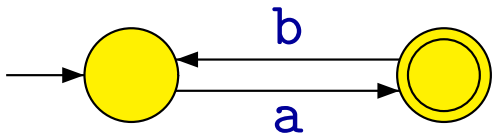
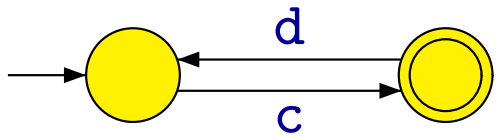
Zřetězení jazyků

$\Sigma = \{a, b, c, d\}$

A_1:



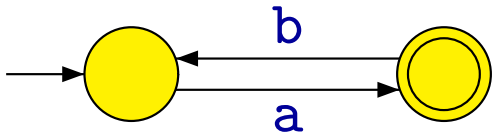
A_2:



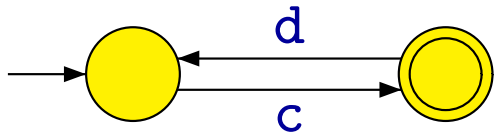
Zřetězení jazyků

$\Sigma = \{a, b, c, d\}$

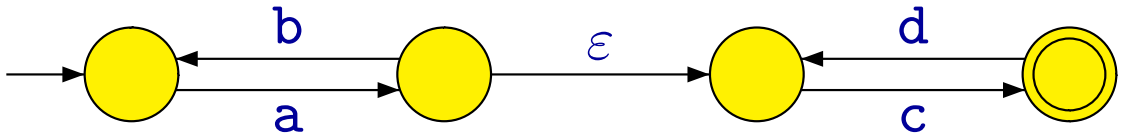
A_1:



A_2:



A:

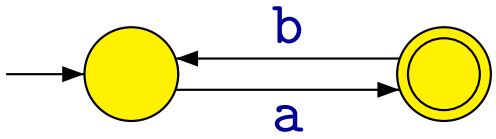


$L(A) = L(A_1) \cdot L(A_2)$

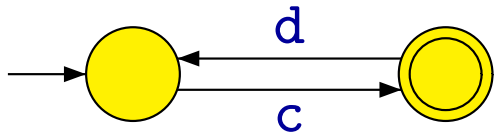
Zřetězení jazyků

$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

A_1:

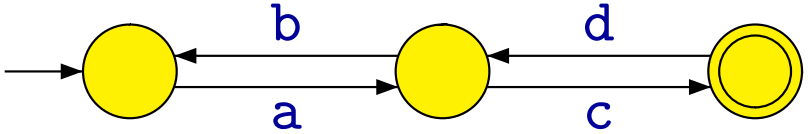


A_2:



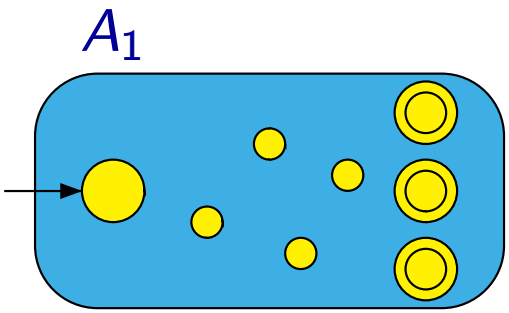
Chybná konstrukce:

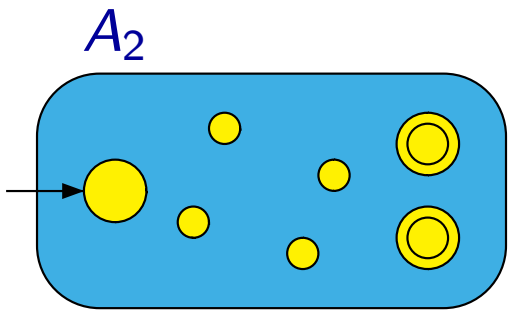
A:



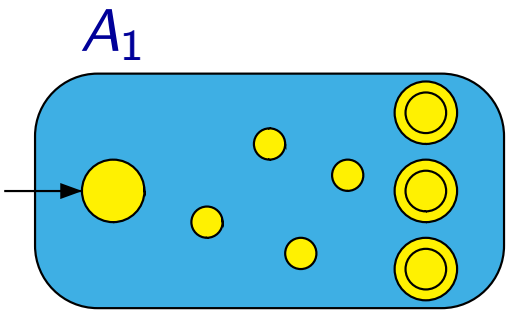
$\mathtt{acdbac} \in L(A)$, ale $\mathtt{acdbac} \notin L(A_1) \cdot L(A_2)$

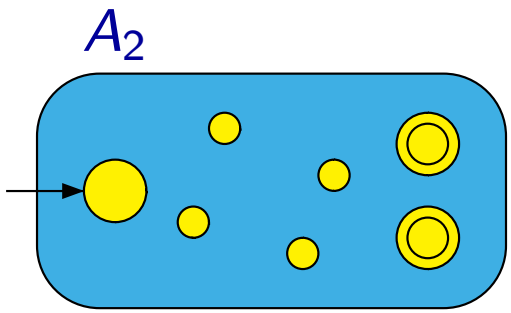
Zřetězení jazyků

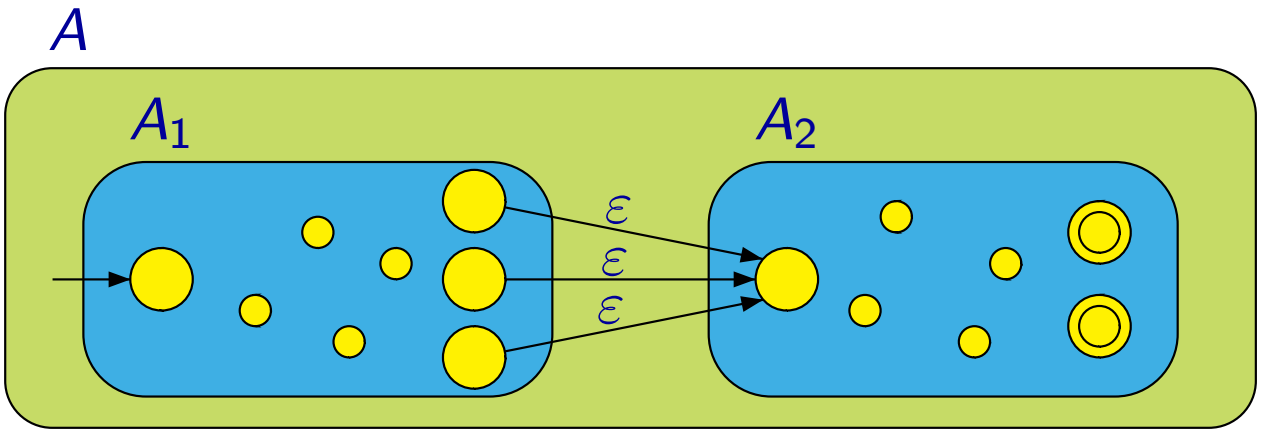




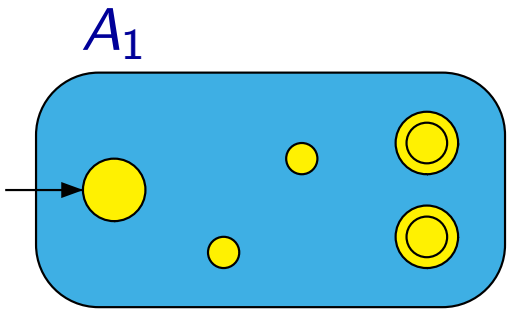
Zřetězení jazyků



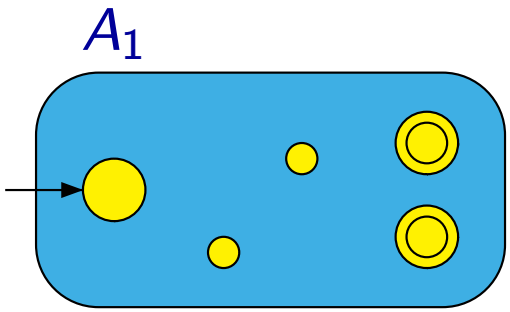


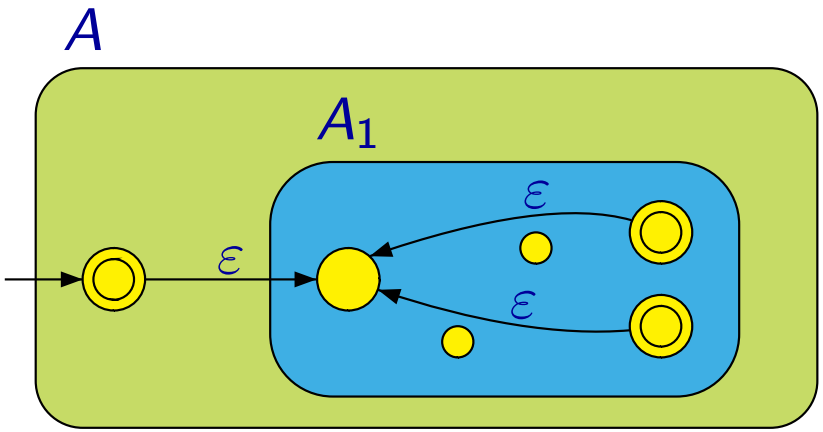


Iterace jazyka



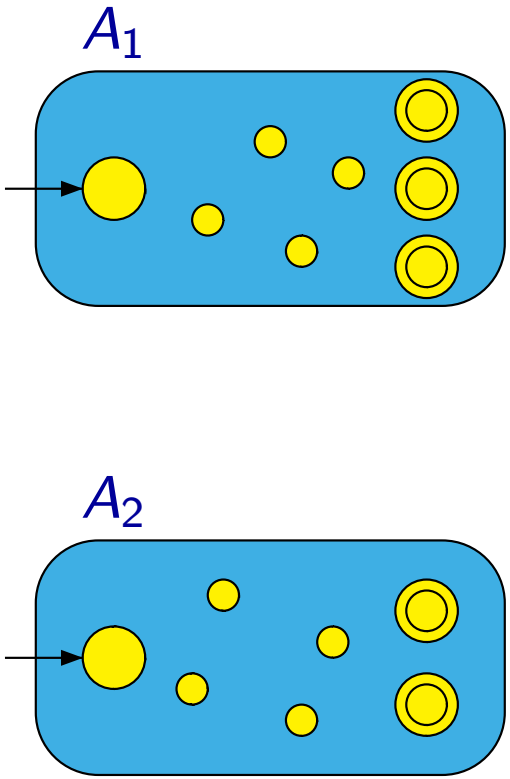
Iterace jazyka





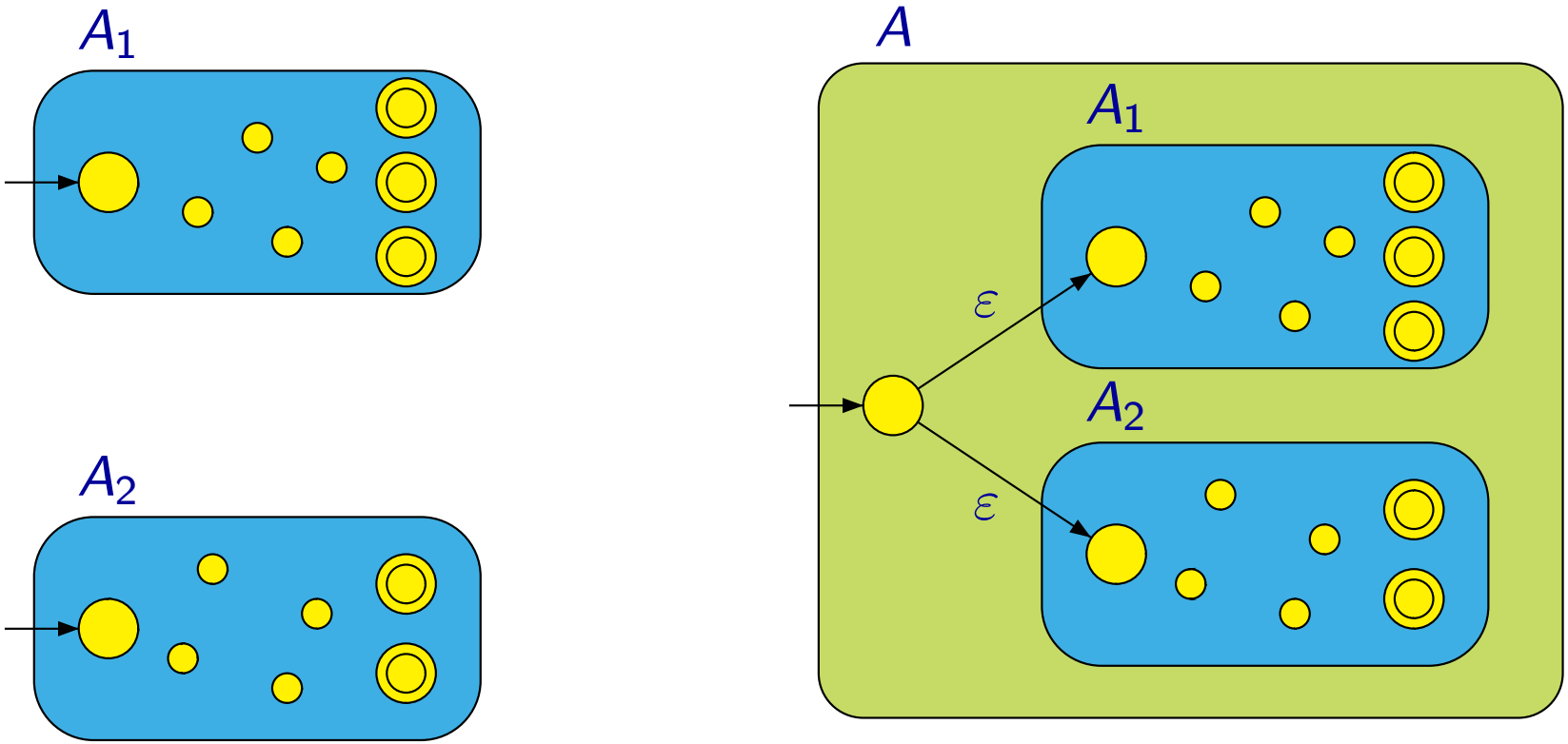
Sjednocení jazyků

Alternativní konstrukce pro sjednocení jazyků:



Sjednocení jazyků

Alternativní konstrukce pro sjednocení jazyků:



Uzavřenost množiny vůči operacím

Předpokládejme, že máme dány množiny X a Y, kde $X \subseteq Y$, a dále nějakou operaci $f : Y \times Y \times \cdots \times Y \to Y$.

O množině X řekneme, že je **uzavřená** vůči operaci f, jestliže platí, že pokud $x_1, x_2, \cdots, x_k \in X$, pak $f(x_1, x_2, \cdots, x_k) \in X$.

Příklad: Uvažujme množinu přirozených čísel $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ a množinu reálných čísel $\mathbb{R}$.

- Množina $\mathbb{N}$ je uzavřená vůči operacím $+$ a $\times$, ale není uzavřená vůči operacím $-$ a $/$.

  Například $3 + 5 \in \mathbb{N}$, ale $3 - 5 \notin \mathbb{N}$ a $3/5 \notin \mathbb{N}$
- Množina $\mathbb{R}$ je uzavřená vůči operacím $+$, $-$, $\times$.
- Množina $\mathbb{R} - \{0\}$ je uzavřená vůči operaci $/$.

Uzavřenost třídy regulárních jazyků

Množina (všech) regulárních jazyků je uzavřená vůči operacím:

- sjednocení
- průniku
- doplňku
- zřetězení
- iteraci
- ...

Poznámka: Jazyk je regulární, pokud existuje konečný automat, který ho rozpoznává.

Existují i jazyky, které nejsou regulární.

Regulární výrazy

Regulární výrazy

Jako například v aritmetice můžeme pomocí operátorů $+$ a $\times$ vytvářet výrazy jako

$$(5 + 3) \times 4$$

můžeme v teorii formálních jazyků pomocí operátorů $+$, $\cdot$ a * vytvářet tzv. **regulární výrazy**, jako třeba

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

které reprezentují jazyky.

Jako je hodnotou aritmetického výrazu $(5 + 3) \times 4$ číslo 32, je hodnotou regulárního výrazu $(0 + 1) \cdot 0^*$ jazyk

$$(\{0\} \cup \{1\}) \cdot \{0\}^*$$

Regulární výrazy

Induktivní definice regulárních výrazů nad abecedou Σ:

- $\emptyset$, ε, a (kde $a \in \Sigma$) jsou regulární výrazy:

  $\emptyset$ ... označuje prázdný jazyk
  ε ... označuje jazyk $\{\varepsilon\}$
  a ... označuje jazyk $\{a\}$

- Jestliže α, β jsou regulární výrazy, pak i $(\alpha + \beta)$, $(\alpha \cdot \beta)$, (α^*) jsou regulární výrazy:

  $(\alpha + \beta)$ ... označuje sjednocení jazyků označených α a β
  $(\alpha \cdot \beta)$ ... označuje zřetězení jazyků označených α a β
  (α^*) ... označuje iteraci jazyka označeného α

- Neexistují žádné další regulární výrazy než ty definované podle předchozích dvou bodů.

Regulární výrazy

Příklad:

- Podle definice jsou 0 i 1 regulární výrazy.

Regulární výrazy

Příklad:

- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.

Regulární výrazy

Příklad:

- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.
- Protože 0 je regulární výraz, je i (0^*) regulární výraz.

Regulární výrazy

Příklad:

- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.
- Protože 0 je regulární výraz, je i (0^*) regulární výraz.
- Protože $(0 + 1)$ i (0^*) jsou regulární výrazy, je i $((0 + 1) \cdot (0^*))$ regulární výraz.

Regulární výrazy

Příklad:

- Podle definice jsou 0 i 1 regulární výrazy.
- Protože 0 i 1 jsou regulární výrazy, je i $(0 + 1)$ regulární výraz.
- Protože 0 je regulární výraz, je i (0^*) regulární výraz.
- Protože $(0 + 1)$ i (0^*) jsou regulární výrazy, je i $((0 + 1) \cdot (0^*))$ regulární výraz.

Poznámka: Jestliže α je regulární výraz, zápisem $[\alpha]$ označujeme jazyk definovaný regulárním výrazem α.

$$[((0 + 1) \cdot (0^*))] = \{0, 1, 00, 10, 000, 100, 0000, 1000, 00000, \dots\}$$

Regulární výrazy

Aby byl zápis regulárních výrazů přehlednější a stručnější, používáme následují pravidla:

- Vynecháváme vnější pár závorek.
- Vynecháváme závorky, které jsou zbytečné vzhledem k asociativitě operací sjednocení ($+$) a zřetězení ($\cdot$).
- Vynecháváme závorky, které jsou zbytečné vzhledem k prioritě operací (nejvyšší prioritu má iterace (*), menší zřetězení ($\cdot$) a nejmenší sjednocení ($+$)).
- Nepíšeme tečku pro zřetězení.

Příklad: Místo

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

obvykle píšeme

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

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 . . . jazyk tvořený jediným slovem 0

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 . . . jazyk tvořený jediným slovem 0

01 . . . jazyk tvořený jediným slovem 01

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$ ... jazyk tvořený dvěma slovy 0 a 1

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$ ... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy ε, 0, 00, 000, ...

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$ ... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy ε, 0, 00, 000, ...

$(01)^*$ ... jazyk tvořený slovy ε, 01, 0101, 010101, ...

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$ ... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy ε, 0, 00, 000, ...

$(01)^*$ ... jazyk tvořený slovy ε, 01, 0101, 010101, ...

$(0 + 1)^*$ ... jazyk tvořený všemi slovy nad abecedou $\{0, 1\}$

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$ ... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy ε, 0, 00, 000, ...

$(01)^*$ ... jazyk tvořený slovy ε, 01, 0101, 010101, ...

$(0 + 1)^*$ ... jazyk tvořený všemi slovy nad abecedou $\{0, 1\}$

$(0 + 1)^*00$ ... jazyk tvořený všemi slovy končícími 00

Regulární výrazy

Příklady: Ve všech případech $\Sigma = \{0, 1\}$.

