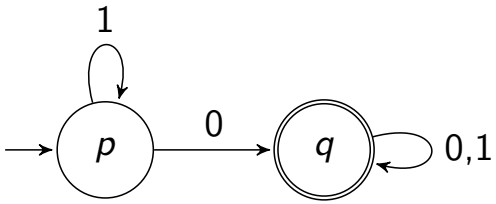
$$Q' = \{[q, u] \,|\, q \in Q, u \in \Sigma^*, \exists(a \in T)\exists(v \in \Sigma^*)h(a) = vu\} \quad u \text{ je buffer}$$
$$\delta'([q, \epsilon], a) = [q, h(a)] \quad \text{naplňuje buffer}$$
$$\delta'([q, bv], \epsilon) = [p, v] \text{ kde } \delta(q, b) = p \quad \text{čte buffer.}$$

Příklad: Navštiv všechny stavy (Letos vynechán)

Example 2.7

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 $O(|Q|^2)$.

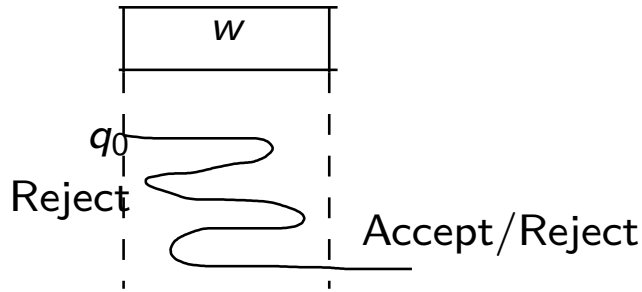
Lemma (Test náležení do regulárního jazyka)

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.

Dvousměrné (dvoucestné) konečné automaty

- Konečný automat provádí následující činnosti:
  - přečte písmeno
  - změní stav vnitřní jednotky
  - posune čtecí hlavu doprava
- Čtecí hlava se nesmí vracet.



Definition 2.5 (Deterministické Dvousměrné konečné automaty)

Deterministickým dvousměrným konečným automatem nazýváme pětici $A = (Q, \Sigma, \delta, q_0, F)$, kde

1. Q je konečná množina stavů,
2. Σ je konečná množina vstupních symbolů
3. přechodové funkce δ je zobrazení $Q \times \Sigma \to Q \times \{-1, 1\}$ rozšířené o pohyb hlavy
4. $q_0 \in Q$ počáteční stav
5. množina přijímajících stavů $F \subseteq Q$.

Pozn.: Je deterministický, nedeterministický $\delta_N : Q \times \Sigma \to \mathcal{P}(Q \times \{-1, 1\})$.

Pozn.2: Nulový pohyb hlavy lze, jen trochu zkomplikuje důkaz dále.

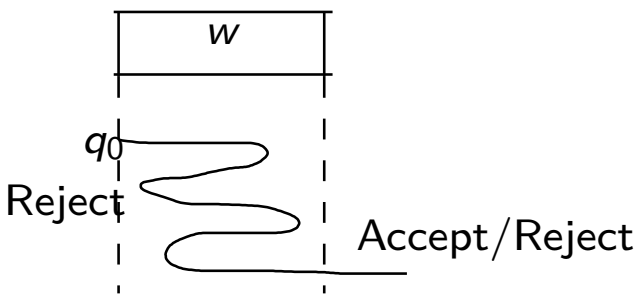
- Reprezentujeme opět stavovým diagramem, tabulkou.

Výpočet dvousměrného automatu

Definition 2.6 (Výpočet dvousměrného automatu)

Slovo w je **přijato dvousměrným konečným automatem**, pokud:

- výpočet začal na prvním písmenu slova w vlevo v počátečním stavu
- čtecí hlava poprvé opustila slovo w vpravo v některém přijímajícím stavu
- mimo čtené slovo není výpočet definován (výpočet zde končí a slovo není přijato).



- Ke slovům si můžeme přidat speciální koncové znaky $\# \notin \Sigma$
- funkce $\partial_\#$ odstraní $\#$ zleva, $\partial_\#^R$ zprava.

Lemma

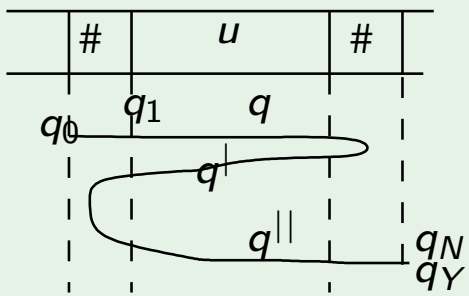
Je–li $L(A) = \{\#w\# | w \in L \subseteq \Sigma^*\}$ regulární,
potom je regulární i $L = \partial_\# \partial_\#^R (L(A) \cap \#\Sigma^*\#)$.

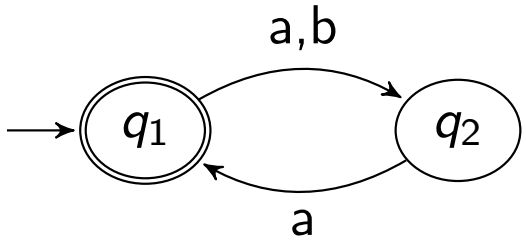
Příklad dvousměrného automatu

Example 2.8 (Příklad dvousměrného automatu)

Nechť $A = (Q, \Sigma, \delta, q_1, F)$. Dvousměrný konečný automat $B = (Q \cup Q^{|} \cup Q^{||} \cup \{q_0, q_N, q_Y\}, \Sigma \cup \{\#\}, \delta^{|}, q_0, \{q_Y\})$ přijímající jazyk $L(B) = \{\#u\# | uu \in L(A)\}$ (toto NENÍ levý ani pravý kvocient!) definujeme následovně:

$\delta^{\|}$	$x \in \Sigma$	$\#$	poznámka
q_0	$q_N, -1$	$q_1, +1$	q_1 je počátek A
q	$p, +1$	$q^{\|}, -1$	$p = \delta(q, x)$
$q^{\|}$	$q^{\|}, -1$	$q^{\|\|}, +1$	
$q^{\|\|}$	$p^{\|\|}, +1$	$q_Y, +1$	$q \in F, p = \delta(q, x)$
$q^{\|\|}$	$p^{\|\|}, +1$	$q_N, +1$	$q \notin F, p = \delta(q, x)$
q_N	$q_N, +1$	$q_N, +1$	
q_Y	$q_N, +1$	$q_N, +1$	





Dvousměrné a jednosměrné konečné automaty

Theorem 2.4

Jazyky přijímané dvousměrnými konečnými automaty jsou právě regulární jazyky.

Proof: konečný automat $\rightarrow$ dvousměrný automat

- Konečný automat předeveme na dvousměrný přidáním posunu hlavy vpravo
- $A = (Q, \Sigma, \delta, q_0, F) \rightarrow 2A = (Q, \Sigma, \delta^{|}, q_0, F)$, kde $\delta^{|}(q, x) = (\delta(q, x), +1)$.

□

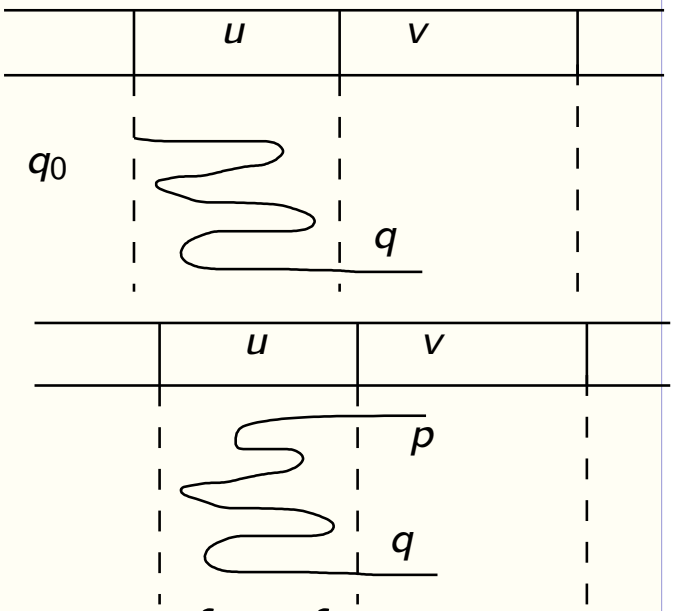
- Možnost pohybovat čtecí hlavou po pásce nezvětšila sílu konečného automatu (dokud na pásku nic nepíšeme!).
- Pro důkaz potřebujeme přípravu.