0 ... jazyk tvořený jediným slovem 0

01 ... jazyk tvořený jediným slovem 01

$0 + 1$ ... jazyk tvořený dvěma slovy 0 a 1

0^* ... jazyk tvořený slovy ε, 0, 00, 000, ...

$(01)^*$ ... jazyk tvořený slovy ε, 01, 0101, 010101, ...

$(0 + 1)^*$ ... jazyk tvořený všemi slovy nad abecedou $\{0, 1\}$

$(0 + 1)^*00$ ... jazyk tvořený všemi slovy končícími 00

$(01)^*111(01)^*$ ... jazyk tvořený všemi slovy obsahujícími podslovo 111 předcházené i následované libovolným počtem slov 01

Regulární výrazy

$(0+1)^*00 + (01)^*111(01)^*$ ... jazyk tvořený všemi slovy, která buď končí 00 nebo obsahují podslovo 111 předcházené i následované libovolným počtem slov 01

Regulární výrazy

$(0+1)^*00+(01)^*111(01)^*$ ... jazyk tvořený všemi slovy, která buď končí 00 nebo obsahují podslovo 111 předcházené i následované libovolným počtem slov 01

$(0+1)^*1(0+1)^*$ ... jazyk tvořený všemi slovy obsahujícími alespoň jeden symbol 1

Regulární výrazy

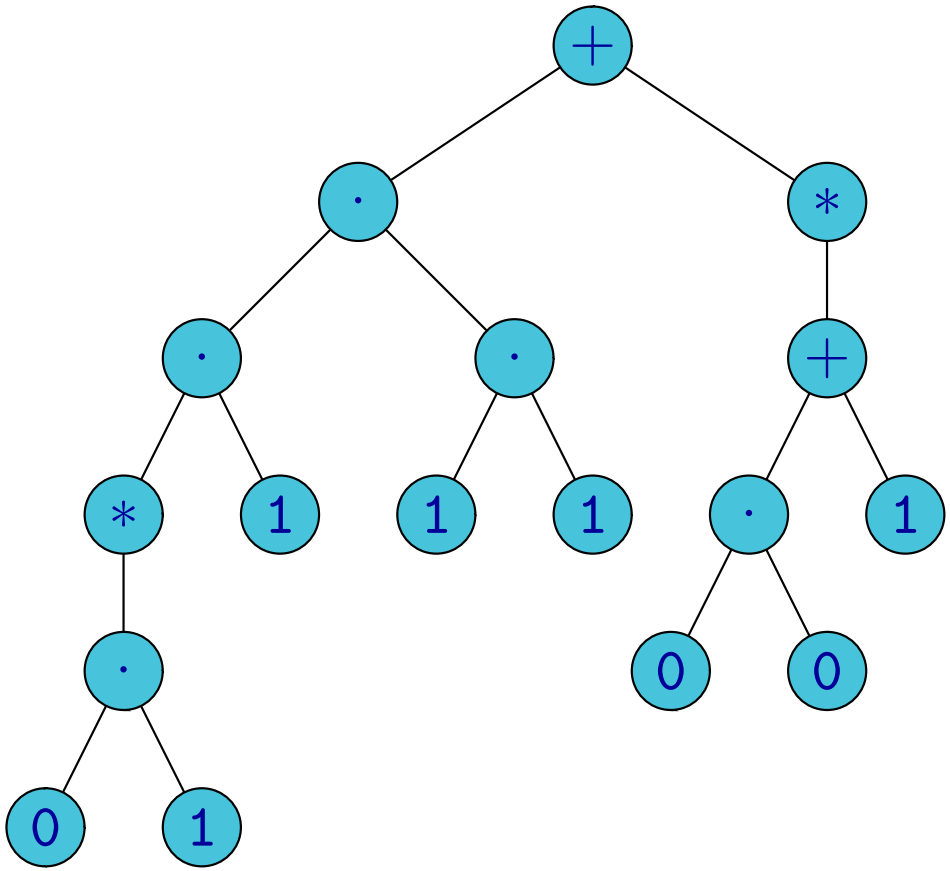
$(0+1)^*00+(01)^*111(01)^*$ ... jazyk tvořený všemi slovy, která buď končí 00 nebo obsahují podslovo 111 předcházené i následované libovolným počtem slov 01

$(0+1)^*1(0+1)^*$ ... jazyk tvořený všemi slovy obsahujícími alespoň jeden symbol 1

$(0^*10^*10^*)^*$ ... jazyk tvořený všemi slovy obsahujícími sudý počet symbolů 1

Regulární výrazy

Strukturu regulárního výrazu si můžeme znázornit jako strom:



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

Převod regulárního výrazu na konečný automat

Tvrzení

Každý jazyk, který je možné vyjádřit regulárním výrazem, je regulární (tj. rozpoznávaný nějakým konečným automatem).

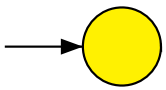
Důkaz: Stačí ukázat, jak k danému regulárnímu výrazu α zkonstruovat konečný automat, který rozpoznává jazyk $[\alpha]$.

Konstrukce je rekurzivní a postupuje podle struktury výrazu α:

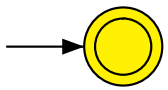
- Pokud je α elementární výraz (tj. $\emptyset$, ε nebo a):
  - Sestrojíme přímo odpovídající automat.
- Pokud je α tvaru $(\beta + \gamma)$, $(\beta \cdot \gamma)$ nebo (β^*):
  - Rekurzivně sestrojíme automaty rozpoznávající jazyky $[\beta]$ a $[\gamma]$.
  - Z nich sestrojíme automat rozpoznávající jazyk $[\alpha]$.

Převod regulárního výrazu na konečný automat

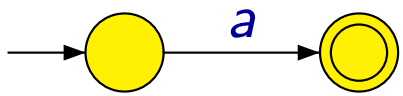
Automaty pro elementární výrazy:



$\emptyset$



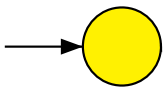
ε



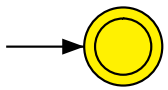
a

Převod regulárního výrazu na konečný automat

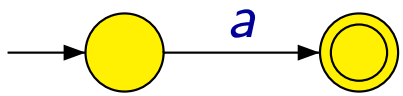
Automaty pro elementární výrazy:



$\emptyset$

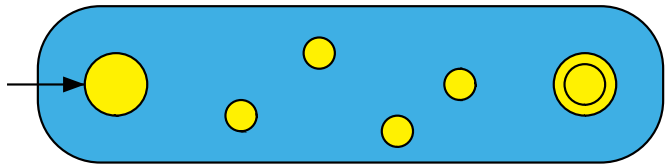


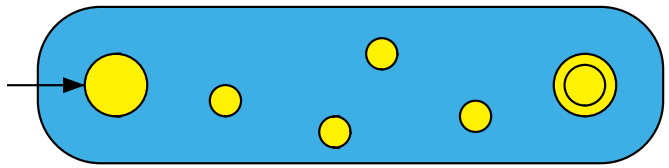
ε



a

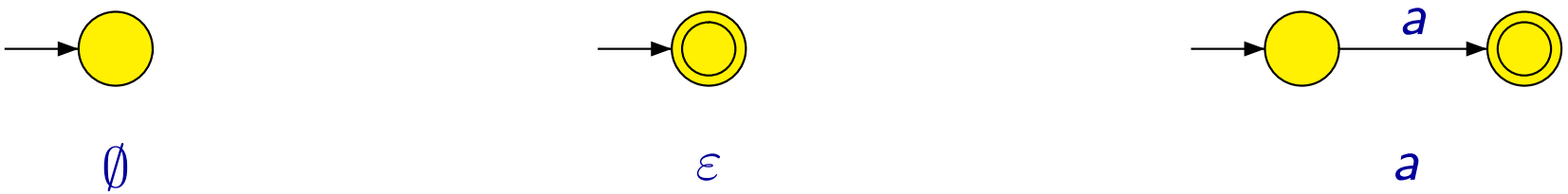
Konstrukce pro sjednocení:



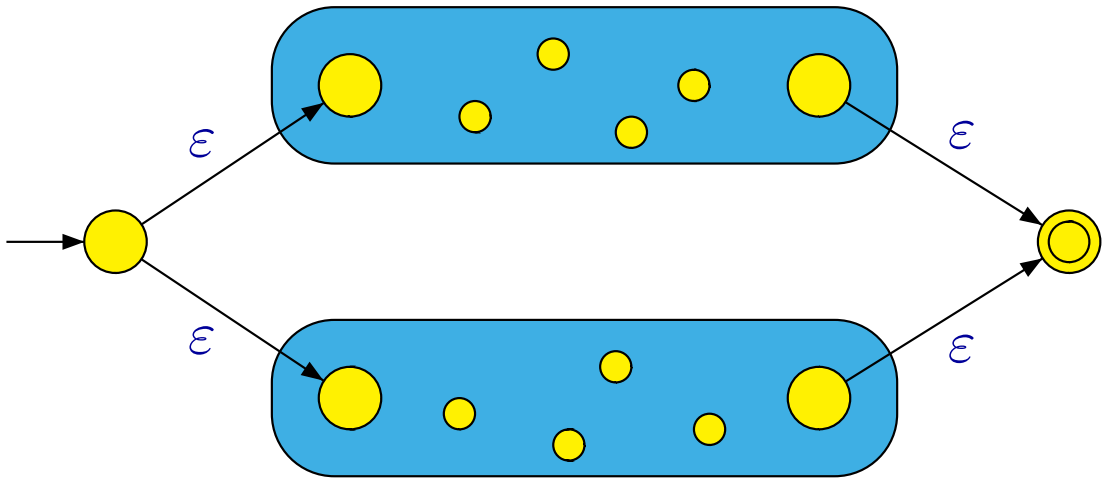


Převod regulárního výrazu na konečný automat

Automaty pro elementární výrazy:

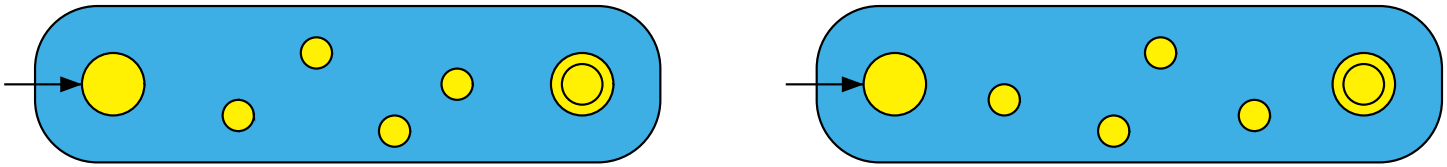


Konstrukce pro sjednocení:



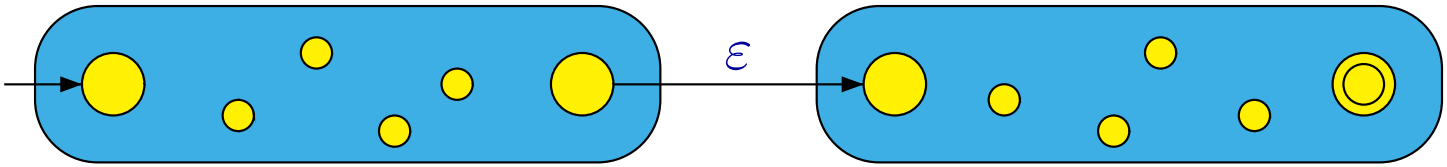
Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:



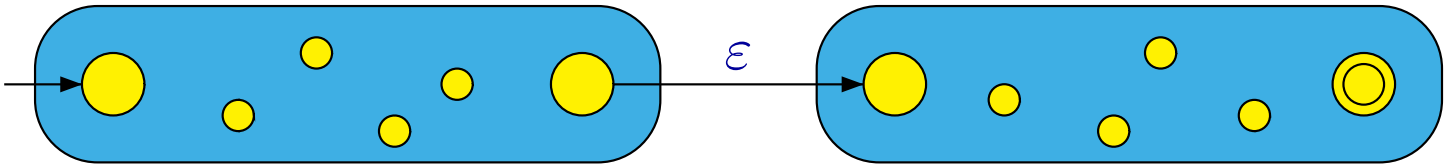
Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:

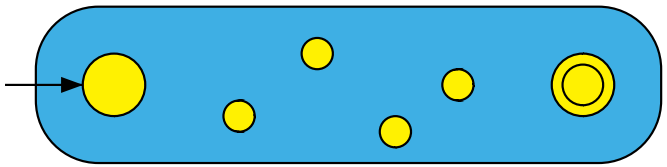


Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:

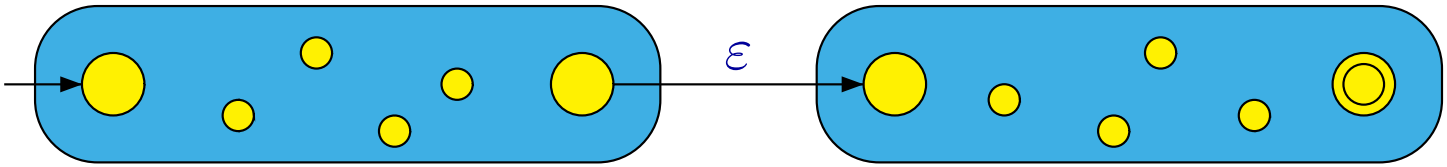


Konstrukce pro iteraci:

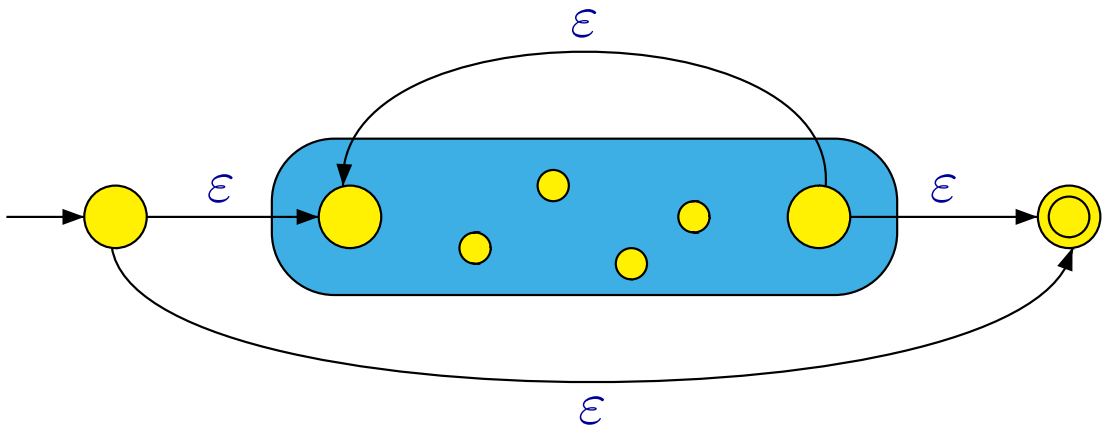


Převod regulárního výrazu na konečný automat

Konstrukce pro zřetězení:



Konstrukce pro iteraci:

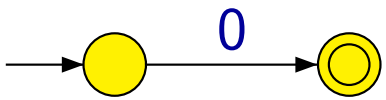


Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((0 + 1) \cdot 1)^*$:

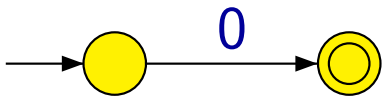
Převod regulárního výrazu na konečný automat

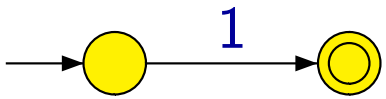
Příklad: Konstrukce automatu pro výraz $((0 + 1) \cdot 1)^*$:



Převod regulárního výrazu na konečný automat

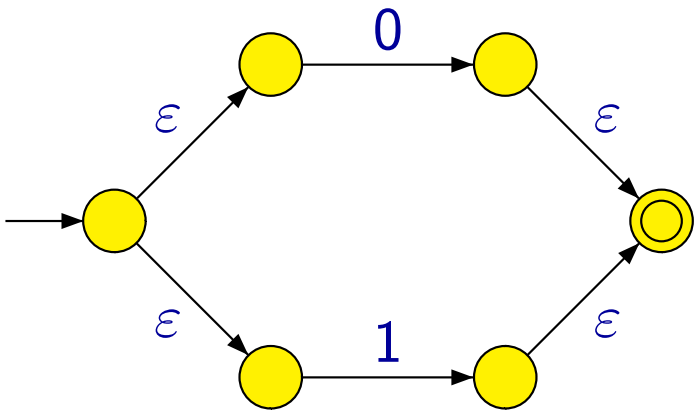
Příklad: Konstrukce automatu pro výraz $((0 + 1) \cdot 1)^*$:





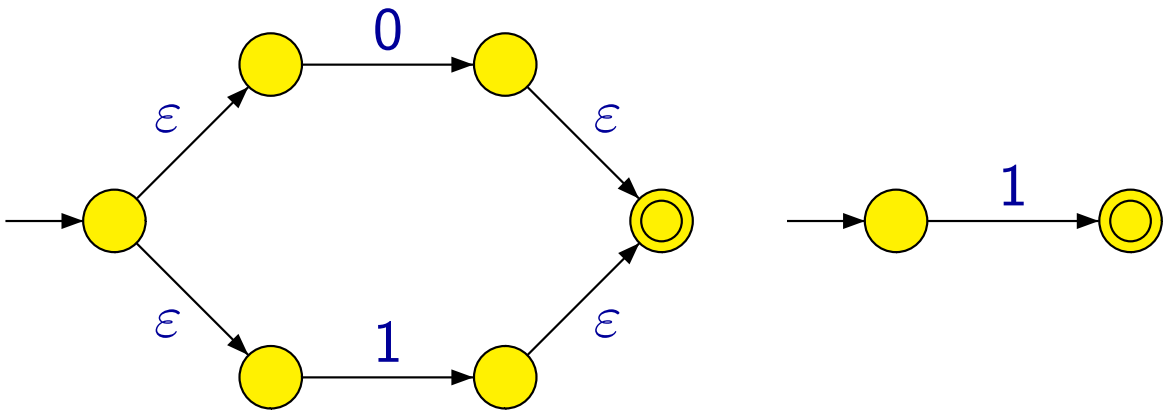
Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((0+1)\cdot 1)^*$:



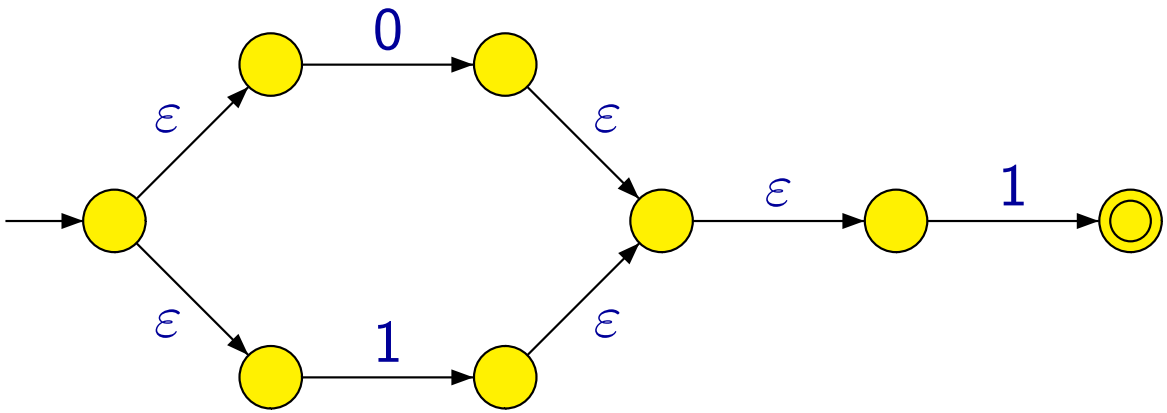
Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((0 + 1) \cdot 1)^*$:



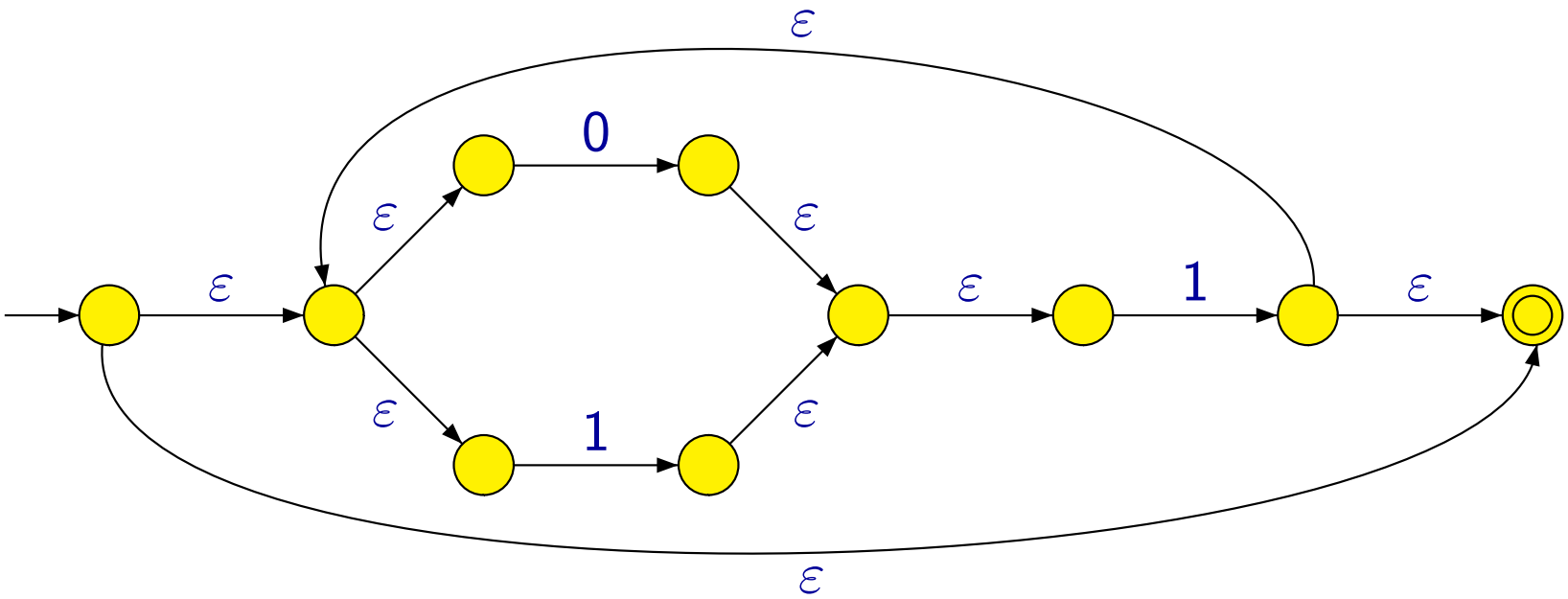
Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((0+1)\cdot 1)^*$:



Převod regulárního výrazu na konečný automat

Příklad: Konstrukce automatu pro výraz $((0+1)\cdot 1)^*$:



Převod regulárního výrazu na konečný automat

Pokud se výraz α skládá z n znaků (nepočítáme-li závorky), má výsledný automat:

- nejvýše $2n$ stavů,
- nejvýše $4n$ přechodů.

Poznámka: Převodem ze zobecněného nedeterministického automatu na deterministický však může počet stavů vzrůst exponenciálně, tj. výsledný automat pak může mít až $2^{2n} = 4^n$ stavů.

Převod konečného automatu na regulární výraz

Tvrzení

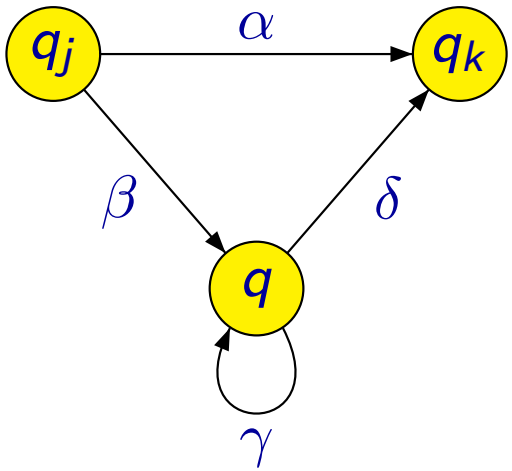
Každý regulární jazyk je možné popsat nějakým regulárním výrazem.

Důkaz: Stačí ukázat, jak pro libovolný konečný automat A zkonstruovat regulární výraz α takový, že $[\alpha] = L(A)$.

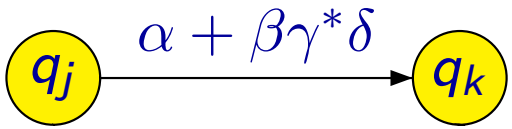
- A upravíme tak, aby měl právě jeden počáteční a právě jeden koncový stav.
- Budeme postupně odebírat jednotlivé stavy.
- Přechody budou označeny regulárními výrazy.
- Zbude automat se dvěma stavy – počátečním a koncovým, a jedním přechodem ohodnoceným výsledným regulárním výrazem.

Převod konečného automatu na regulární výraz

Hlavní myšlenka: Při odstraňování stavu q nahradit pro každou dvojici zbylých stavů q_j, q_k cestu z q_j do q_k vedoucí přes q.

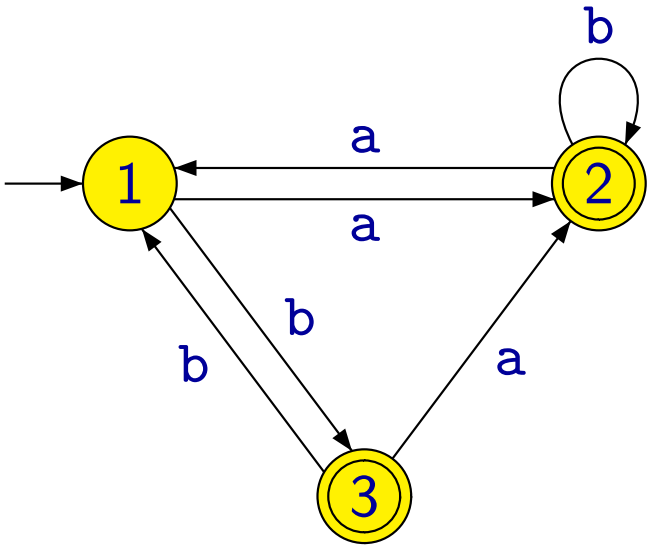


Po odstranění stavu q:



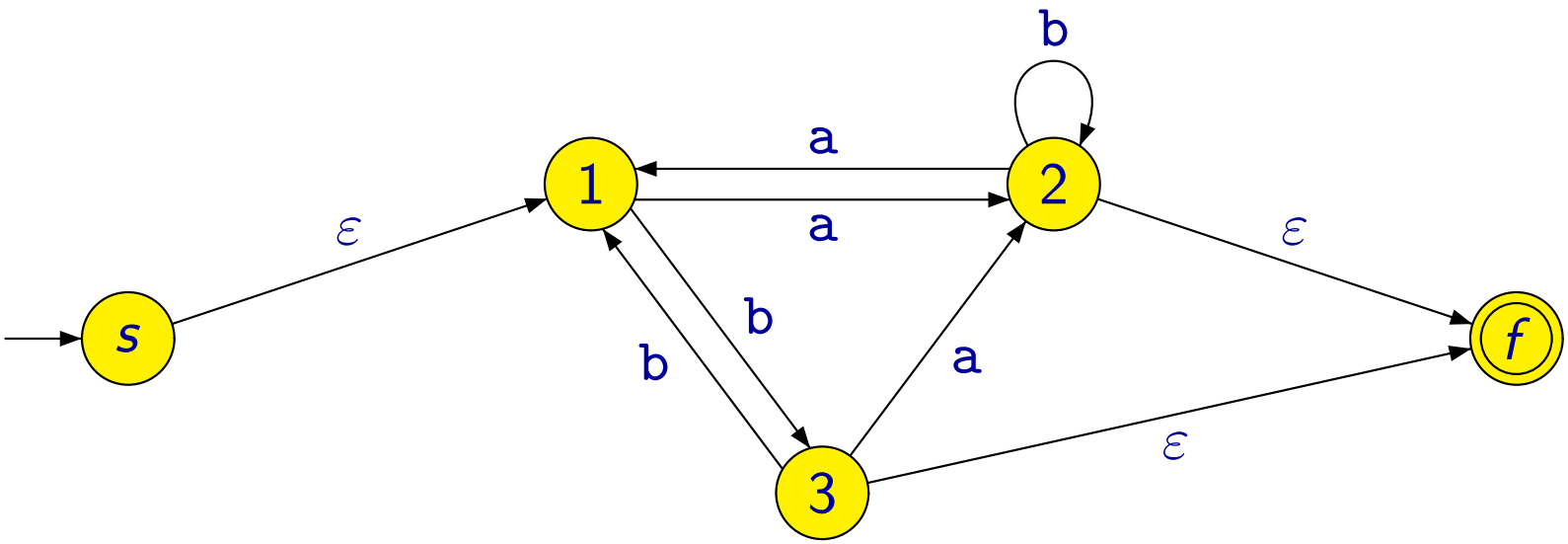
Převod konečného automatu na regulární výraz

Příklad:



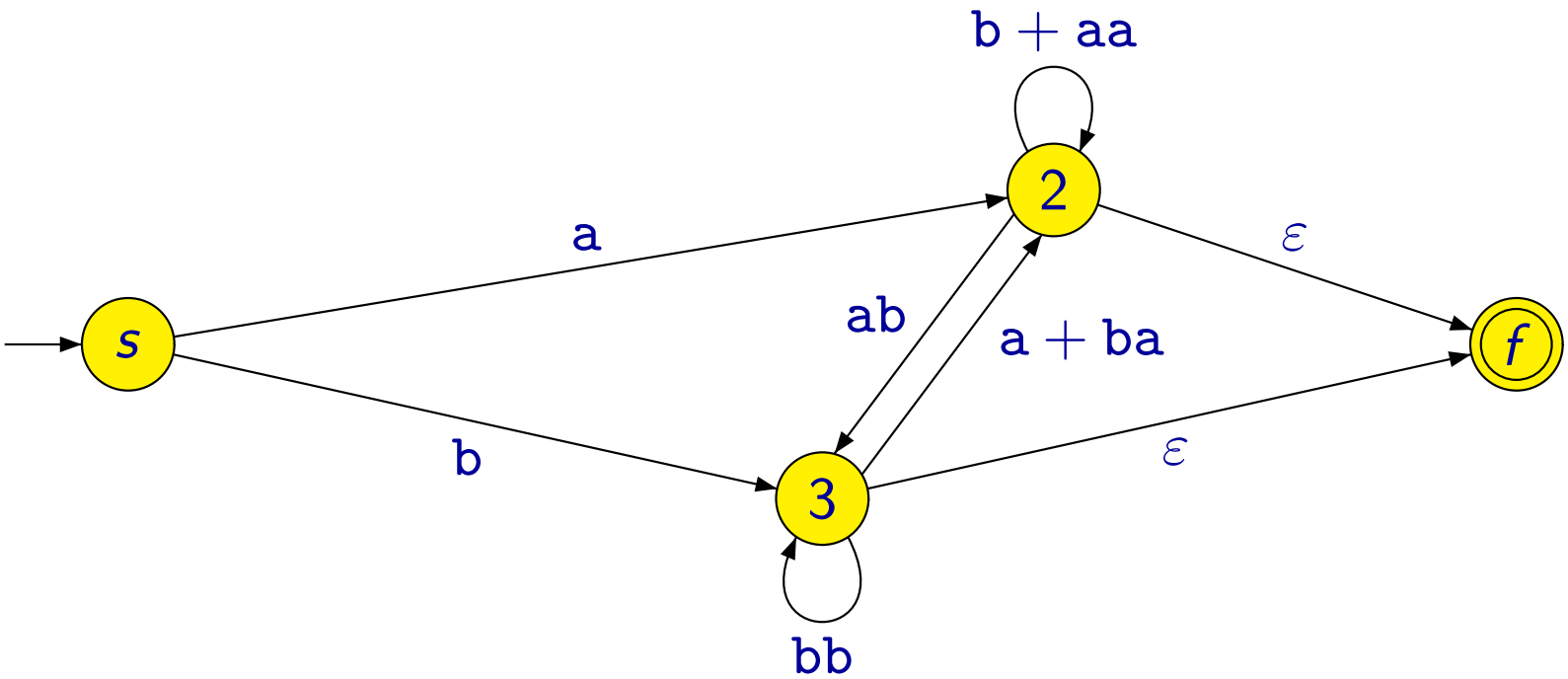
Převod konečného automatu na regulární výraz

Příklad:



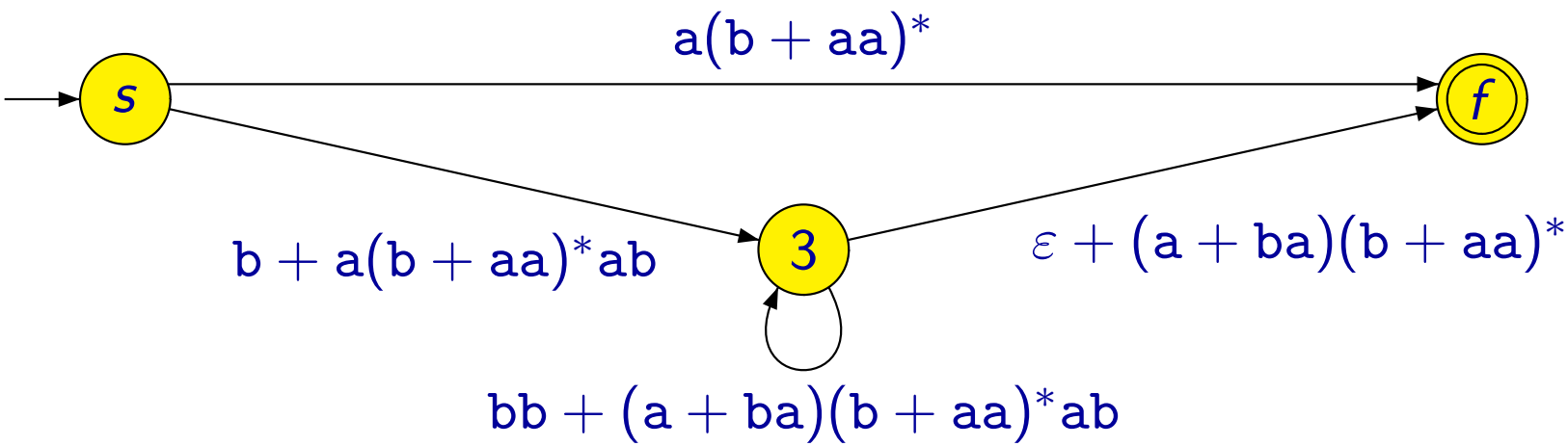
Převod konečného automatu na regulární výraz

Příklad:



Převod konečného automatu na regulární výraz

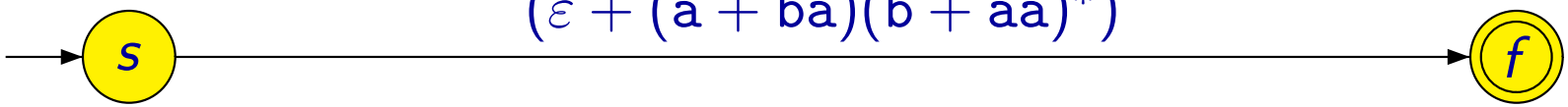
Příklad:



Převod konečného automatu na regulární výraz

Příklad:

$$\mathtt{a}(\mathtt{b}+\mathtt{aa})^*+$$
$$(\mathtt{b}+\mathtt{a}(\mathtt{b}+\mathtt{aa})^*\mathtt{ab})$$
$$(\mathtt{bb}+(\mathtt{a}+\mathtt{ba})(\mathtt{b}+\mathtt{aa})^*\mathtt{ab})^*$$
$$(\varepsilon+(\mathtt{a}+\mathtt{ba})(\mathtt{b}+\mathtt{aa})^*)$$

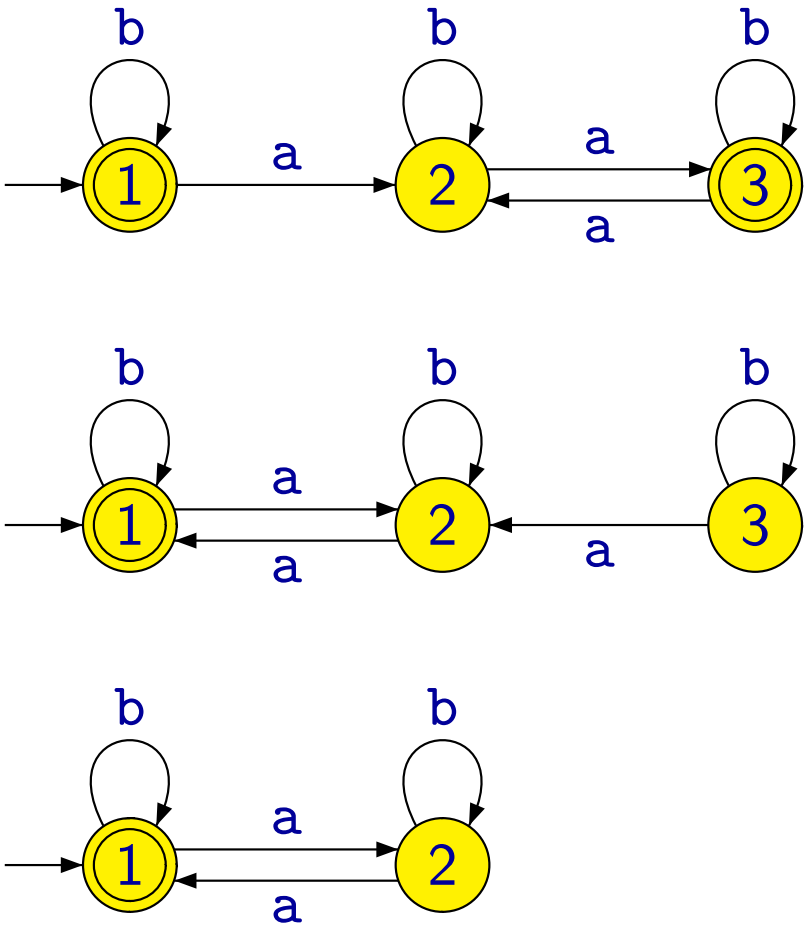


Ekvivalence konečných automatů a regulárních výrazů

Věta

Jazyk je regulární právě tehdy, když je ho možné popsat regulárním výrazem.