Funkce f_u popisující výpočet 2DFA nad slovem u

Algorithm: Funkce f_u popisující výpočet 2DFA nad slovem u

Definujeme funkci $f_u : Q \cup \{q_0^{|}\} \to Q \cup \{0\}$

- $f_u(q_0^{|})$ popisuje v jakém stavu poprvé odejdeme vpravo, pokud začneme výpočet vlevo v počátečním stavu q_0,
- symbol 0 značí, že daná situace nenastane (odejdeme vlevo nebo cyklus),
- $f_u(p); p \in Q$ v jakém stavu opět odejdeme vpravo, pokud začneme výpočet vpravo v p



- Definujeme ekvivalenci slov následovně: $u \sim w \Leftrightarrow_{def} f_u = f_w$,
  - tj. slova jsou ekvivalentní pokud mají stejné 'výpočtové' funkce.

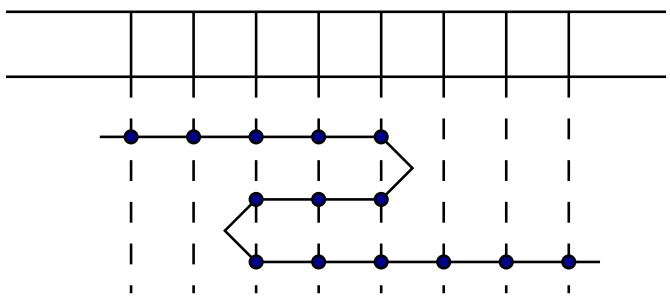
Regulárnost 2DFA

Ekvivalence $\sim$ je ekvivalence, má konečný index, je to pravá kongruence, jazyk 2DFA odpovídá sjednocení tříd $f_w(q_0^{|}) \in F$.

Podle Myhill–Nerodovy věty je $L(A)$ regulární jazyk.

Konstruktivní důkaz věty o 2DFA

- Potřebujeme převést návraty na lineární výpočet.
- Zajímají nás jen přijímající výpočty
- Díváme se na řezy mezi symboly (v jakém stavu přechází na další políčko)



Pozorování:

- stavy se v přechodu řezu střídají (doprava, doleva)
- první stav jde doprava, poslední také doprava
- v deterministických přijímajících výpočtech nejsou cykly
- první a poslední řez obsahují jediný stav.

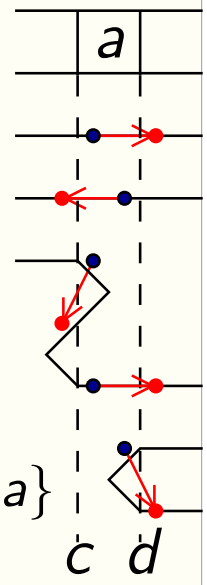
Algorithm: 2DFA $\rightarrow$ NFA

- Najdeme všechny možné řezy – posloupnosti stavů (je jich konečně mnoho).
- Mezi řezy definujeme nedeterministické přechody podle čteného symbolu.
- Rekonstruujeme výpočet skládáním řezů jako puzzle.

Algorithm: Formální převod 2DFA na NFA

Nechť $A = (Q, \Sigma, \delta, q_0, F)$ je dvousměrný (deterministický) konečný automat. Definujeme ekvivalentní nedeterministický automat $B = (Q^{|}, \Sigma, \delta^{|}, (q_0), F^{|})$ kde:

- $Q^{|}$ jsou všechny korektní přechodové posloupnosti
  - posloupnosti stavů $(q^1, \ldots, q^k); q^i \in Q$
  - délka posloupnosti je lichá $(k = 2m + 1)$
  - žádný stav se neopakuje na liché ani na sudé pozici $(\forall i \neq j)\ (q^{2i} \neq q^{2j})\ \&\ (\forall i \neq j)(q^{2i+1} \neq q^{2j+1})$
- $F^{|} = \{(q) | q \in F\}$ posloupnosti délky 1
- $\delta^{|}(c, a) = \{d | d \in Q^{|} \& c \xrightarrow{a} d$ je lokálně konzistentní přechod pro $a\}$
  - existuje bijekce: $h : c_{odd} \cup d_{even} \to c_{even} \cup d_{odd}$, tak, že:
  - zachovává uspořádání
  - pro $h(q) \in c_{even}$ je $(h(q), -1) = \delta(q, a)$
  - pro $h(q) \in d_{odd}$ je $(h(q), +1) = \delta(q, a)$.



$L(A) = L(B)$

Trajektorie 2DFA A odpovídá řezům v FA B, odtud $L(A) = L(B)$.

Příklad převodu 2DFA na NKA

- Mějme následující dvousměrný konečný automat:

	a	b
$\rightarrow p$	p,+1	q,+1
$*q$	q,+1	r,-1
r	p,+1	r,-1

Možné řezy a jejich přechody

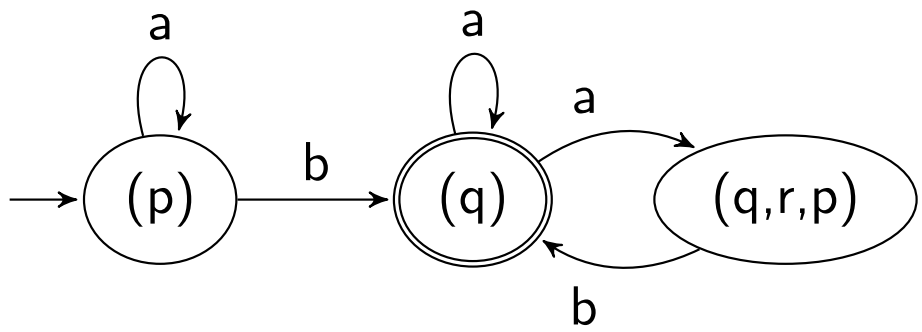
- Doleva jedině r – všechny sudé pozice r, tj. jediná sudá
- možné řezy: $(p), (q), (p, r, q), (q, r, p)$.

	a	b
$\rightarrow (p)$	(p)	(q)
$*(q)$	(q),(q,r,p)	
(p,r,q)		
(q,r,p)		(q)

Ukázka (zacykleného, nepřijímajícího) výpočtu:

a	a	b	a	a	b	a	a	b	b
p	p	p	q	q	q				
				r					
				p	q	q	q		
						r			
						p	q		
						r	r		
						p	q		

Výsledný NFA:



Automaty s výstupem (motivace)

- Dosud jediná zpráva z automatu: 'Jsme v přijímajícím stavu'.
- Můžeme z FA získat více informací? Můžeme zaznamenat trasu výpočtu?

Moore: indikace stavů (všech, nejen koncových)

- v každé chvíli víme, kde se automat nachází
- Příklad: různé (regulární) čítače

Mealy: indikace přechodů

- po přečtení každého symbolu víme, co automat dělal
- Příklad: regulární překlad slov

Automat už není tak docela černá skříňka.