Ekvivalence automatů



- Všechny 3 automaty přijímají jazyk všech slov se sudým počtem *a*-ček
- Nejvýhodnější je pro nás poslední z nich – má nejméně stavů

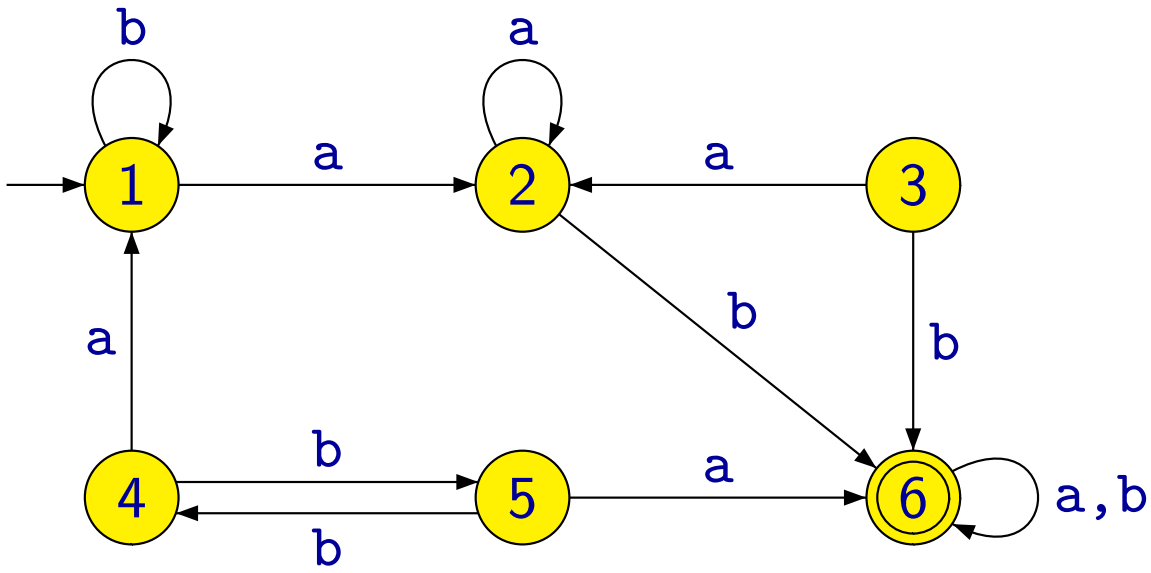
Ekvivalence automatů

Definice

O konečných automatech A_1, A_2 řekneme, že jsou **ekvivalentní**, jestliže $L(A_1) = L(A_2)$.

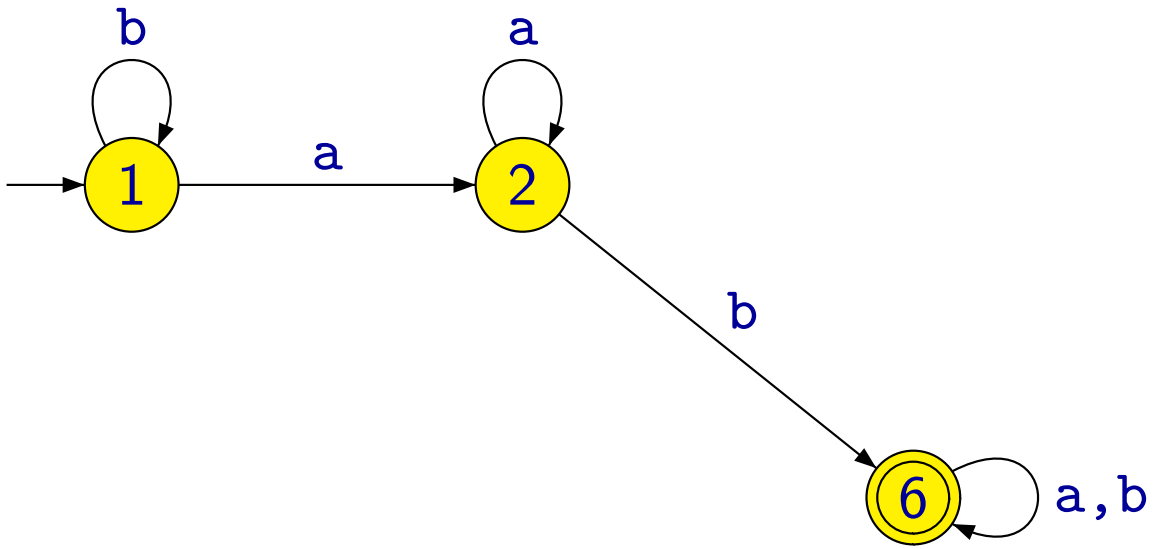
- Existuje nějaký vhodný algoritmus zjišťující, jestli jsou dva automaty ekvivalentní?

Nedosažitelné stavy automatu



- Automat přijímá jazyk $L = \{w \in a, b^* \mid w \text{ obsahuje podslovo } ab\}$
- Pro žádnou posloupnost vstupních symbolů se automat nedostane do stavů 3, 4, nebo 5

Nedosažitelné stavy automatu



- Automat přijímá jazyk $L = \{w \in a, b^* \mid w \text{ obsahuje podslovo } ab\}$
- Pro žádnou posloupnost vstupních symbolů se automat nedostane do stavů 3, 4, nebo 5
- Pokud tyto stavy odstraníme, pořád automat přijímá stejný jazyk $L = \{w \in a, b^* \mid w \text{ obsahuje podslovo } ab\}$

Nedosažitelné stavy automatu

Definice

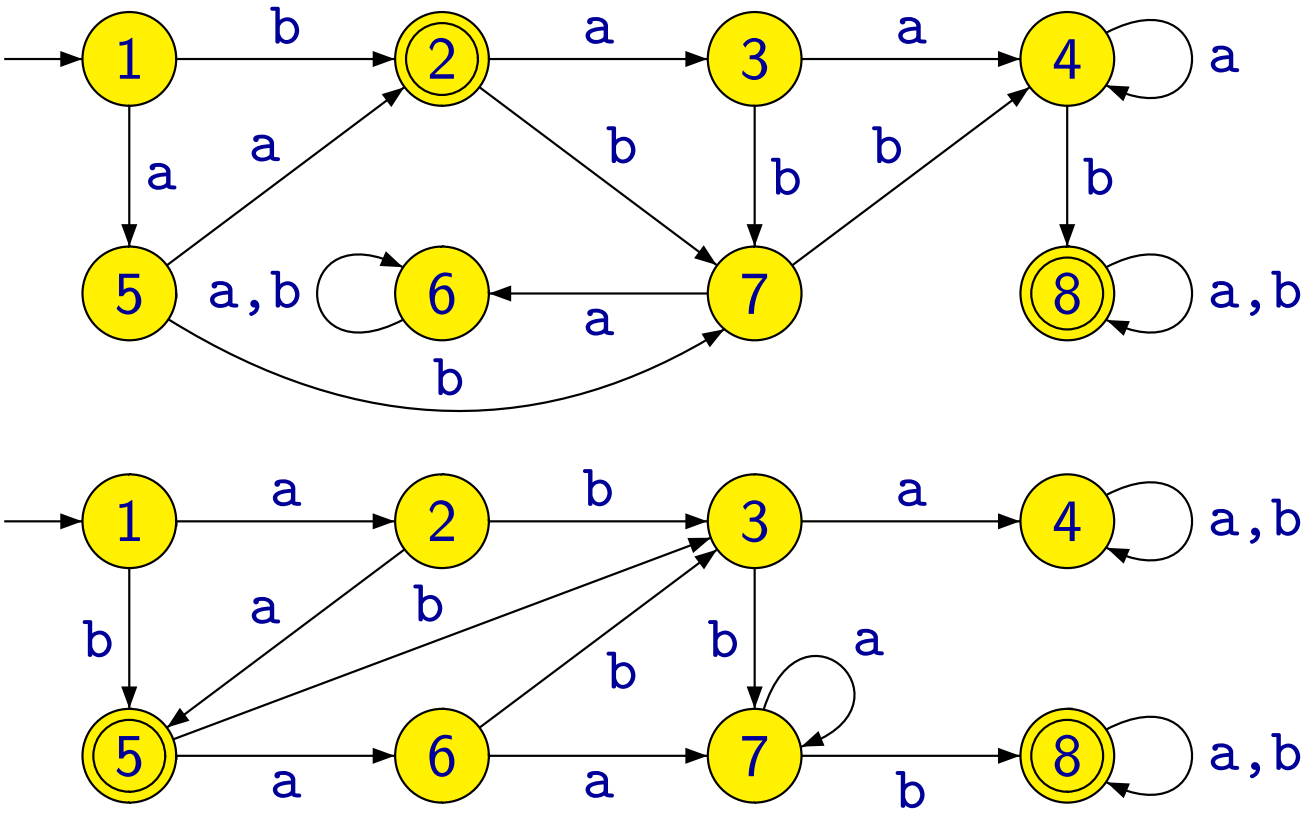
Stav q deterministického konečného automatu $\mathcal{A}$ je **dosažitelný slovem** w, pokud se výpočet $\mathcal{A}$ po přečtení celého slova w zastaví ve stavu q.

Definice

Stav q deterministického konečného automatu $\mathcal{A}$ je **dosažitelný** pokud existuje nějaké slovo, kterým je stav dosažitelný. V opačném případě stav nazýváme **nedosažitelný**.

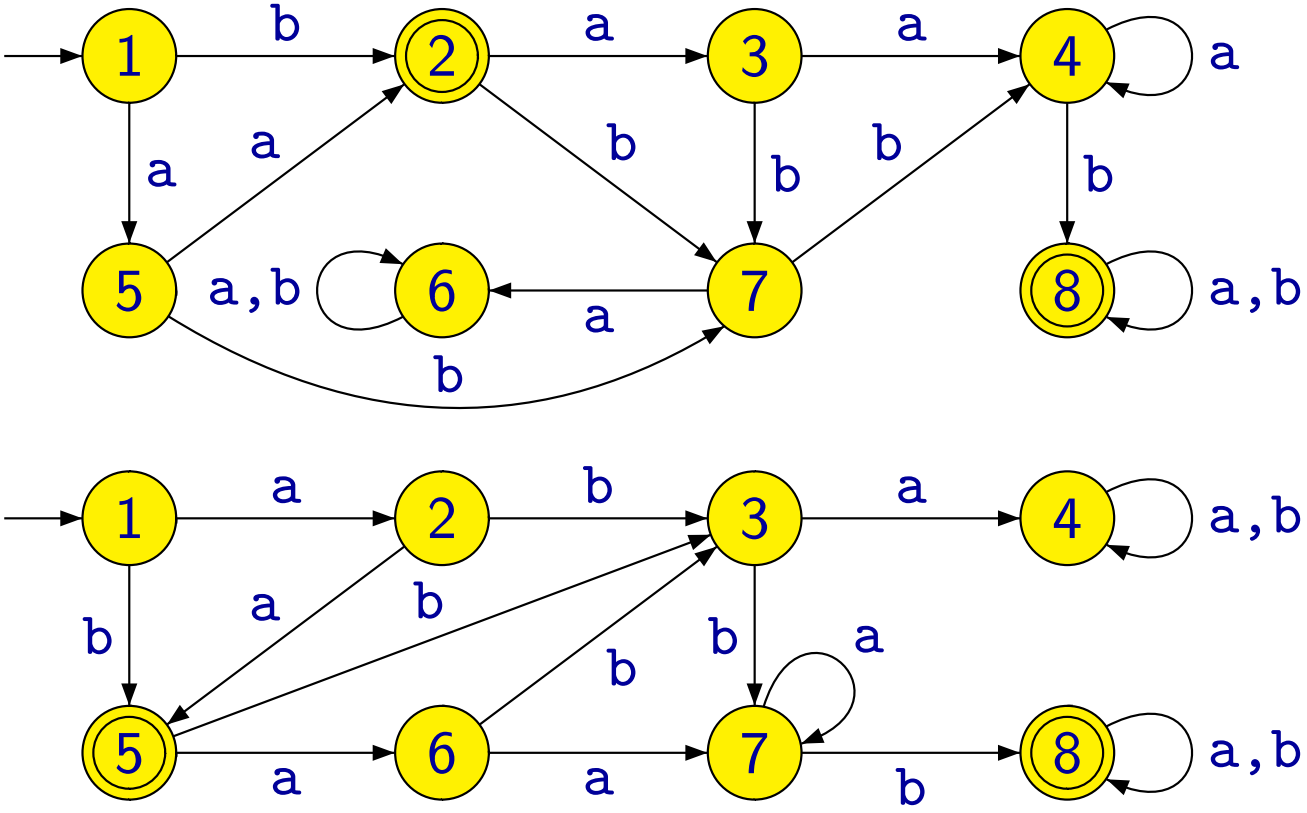
- Do nedosažitelných stavů nevede v grafu automatu žádná orientovaná cesta z počátečního stavu
- Nedosažitelné stavy můžeme z automatu odstranit (spolu se všemi přechody vedoucími z nich). Jazyk přijímaný automatem se nezmění.

Normovaný tvar automatu



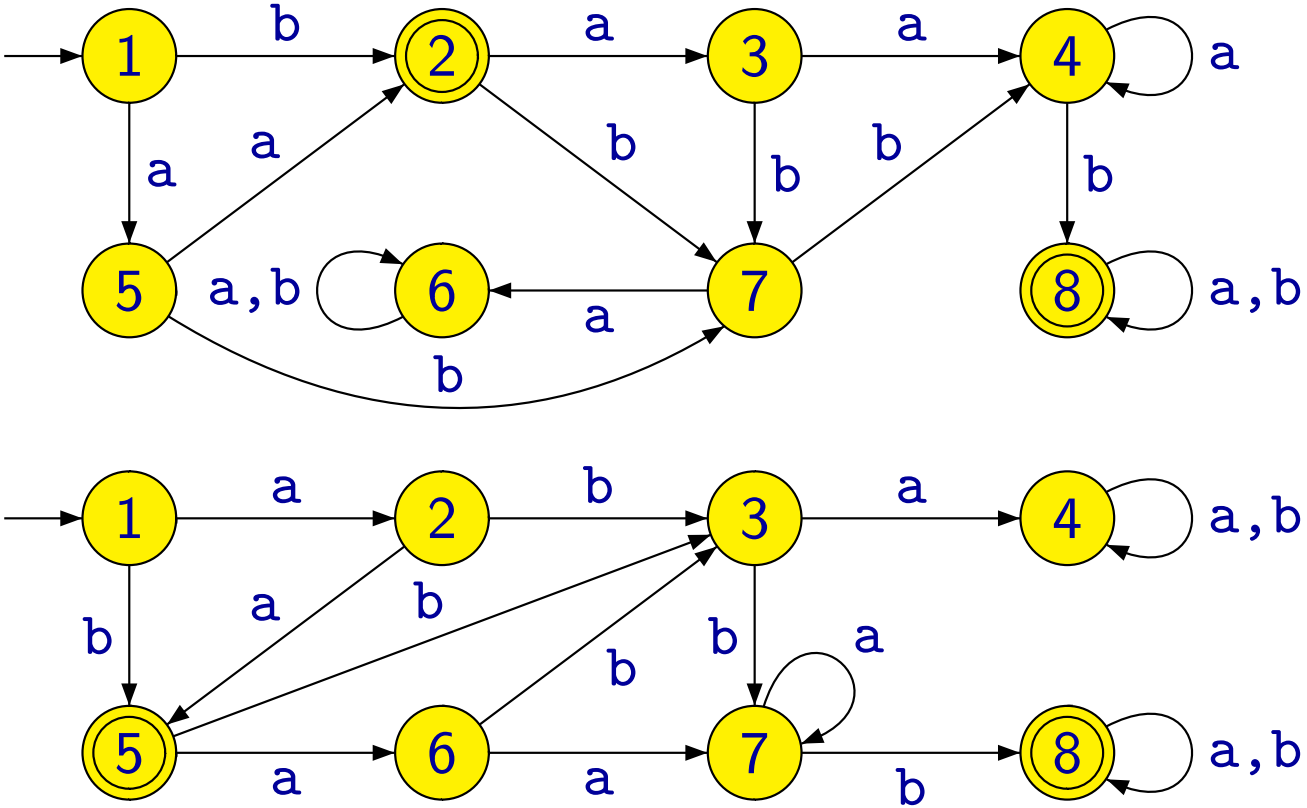
$\mathcal{A}_1$	a	b
$\rightarrow 1$	5	2
$\leftarrow 2$	3	7
3	4	7
4	4	8
5	2	7
6	6	6
7	6	4
$\leftarrow 8$	8	8

Normovaný tvar automatu



$\mathcal{A}_2$	a	b
$\rightarrow 1$	2	5
2	5	3
3	4	7
4	4	4
$\leftarrow 5$	6	3
6	7	3
7	7	8
$\leftarrow 8$	8	8

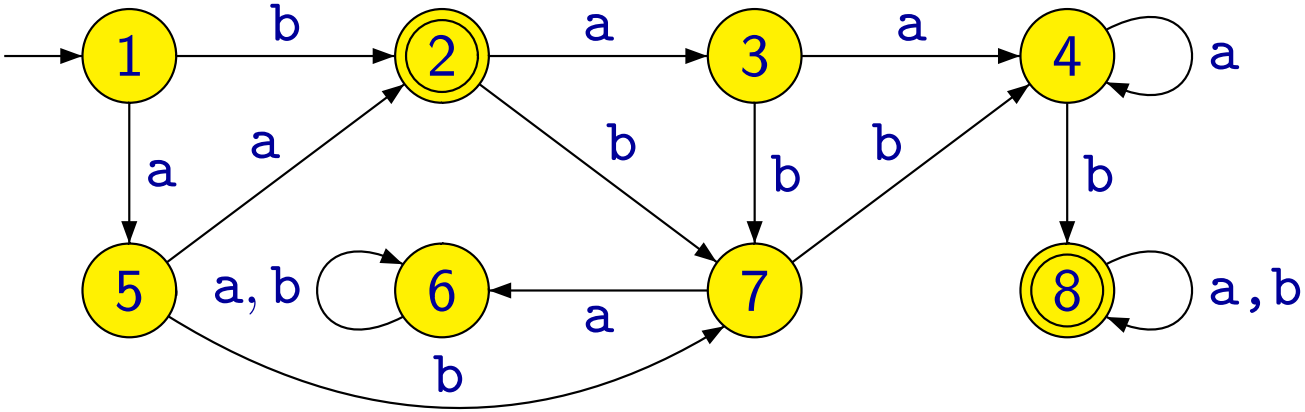
Normovaný tvar automatu



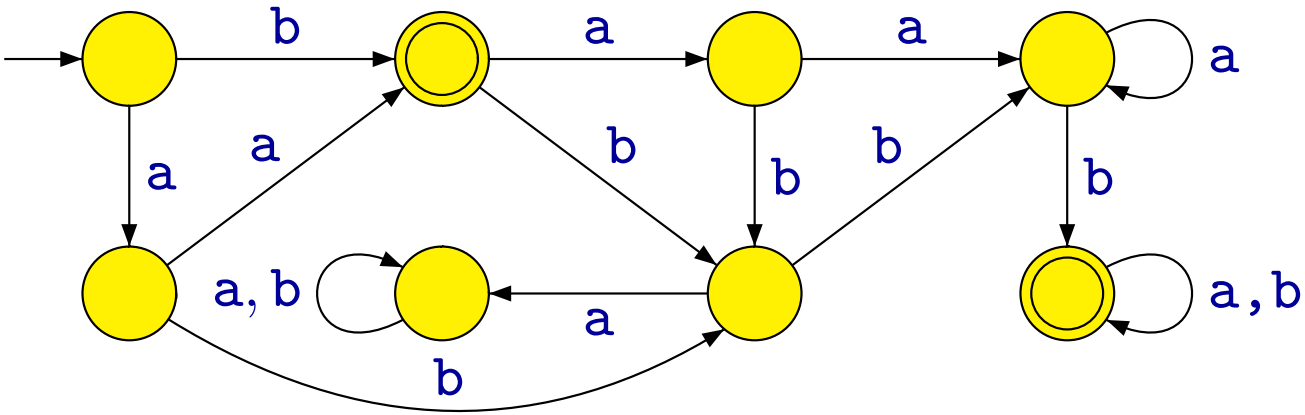
	$\mathcal{A}_2$	a	b
$\rightarrow$	1	2	5
	2	5	3
	3	4	7
	4	4	4
$\leftarrow$	5	6	3
	6	7	3
	7	7	8
$\leftarrow$	8	8	8

- Jak poznáme, že jsou automaty ekvivalentní, když se liší přechodová funkce?
- Automaty na obrázku můžu čísly $1-8$ pojmenovat $8!$ způsoby
- Chtěli bychom jednoznačně vybrat jedno z možných pojmenování

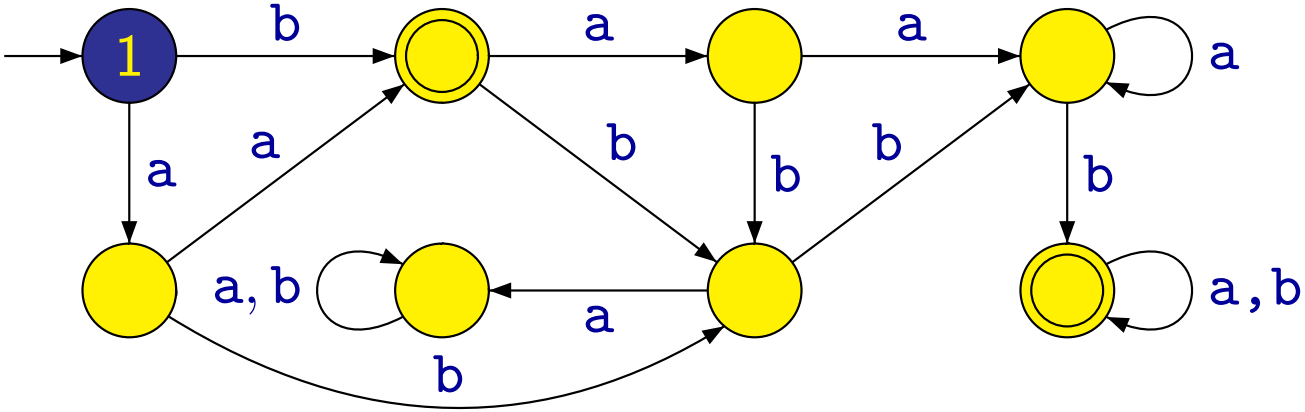
Normovaný tvar automatu



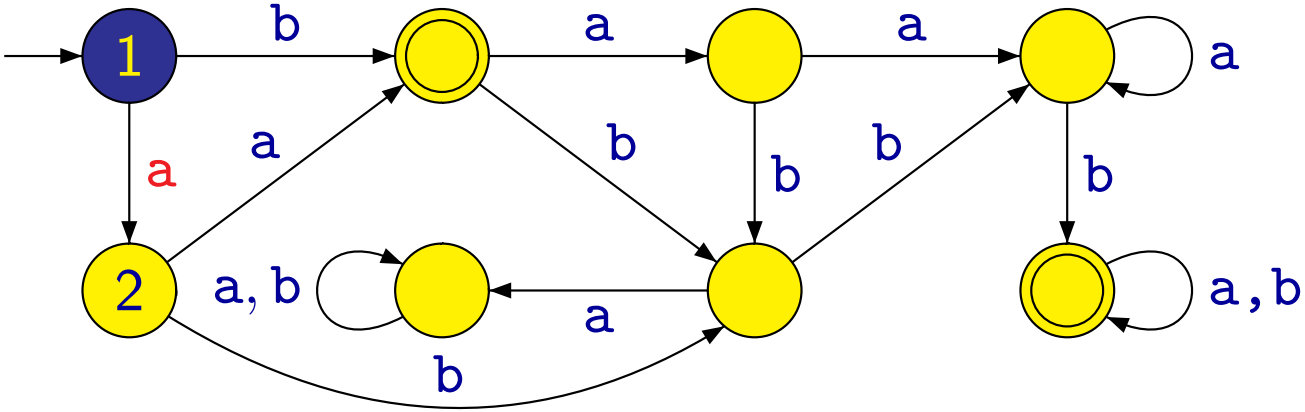
Normovaný tvar automatu



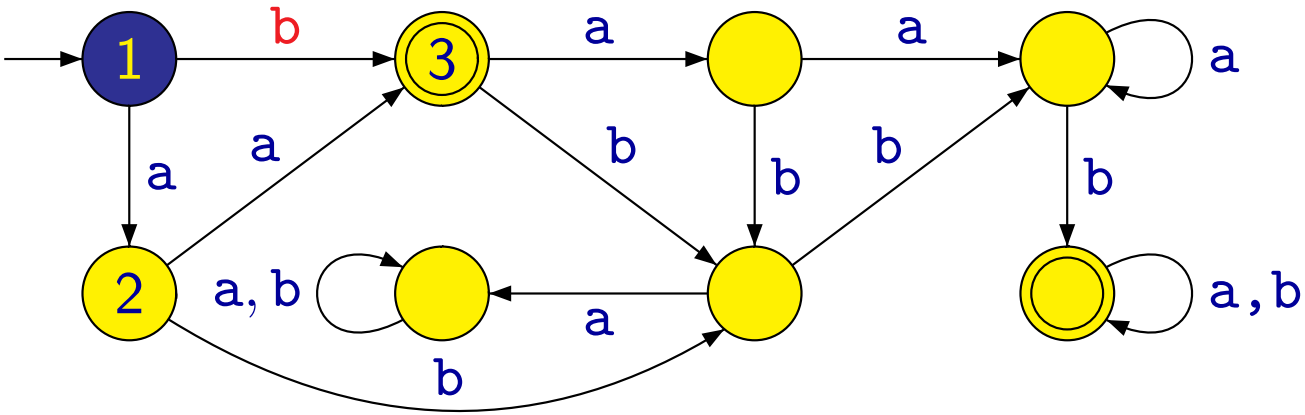
Normovaný tvar automatu



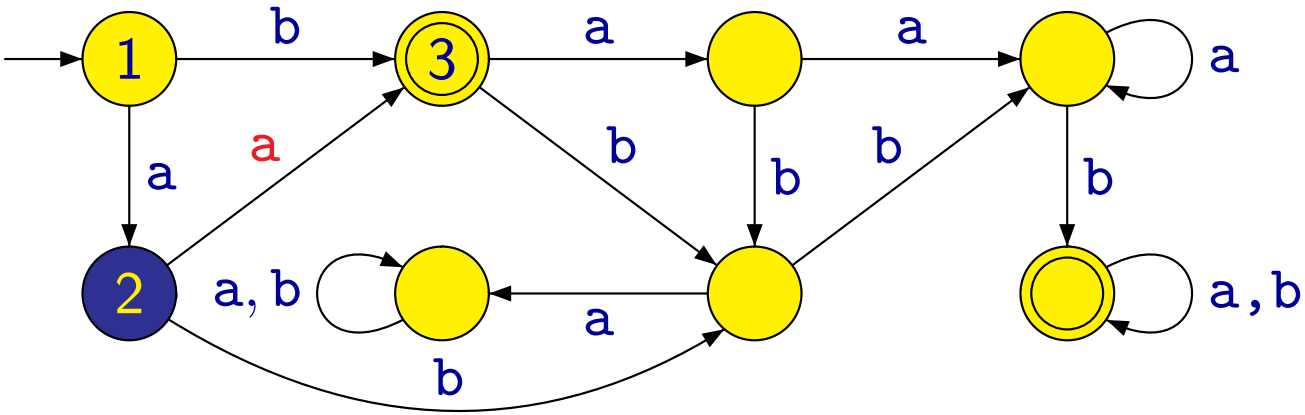
Normovaný tvar automatu



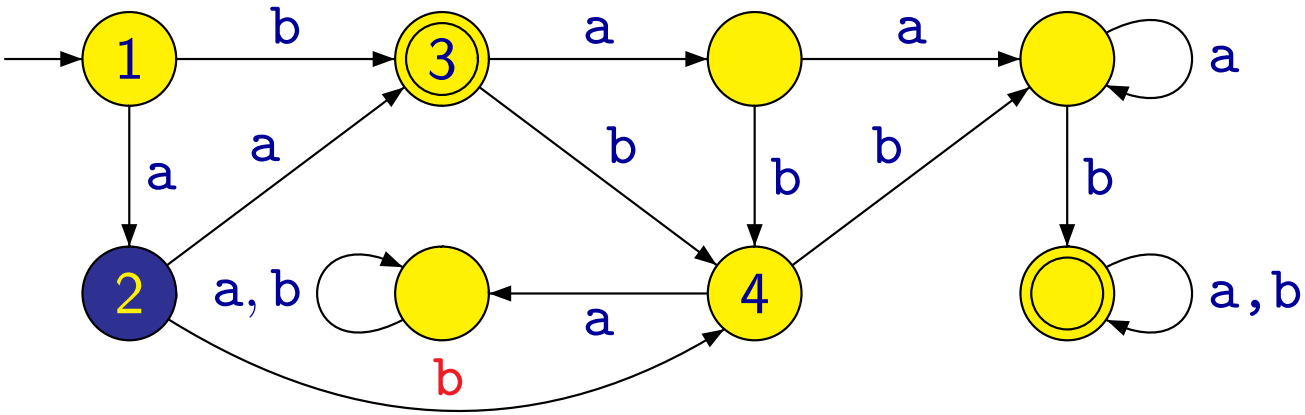
Normovaný tvar automatu



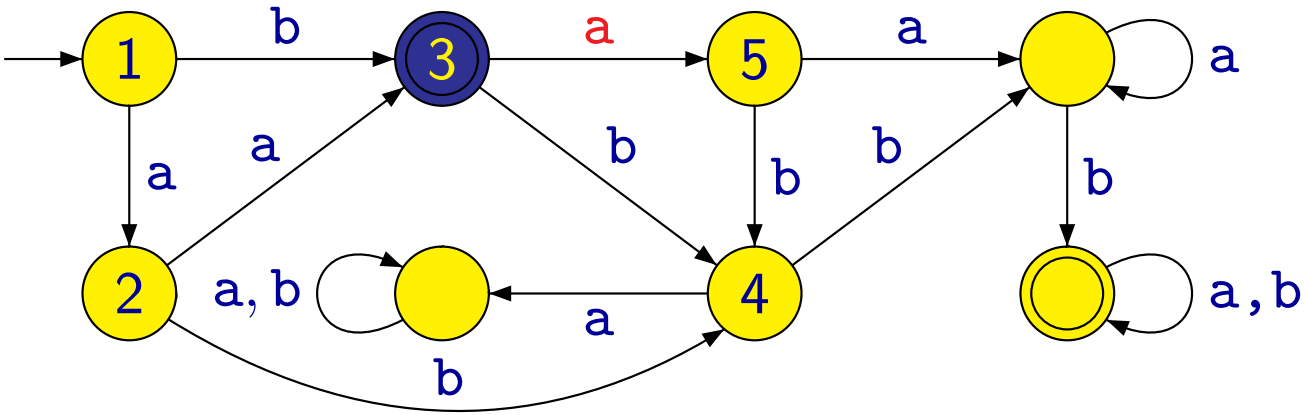
Normovaný tvar automatu



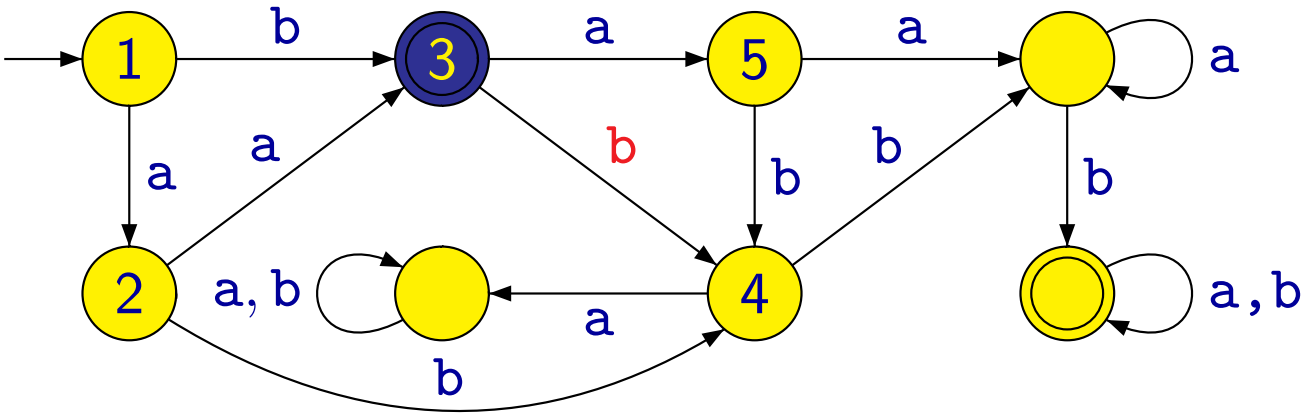
Normovaný tvar automatu



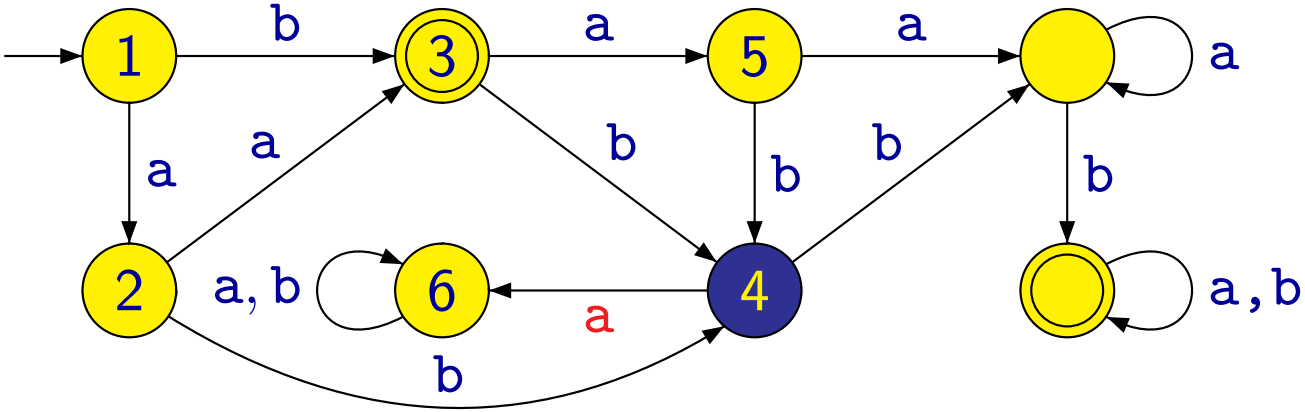
Normovaný tvar automatu



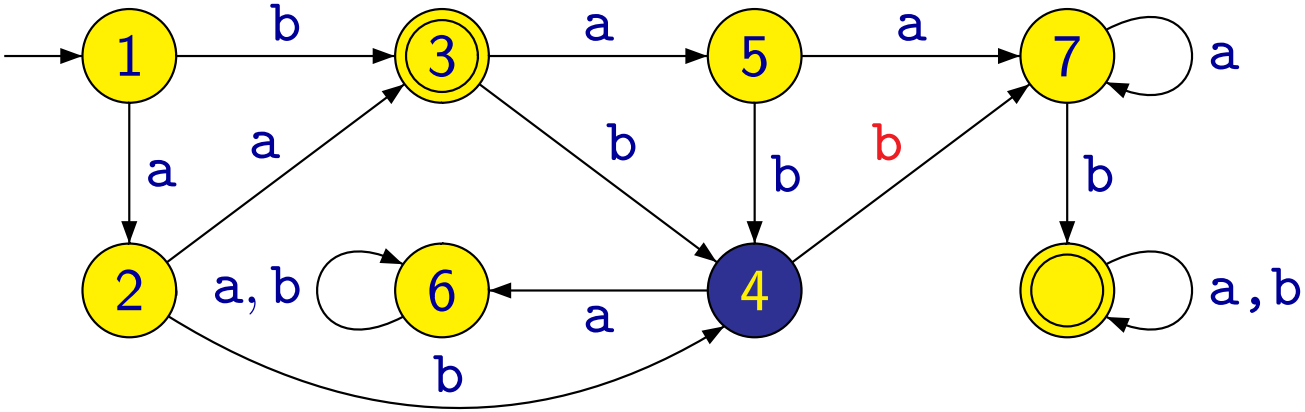
Normovaný tvar automatu



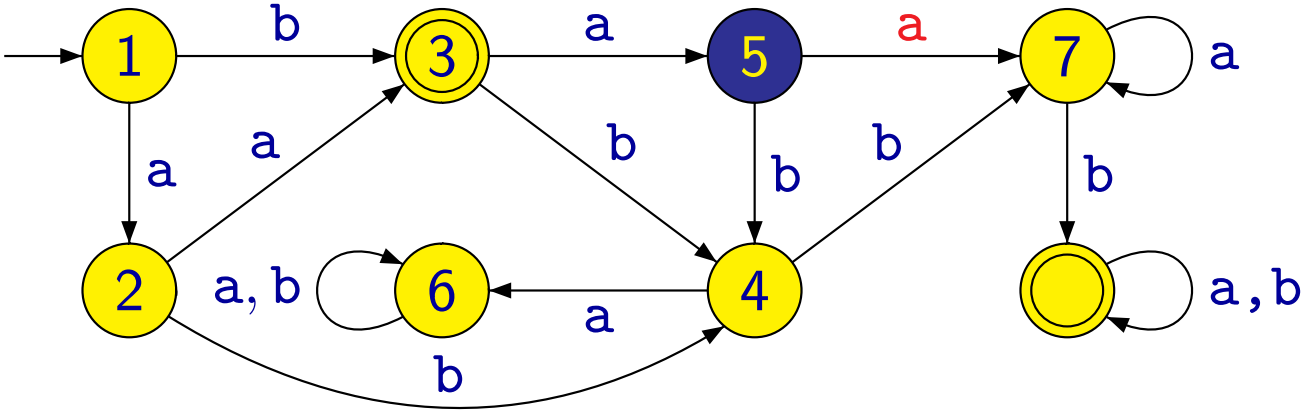
Normovaný tvar automatu



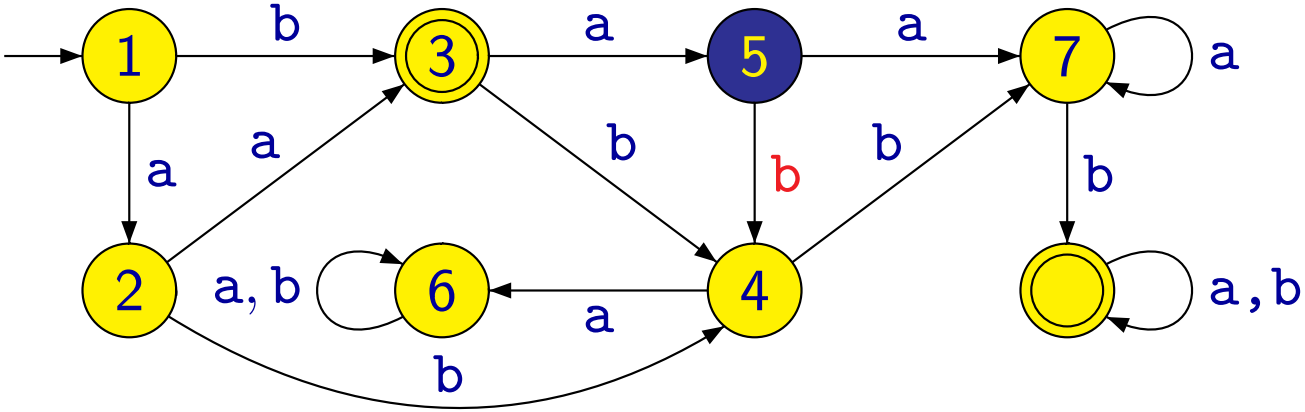
Normovaný tvar automatu



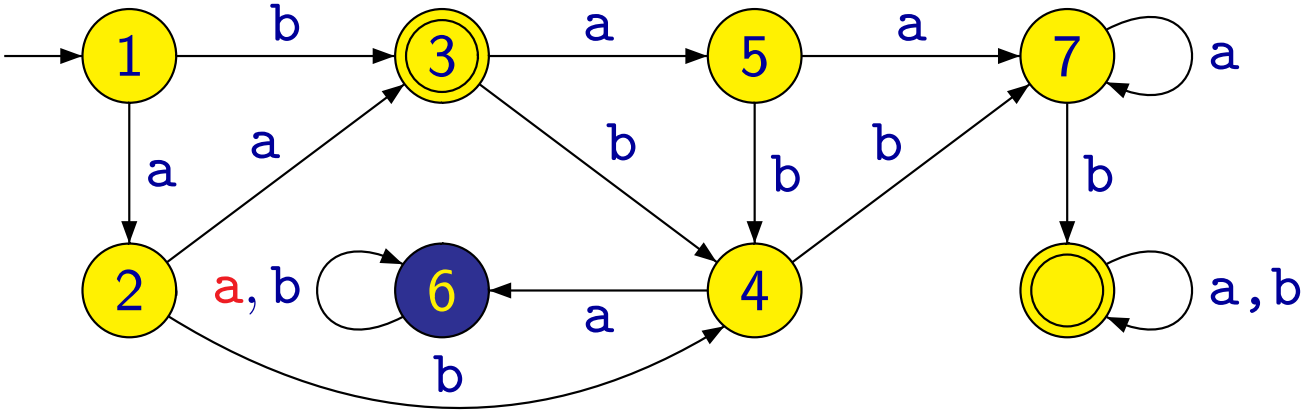
Normovaný tvar automatu



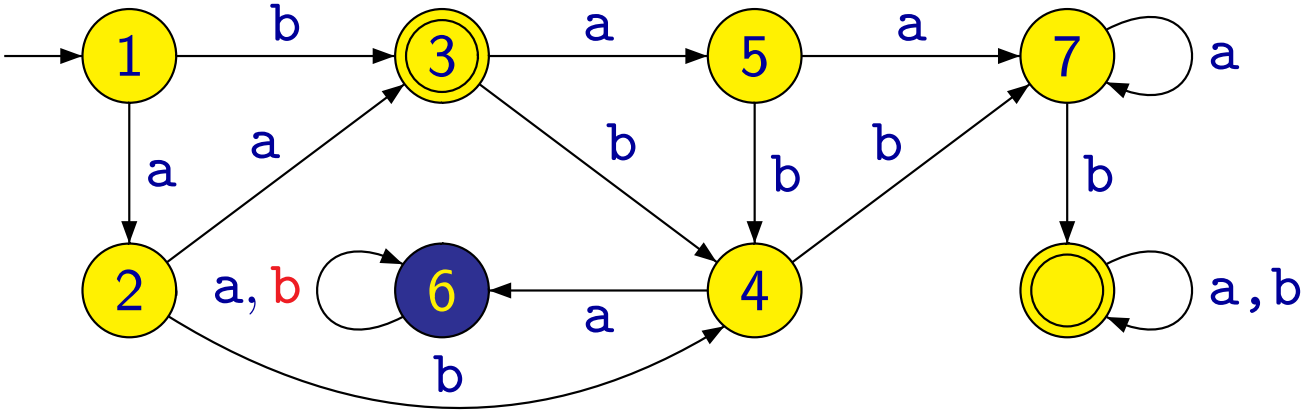
Normovaný tvar automatu



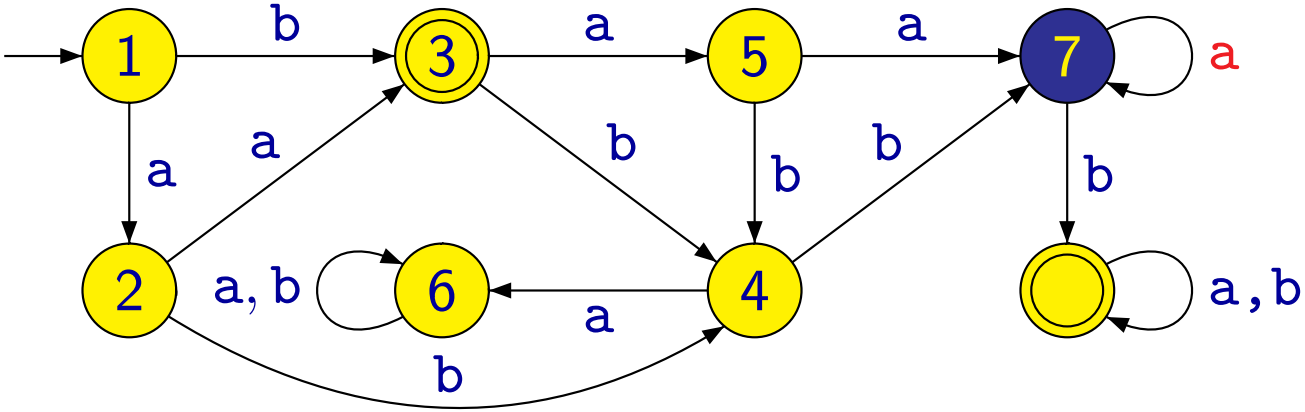
Normovaný tvar automatu



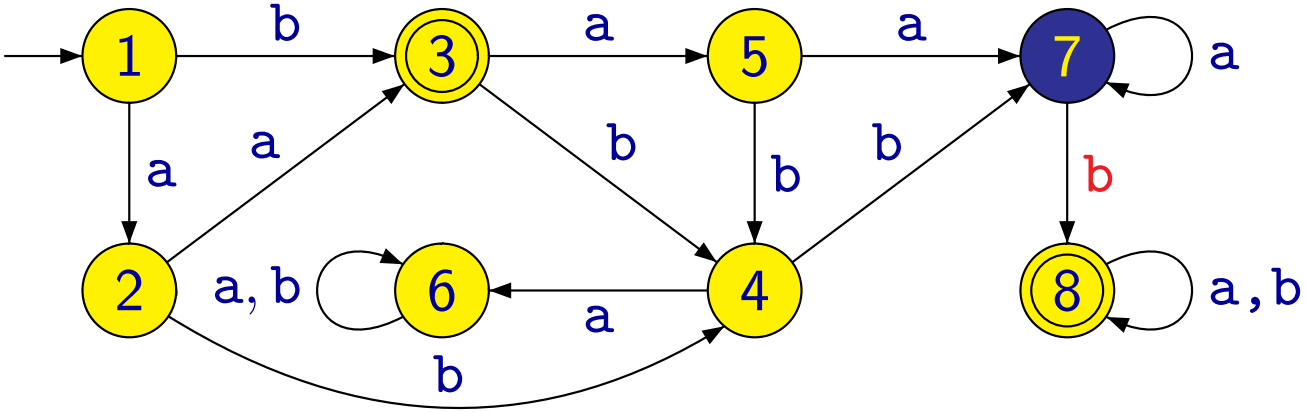
Normovaný tvar automatu



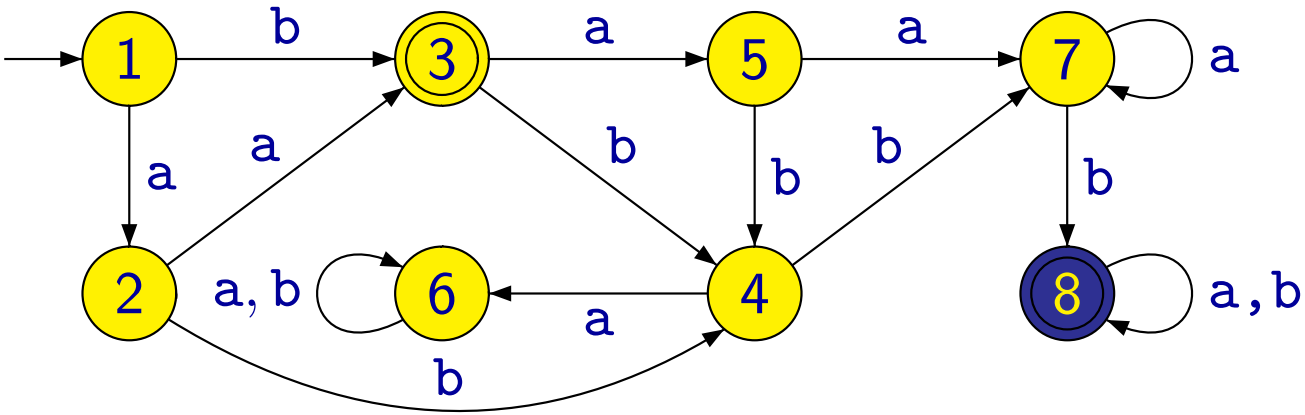
Normovaný tvar automatu



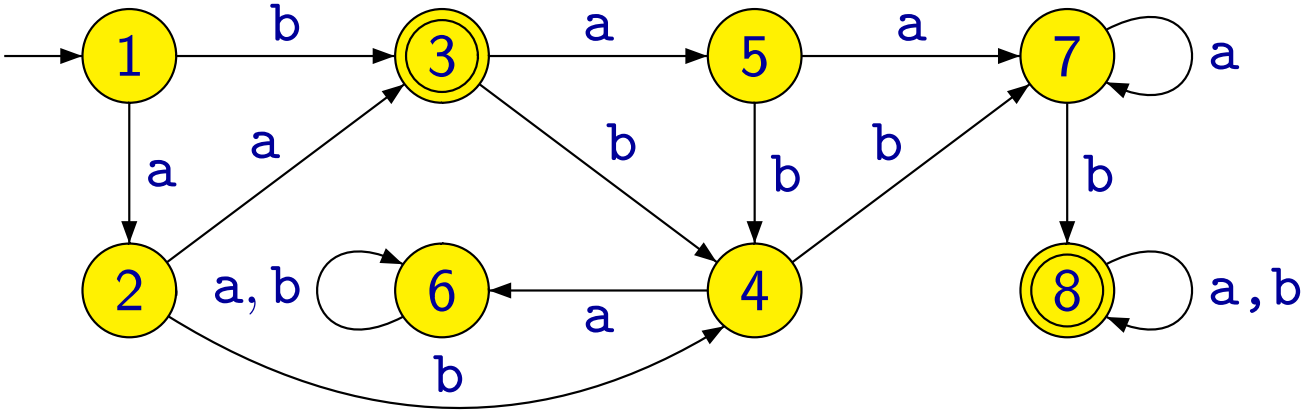
Normovaný tvar automatu



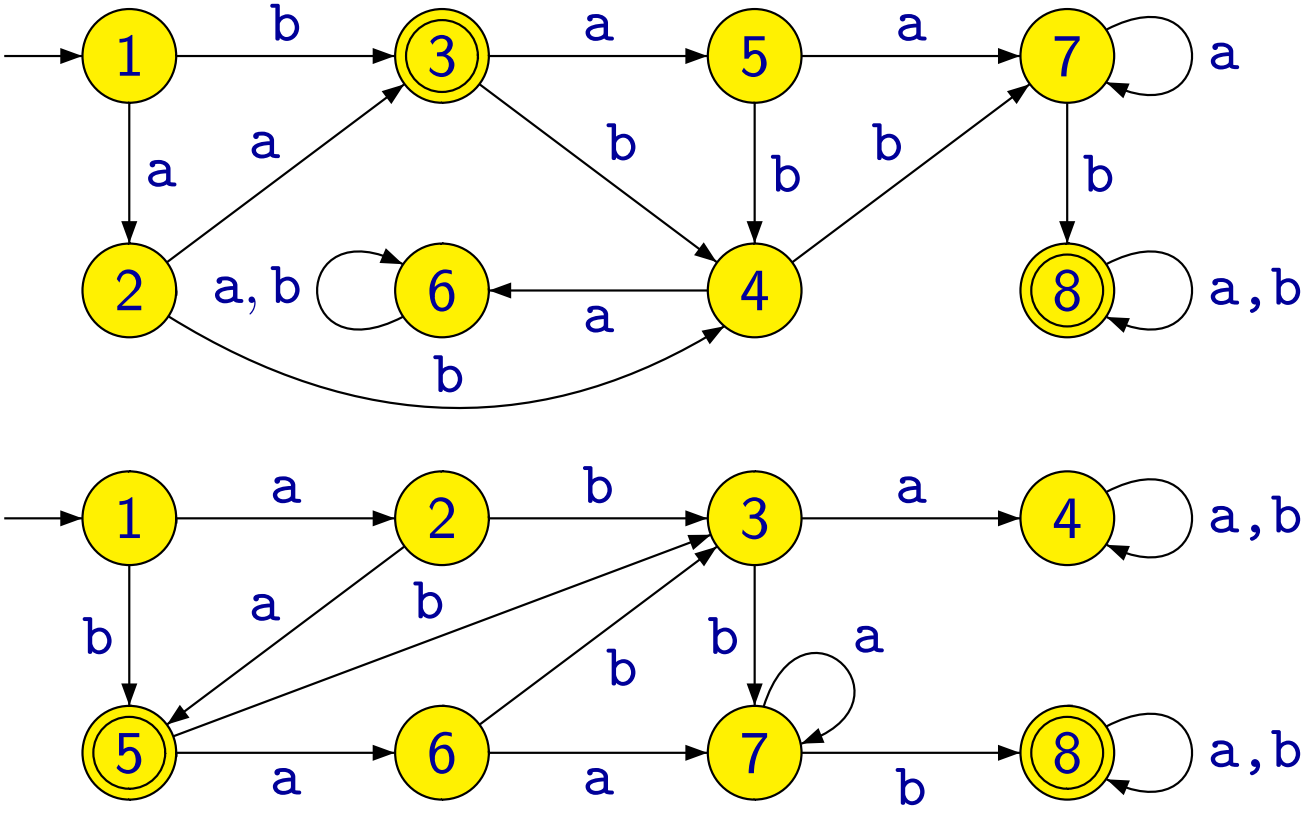
Normovaný tvar automatu



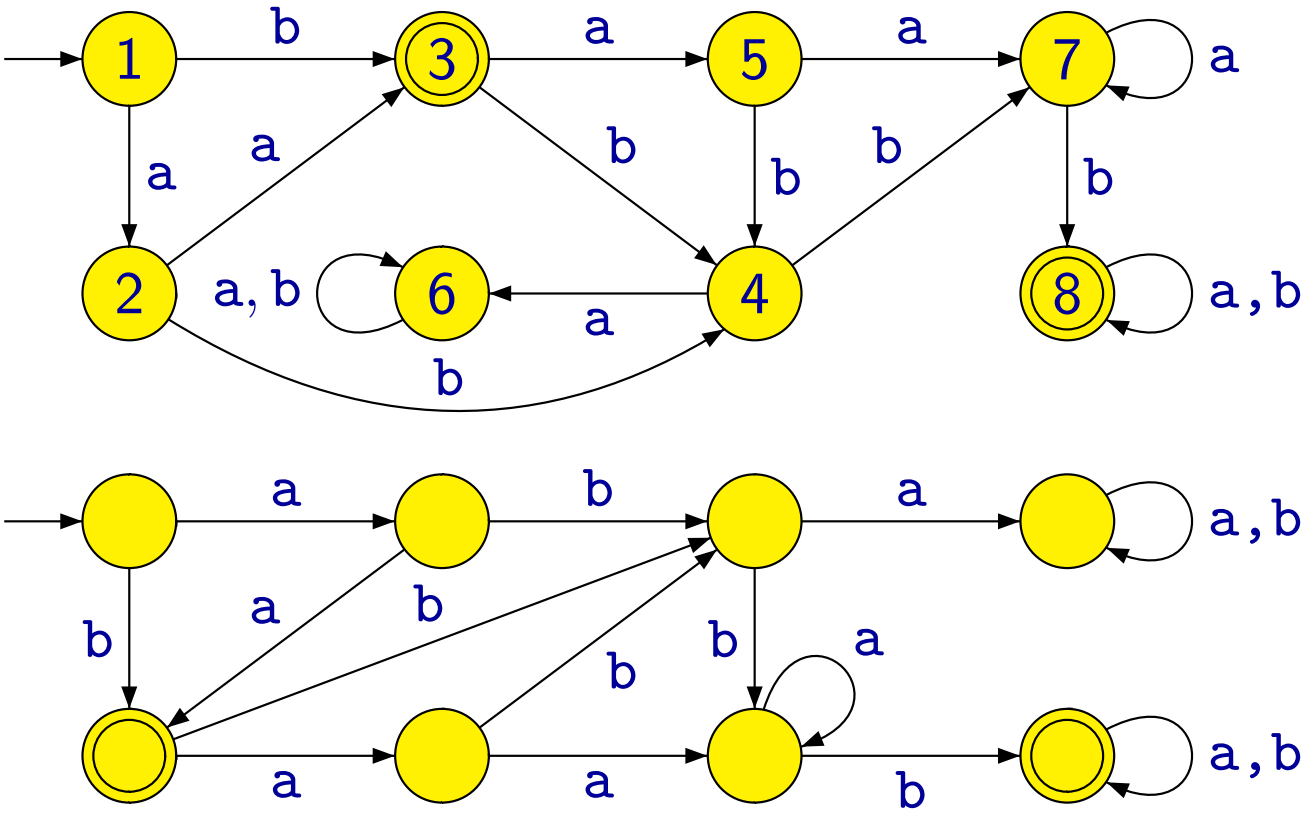
Normovaný tvar automatu



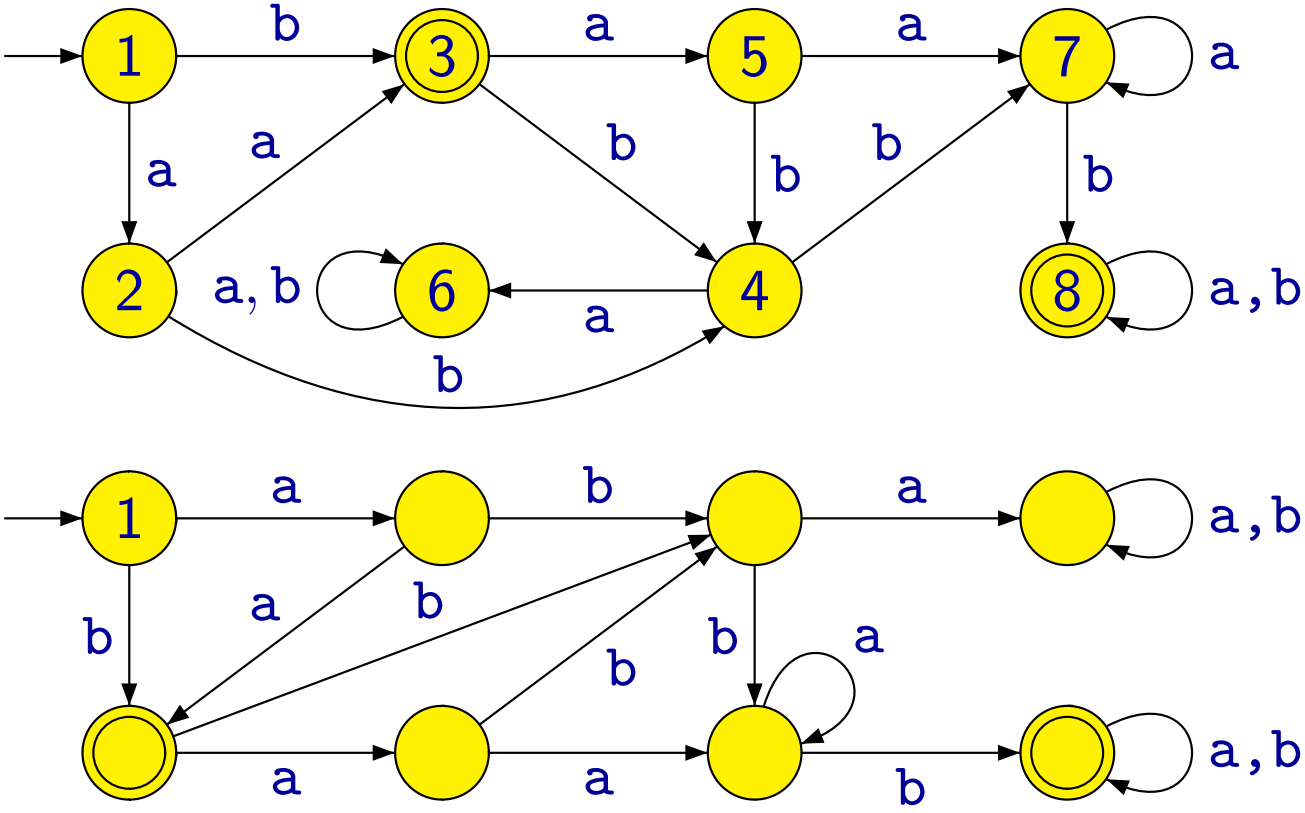
Normovaný tvar automatu



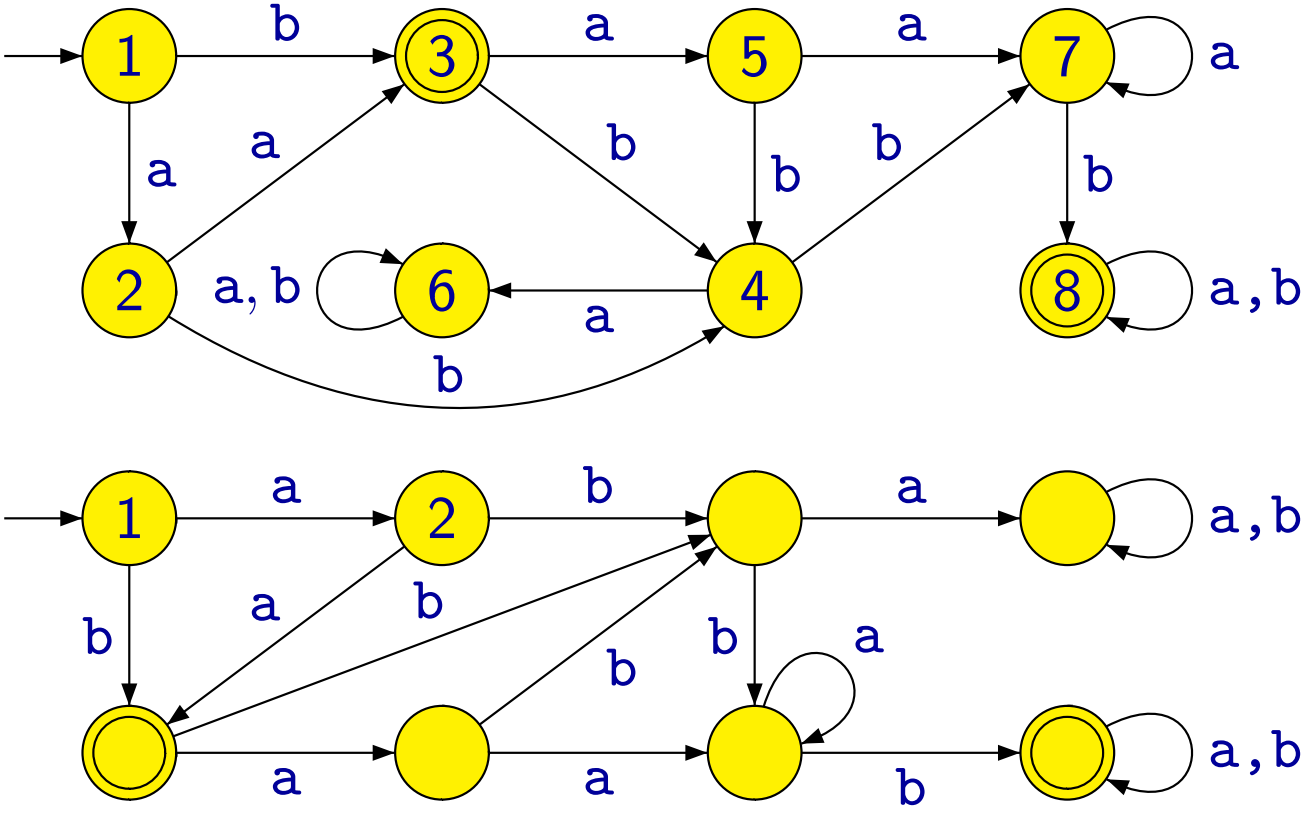
Normovaný tvar automatu



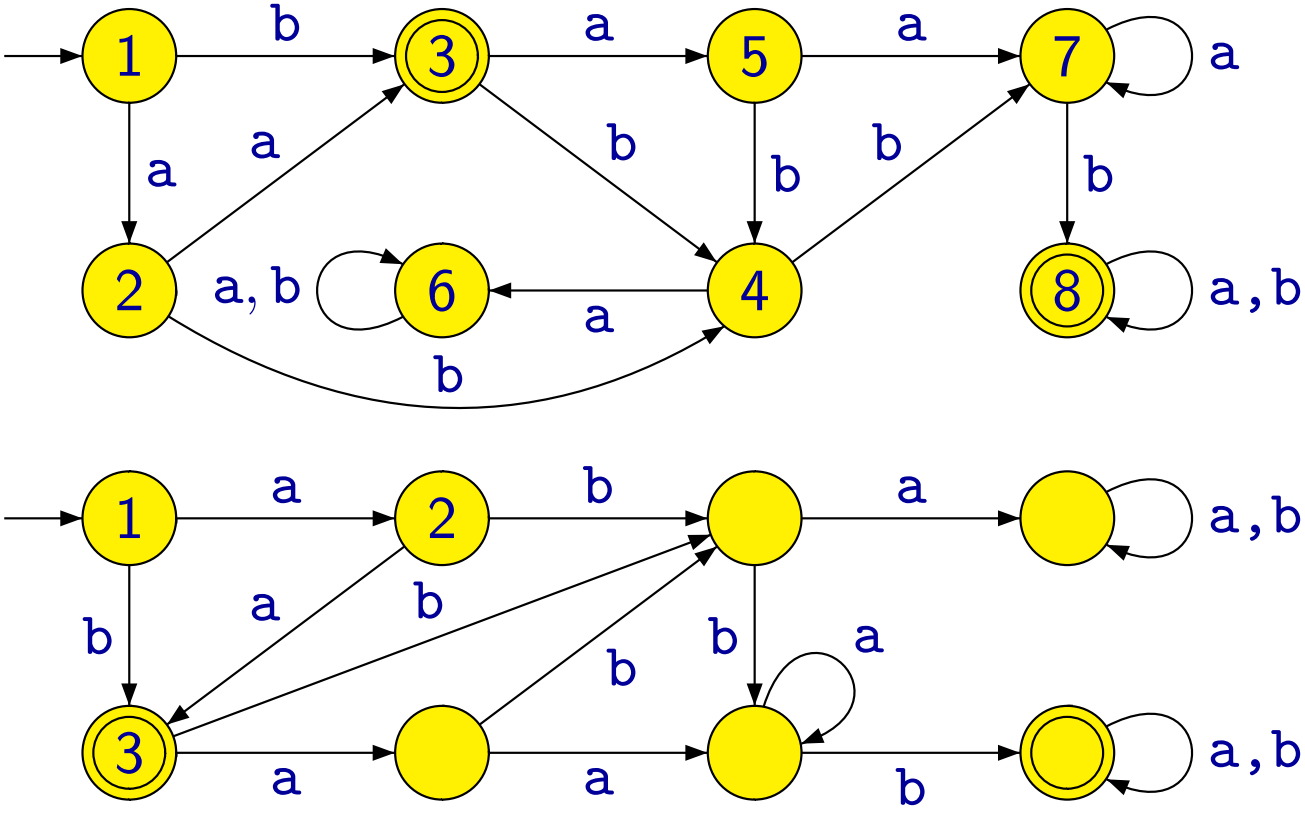
Normovaný tvar automatu