Mooreův stroj

Definition 2.7 (Mooreův stroj)

Mooreovým (sekvenčním) strojem nazýváme šestici $A = (Q, \Sigma, Y, \delta, \mu, q_0)$ resp. pětici $A = (Q, \Sigma, Y, \delta, \mu)$, kde

Q je konečná neprázdná množina stavů

Σ je konečná neprázdná množina symbolů (vstupní abeceda)

Y je konečná neprázdná množina symbolů (**výstupní abeceda**)

δ je zobrazení $Q \times \Sigma \to Q$ (přechodová funkce)

μ je zobrazení $Q \to Y$ (**značkovací funkce**)

$q_0 \in Q$ (počáteční stav)

- Někdy nás nezajímá počáteční stav, ale jen práce automatu
- značkovací funkce umožňuje suplovat roli koncových stavů
  - $F \subseteq Q$ nahradíme značkovací funkcí $\mu : Q \to \{0, 1\}$ takto:
    $\mu(q) = 0$ pokud $q \notin F$,
    $\mu(q) = 1$ pokud $q \in F$.

Příklad Mooreova stroje

Example 2.9 (Tenisový set)

Mooreův stroj pro počítání tenisového skóre.

- Vstupní abeceda: ID hráče, který uhrál bod
- Výstupní abeceda & stavy: skóre (tj. $Q = Y$ a $\mu(q) = q$)

Stav/výstup	A	B
00:00	15:00	00:15
15:00	30:00	15:15
15:15	30:15	15:30
00:15	15:15	00:30
30:00	40:00	30:15
30:15	40:15	30:30
30:30	40:30	30:40
15:30	30:30	15:40
00:30	15:30	00:40
40:00	A	40:15
40:15	A	40:30
40:30	A	shoda
30:40	shoda	B
15:40	30:40	B
00:40	15:00	B
shoda	A:40	40:B
A:40	A	shoda
40:B	shoda	B
A	15:00	00:15
B	15:00	00:15

Mealyho stroj

Definition 2.8 (Mealyho stroj)

Mealyho (sekvenčním) strojem nazýváme šestici $A = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$ resp. pětici $A = (Q, \Sigma, Y, \delta, \lambda_M)$, kde

Q je konečná neprázdná množina stavů

Σ je konečná neprázdná množina symbolů (vstupní abeceda)

Y je konečná neprázdná množina symbolů (výstupní abeceda)

δ je zobrazení $Q \times \Sigma \to Q$ (přechodová funkce)

λ_M je zobrazení $Q \times \Sigma \to Y$ (**výstupní funkce**)

$q_0 \in Q$ (počáteční stav)

- Výstup je určen stavem a vstupním symbolem
  - Mealyho stroj je obecnějším prostředkem než stroj Mooreův
  - Značkovací funkci $\mu : Q \to Y$ lze nahradit výstupní funkcí $\lambda_M : Q \times \Sigma \to Y$ například takto:

    $$\forall x \in \Sigma \ \lambda_M(q, x) = \mu(q)$$

    nebo $\forall x \in \Sigma \ \lambda_M(q, x) = \mu(\delta(q, x))$

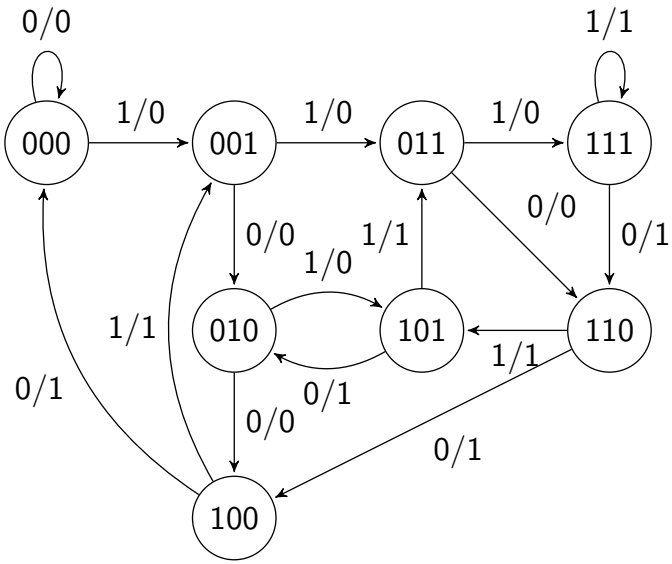
Příklad Mealyho stroje

Example 2.10 (Mealyho stroj)

Automat, který dělí vstupní slovo v binárním tvaru číslem 8 (celočíselně).