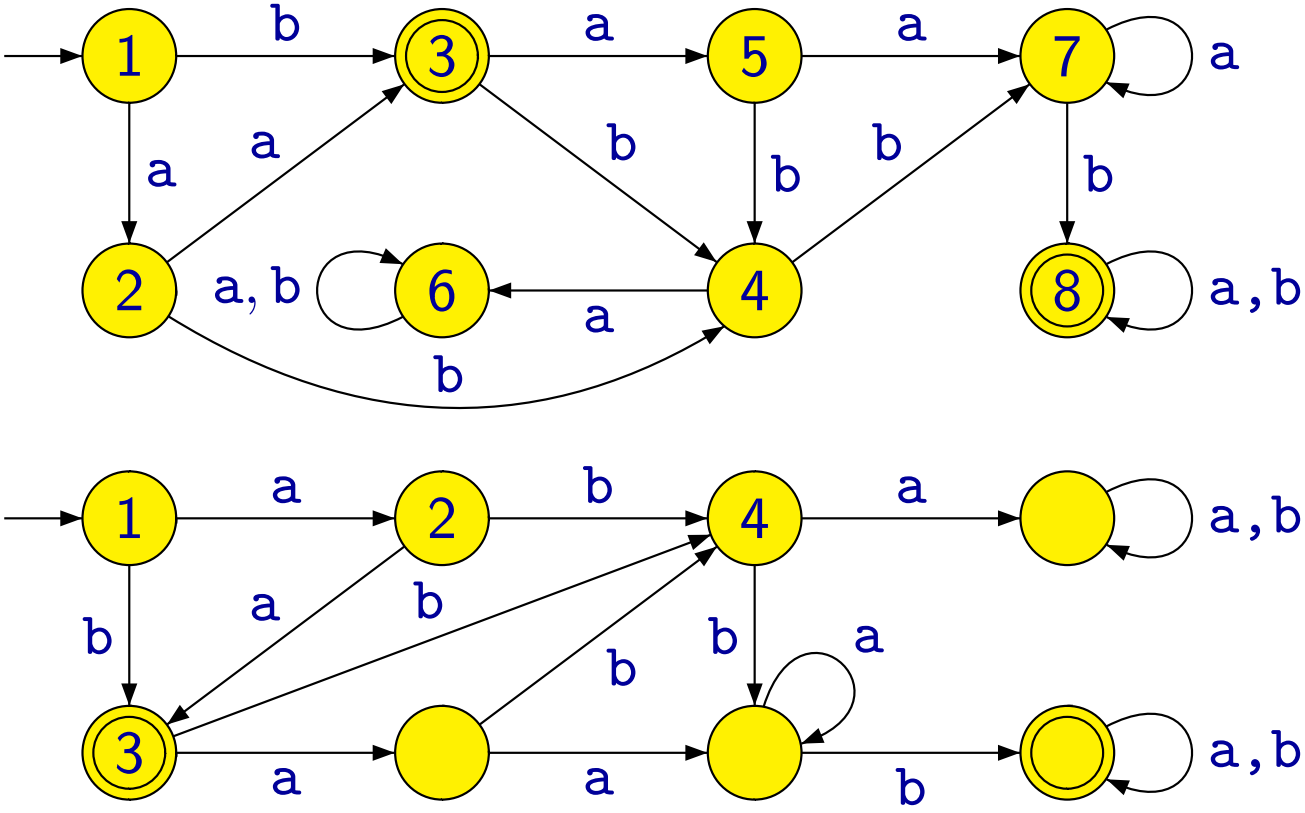
Normovaný tvar automatu



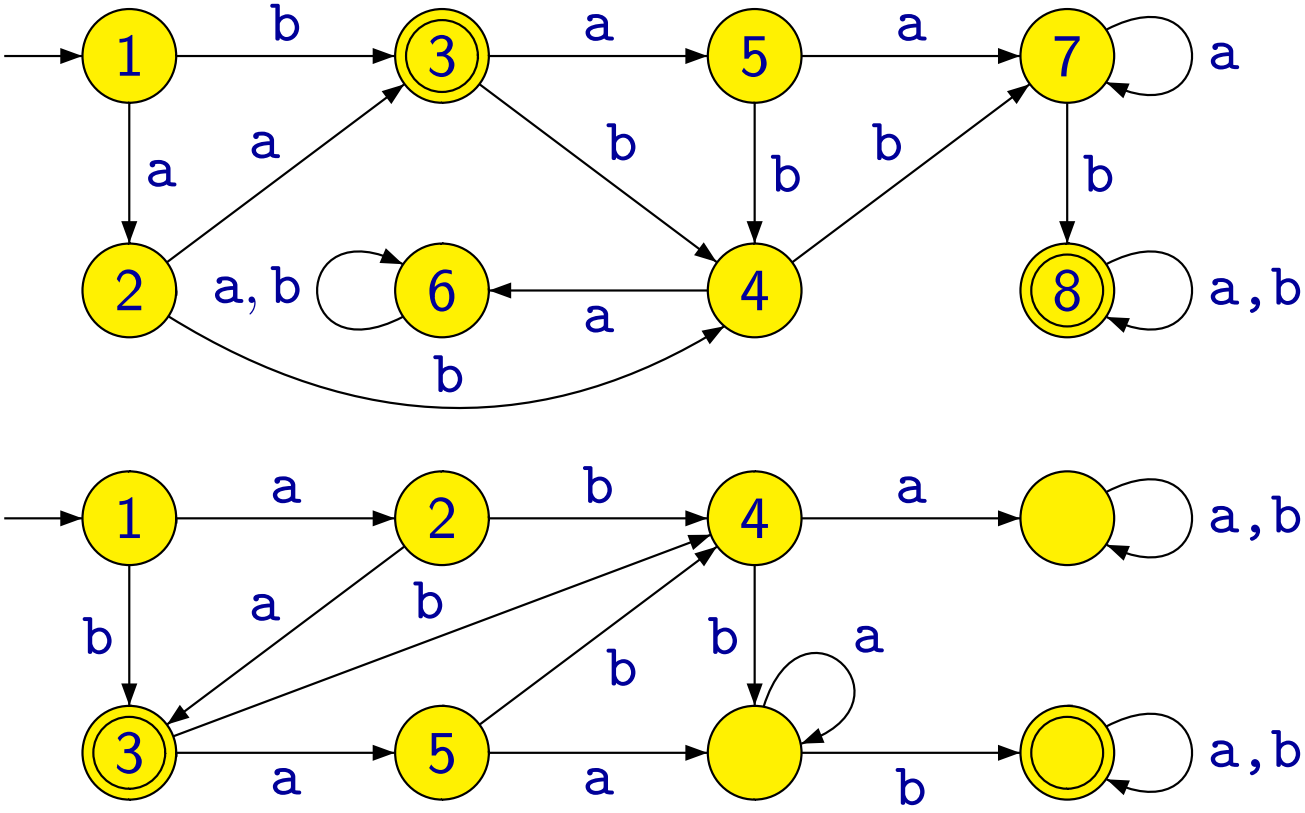
Normovaný tvar automatu



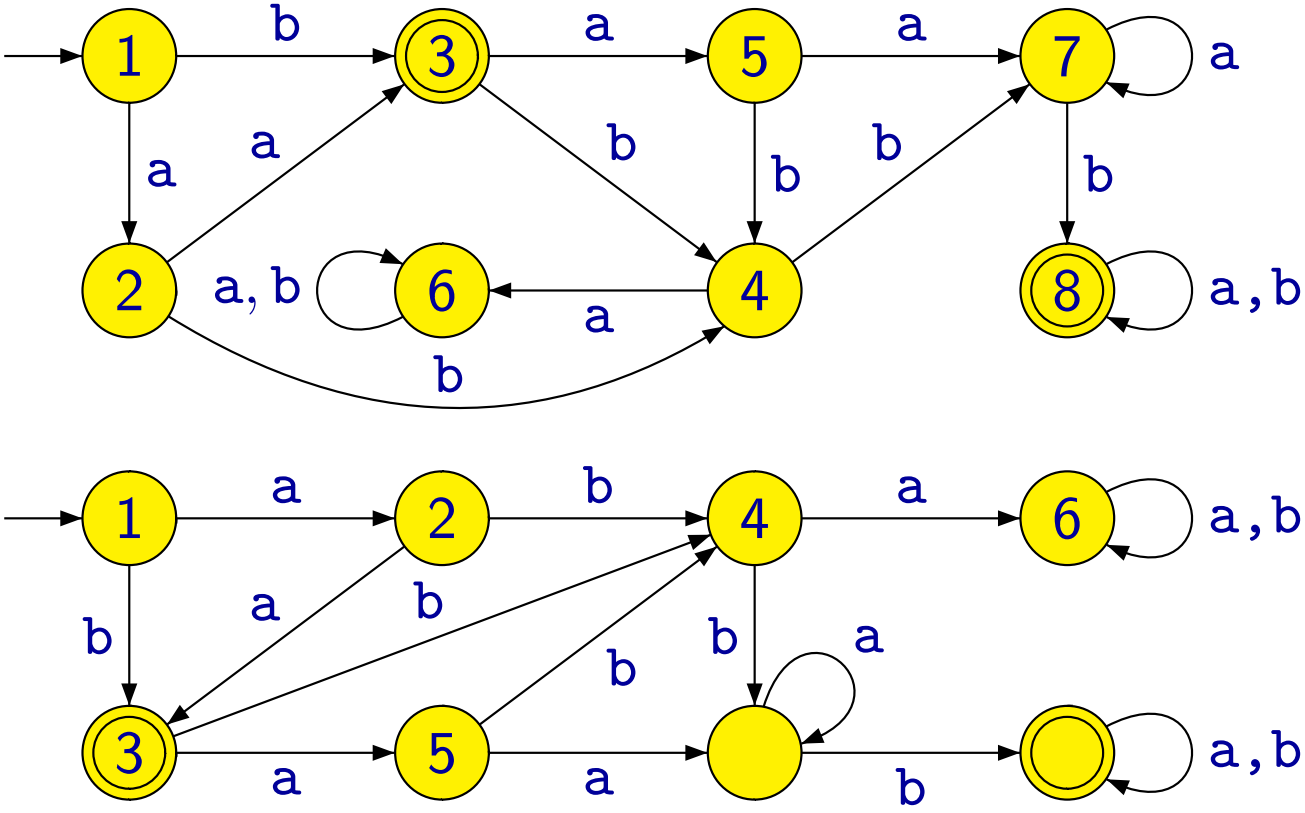
Normovaný tvar automatu



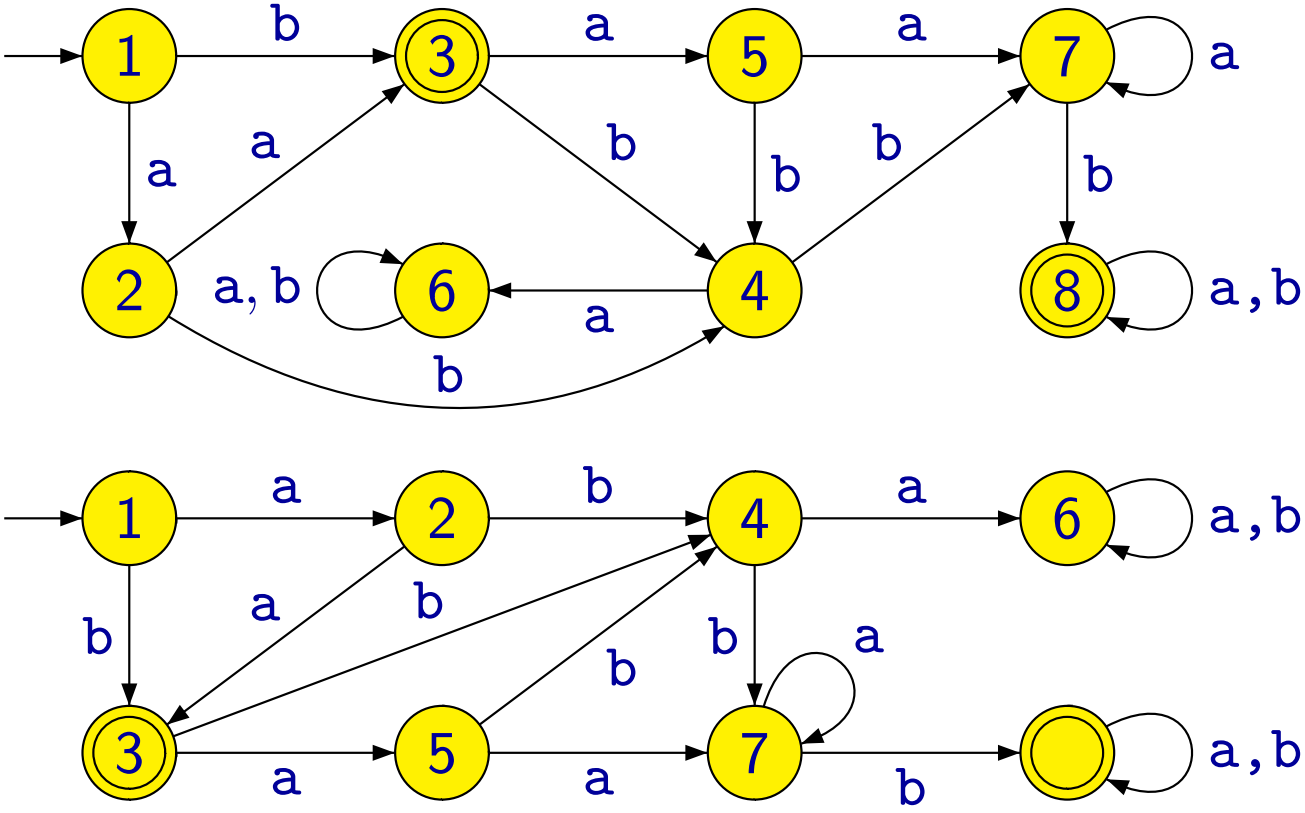
Normovaný tvar automatu



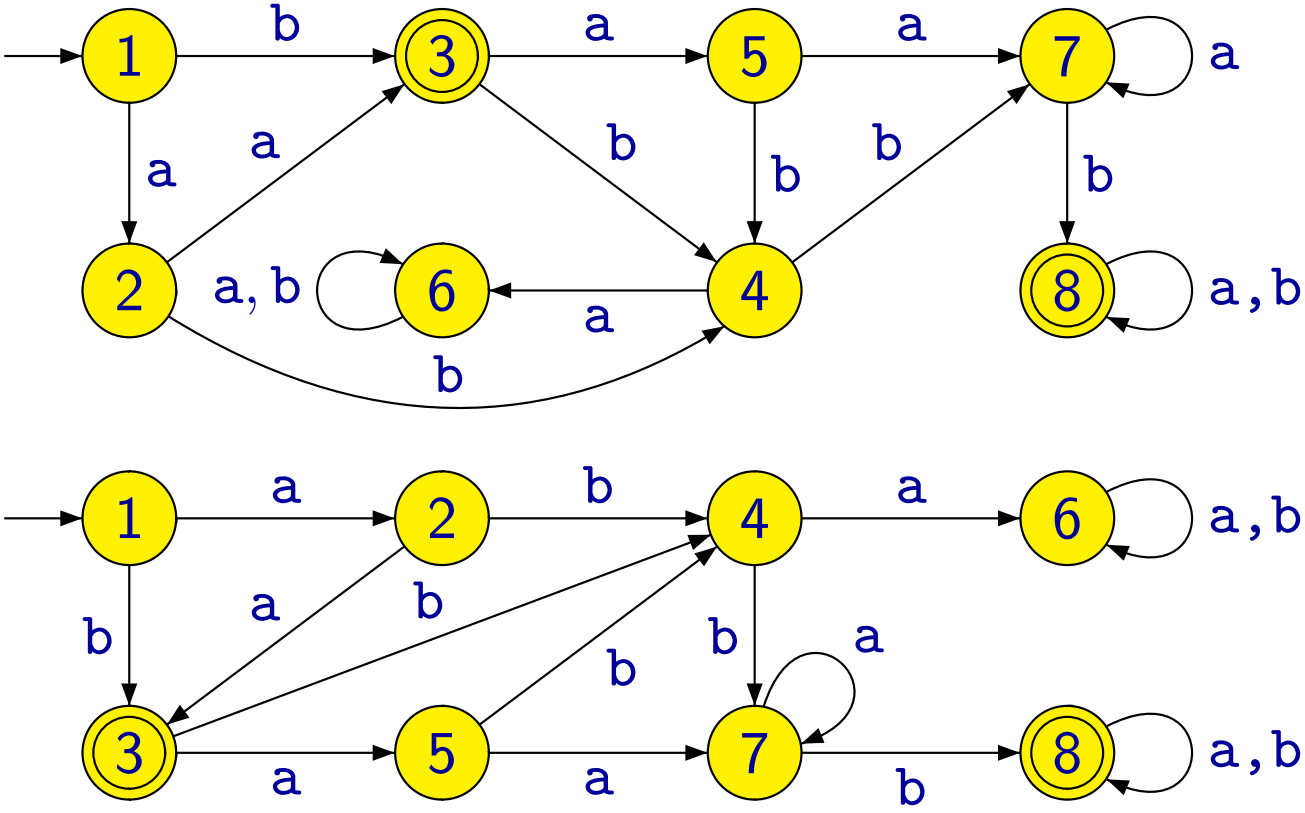
Normovaný tvar automatu



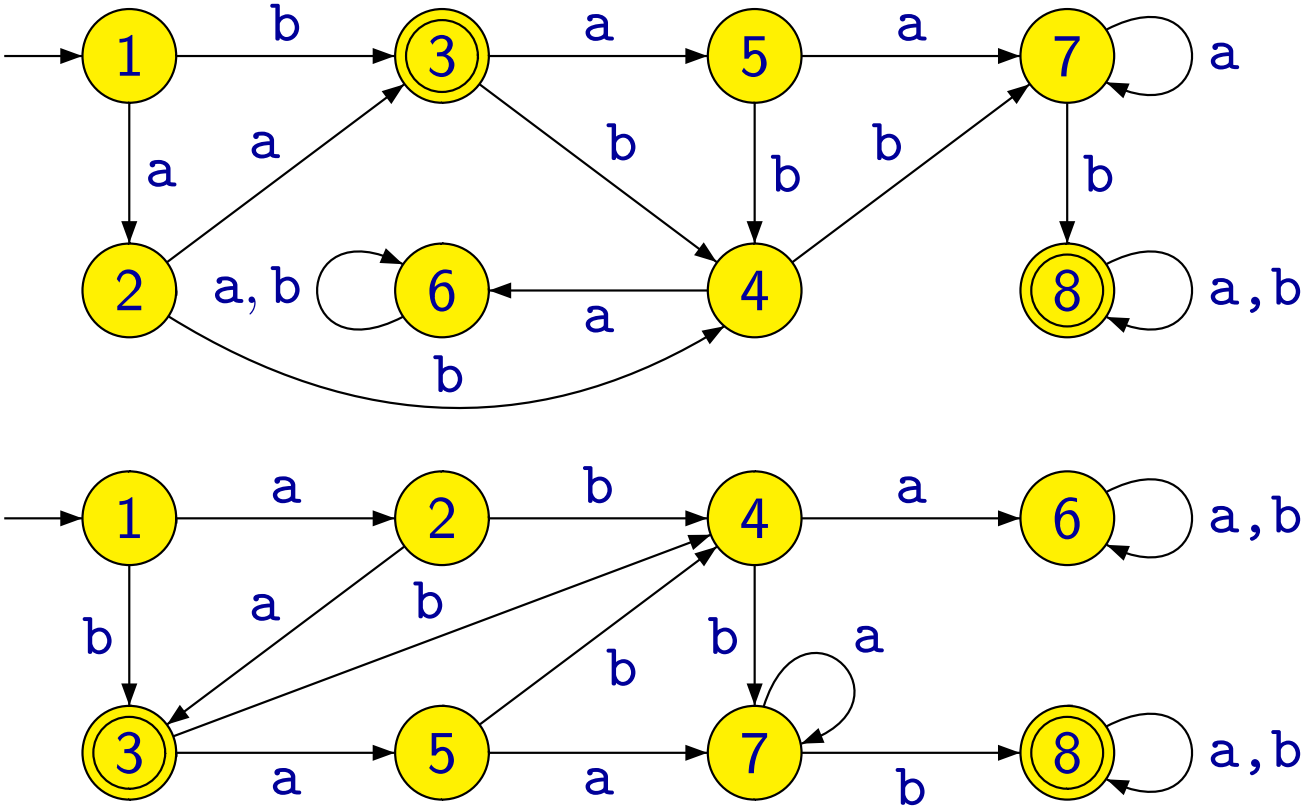
Normovaný tvar automatu



Normovaný tvar automatu

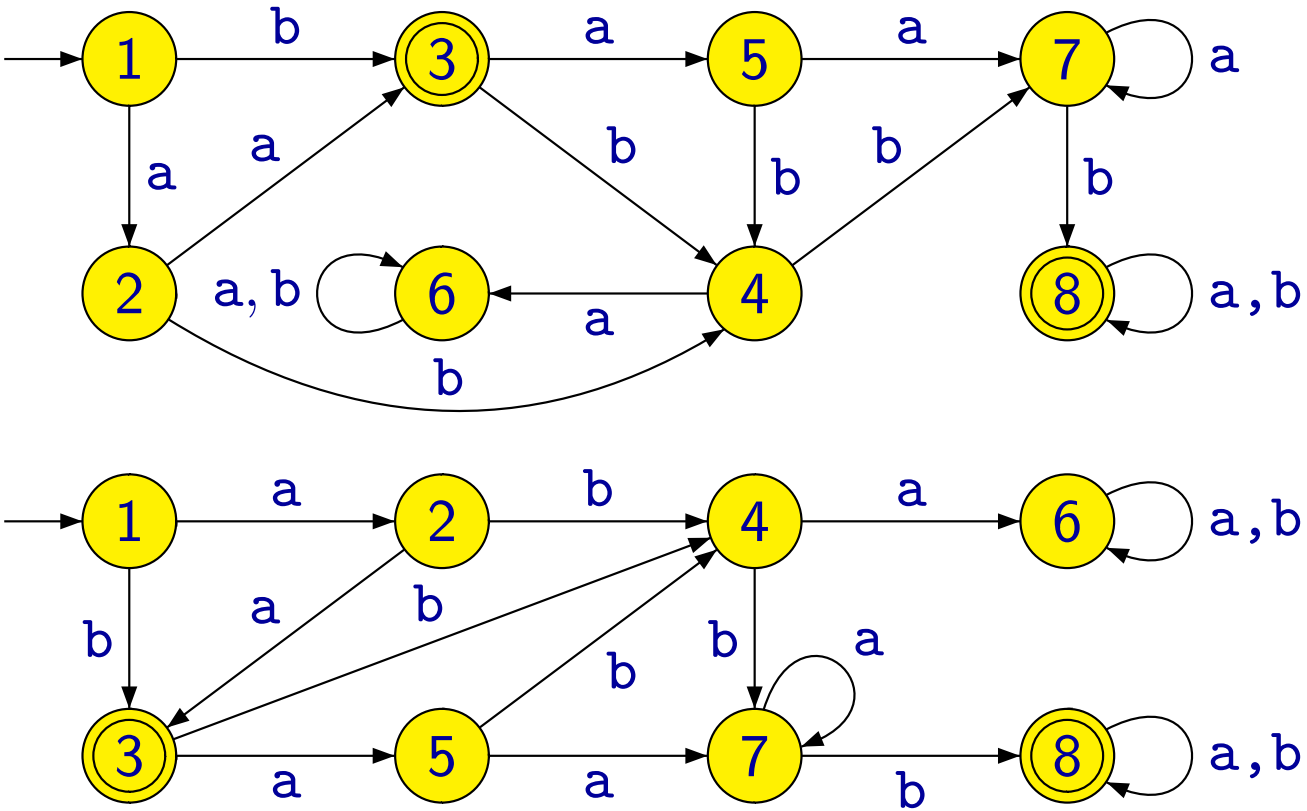


Normovaný tvar automatu



	a	b
$\longrightarrow$ 1	2	3
2	3	4
$\longleftarrow$ 3	5	4
4	6	7
5	7	4
6	6	6
7	7	8
$\longleftarrow$ 8	8	8

Normovaný tvar automatu



	a	b
$\rightarrow 1$	2	3
2	3	4
$\leftarrow 3$	5	4
4	6	7
5	7	4
6	6	6
7	7	8
$\leftarrow 8$	8	8

- Nyní už je přechodová funkce stejná pro oba automaty, stejné jsou i koncové a počáteční stav
- Automaty jsou tedy nejen ekvivalentní, ale dokonce stejné
- Říkáme, že automaty jsou v normovaném tvaru

Normovaný tvar automatu

Definice

Mějme abecedu Σ spolu s úplný ostrým uspořádáním $<$ nad znaky abecedy. Pro slova nad abecedou Σ definujeme úplné ostré uspořádání $<_L$ následovně (pro všechna možná slova $u, v \in \Sigma^*$):

- Pokud $|u| < |v|$, potom $u <_L v$
- Pokud $|u| = |v|$ a u je lexikograficky menší než v, potom $u <_L v$

Normovaný tvar automatu

Definice

Mějme abecedu Σ spolu s úplný ostrým uspořádáním $<$ nad znaky abecedy. Pro slova nad abecedou Σ definujeme úplné ostré uspořádání $<_L$ následovně (pro všechna možná slova $u, v \in \Sigma^*$):

- Pokud $|u| < |v|$, potom $u <_L v$
- Pokud $|u| = |v|$ a u je lexikograficky menší než v, potom $u <_L v$

Příklad: Nad abecedou $\Sigma = \{a, b\}$ uvažujeme běžné uspořádání $a < b$. Potom platí $\varepsilon <_L a <_L b <_L aa <_L bb <_L aba <_L baa <_L aaaa <_L bbbbb$.

Normovaný tvar automatu

Definice

Mějme abecedu Σ spolu s úplný ostrým uspořádáním $<$ nad znaky abecedy. Pro slova nad abecedou Σ definujeme úplné ostré uspořádání $<_L$ následovně (pro všechna možná slova $u, v \in \Sigma^*$):

- Pokud $|u| < |v|$, potom $u <_L v$
- Pokud $|u| = |v|$ a u je lexikograficky menší než v, potom $u <_L v$

Příklad: Nad abecedou $\Sigma = \{a, b\}$ uvažujeme běžné uspořádání $a < b$. Potom platí $\varepsilon <_L a <_L b <_L aa <_L bb <_L aba <_L baa <_L aaaa <_L bbbbb$.

Příklad: Nad abecedou $\Sigma = \{1, 2, 3\}$ uvažujeme běžné uspořádání $1 < 2 < 3$. Potom platí $\varepsilon <_L 1 <_L 3 <_L 31 <_L 33 <_L 111 <_L 321$.

Normovaný tvar automatu

Definice

Mějme abecedu Σ spolu s úplný ostrým uspořádáním $<$ nad znaky abecedy. Pro slova nad abecedou Σ definujeme úplné ostré uspořádání $<_L$ následovně (pro všechna možná slova $u, v \in \Sigma^*$):

- Pokud $|u| < |v|$, potom $u <_L v$
- Pokud $|u| = |v|$ a u je lexikograficky menší než v, potom $u <_L v$

Příklad: Nad abecedou $\Sigma = \{a, b\}$ uvažujeme běžné uspořádání $a < b$. Potom platí $\varepsilon <_L a <_L b <_L aa <_L bb <_L aba <_L baa <_L aaaa <_L bbbbb$.