- Posun o tři bity doprava
- potřebujeme si pamatovat poslední trojici bitů
- vlastně tříbitová dynamická paměť

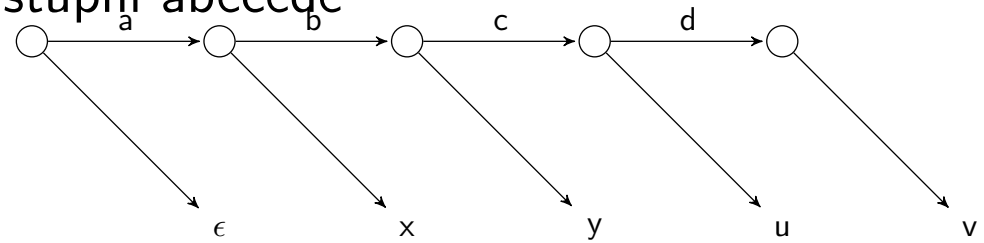
Stav\symbol	0	1
000	000/0	001/0
001	010/0	011/0
010	100/0	101/0
011	110/0	111/0
100	000/1	001/1
101	010/1	011/1
110	100/1	101/1
111	110/1	111/1



- I když nevíme, kde automat startuje, po třech symbolech začne počítat správně.

Výstup sekvenčních strojů

slovo ve vstupní abecedě $\rightarrow$ slovo ve výstupní abecedě



Mooreův stroj

značkovací funkce $\mu : Q \rightarrow Y$

$\mu^* : Q \times \Sigma^* \rightarrow Y^*$

$\mu^*(q, \epsilon) = \epsilon$ (někdy $\mu^*(q, \epsilon) = q$)

$\mu^*(q, wx) = \mu^*(q, w).\mu(\delta^*(q, wx))$

Příklad: μ^*(00:00,AABA)=(00:00 .) 15:00 . 30:00 . 30:15 . 40:15

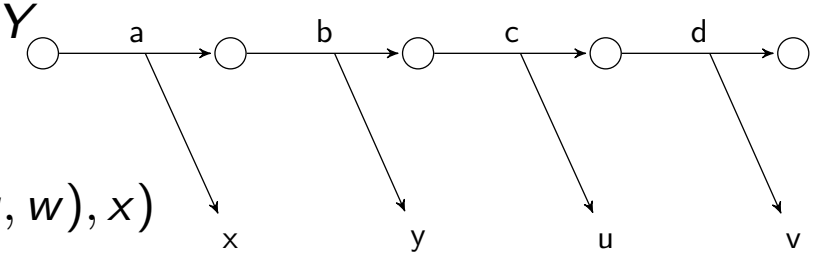
Mealyho stroj

výstupní funkce $\lambda_M : Q \times \Sigma \rightarrow Y$

$\lambda_M^* : Q \times \Sigma^* \rightarrow Y^*$

$\lambda_M^*(q, \epsilon) = \epsilon$

$\lambda_M^*(q, wx) = \lambda_M^*(q, w).\lambda_M(\delta^*(q, w), x)$



Příklad: λ_M^*(000,1101010)=0001101

Konečné automaty – shrnutí

Konečný automat

- redukovaný deterministický automat (lze definovat i jednoznačný)
- nedeterminismus ϵ–NFA, 2^n, (dvousměrný FA n^n)

Regulární výrazy

Automaty a jazyky

- regulární jazyky
- uzavřenost na množinové operace
- uzavřenost na řetězcové operace
- uzavřenost na substituci, homomorfizmus a inverzní homomorfizmus,
- automaty výše i regulární výrazy popisují stejnou třídu jazyků.

Charakteristika regulárních jazyků

- Mihyll–Nerodova věta (kongruence)
- Kleeneova věta (elementární jazyky a operace)
- Iterační (pumping) lemma (iterace podslov, jen nutná podmínka).

Dvousměrný automat, pokud nesmí psát na pásku, přijímá jen regulární jazyky.

- Dvousměrné automaty a automaty s výstupem se nezkouší.