Příklad: Nad abecedou $\Sigma = \{1, 2, 3\}$ uvažujeme běžné uspořádání $1 < 2 < 3$. Potom platí $\varepsilon <_L 1 <_L 3 <_L 31 <_L 33 <_L 111 <_L 321$.

Příklad: Nad abecedou $\Sigma = \{0, 1, 2\}$ uvažujeme běžné uspořádání $0 < 1 < 2$. Potom platí $\varepsilon <_L 1 <_L 2 <_L 01 <_L 03 <_L 111 <_L 0021$.

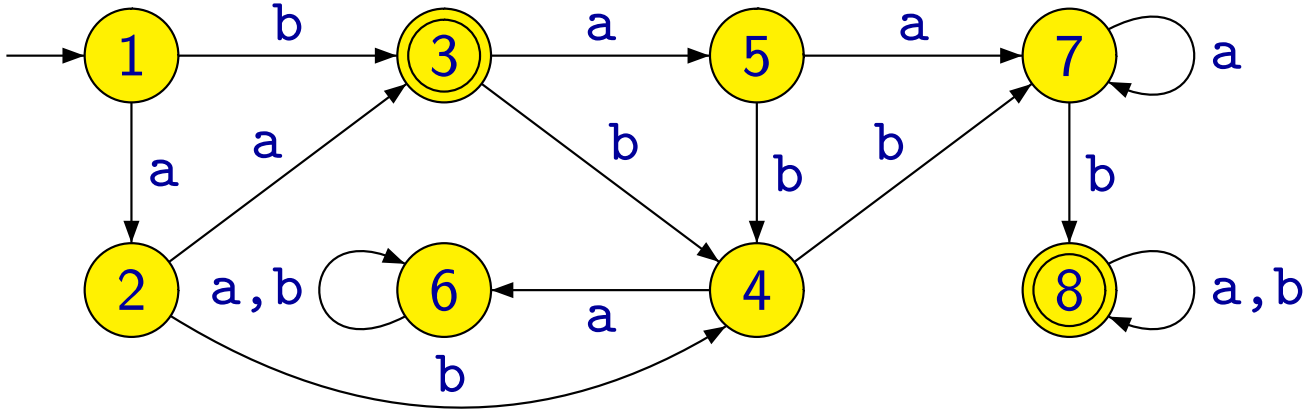
Normovaný tvar automatu

Definice

Mějme deterministický konečný automat $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ bez nedosažitelných stavů a uspořádání prvků abecedy Σ (které indukuje uspořádání $<_L$ na Σ^*).
Označme pro každý stav $i \in \{1, 2, \ldots, n\}$ symbolem u_i nejmenší slovo podle uspořádání $<_L$ takové, že pro něj platí $\delta(1, u_i) = i$.
Potom řekneme, že $\mathcal{A}$ je v normovaném tvaru, pokud

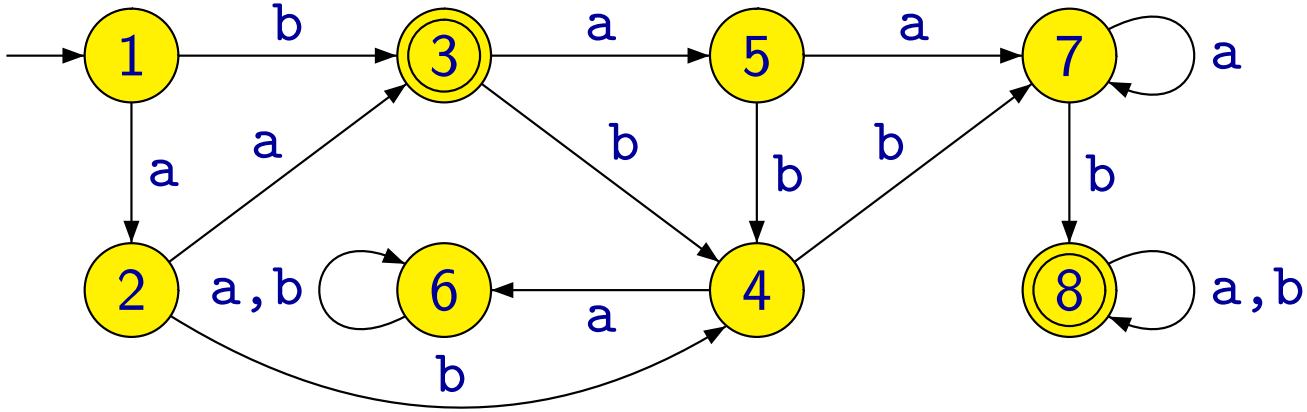
- $Q = \{1, 2, 3, \ldots, n\}$ pro nějaké $n \geq 1$
- 1 je počáteční stav
- Pro každou dvojici stavů $i, j \in \{1, 2, \ldots, n\}$ takovou, že $i < j$, platí $u_i <_L u_j$.

Normovaný tvar automatu



- Do stavu 4 se dostaneme slovy ab, bb, aab, bab tedy $u_4 = ab$
- Do stavu 5 se dostaneme slovy ba, aaa tedy $u_5 = ba$
- $ab <_L ba$ proto jsou stavy $4, 5$ označeny v tomto pořadí

Normovaný tvar automatu



- Do stavu 4 se dostaneme slovy ab, bb, aab, bab tedy $u_4 = ab$
- Do stavu 5 se dostaneme slovy ba, aaa tedy $u_5 = ba$
- $ab <_L ba$ proto jsou stavy $4, 5$ označeny v tomto pořadí
- Pro ostatní stavy jsou nejmenší slova následující:

$$\begin{aligned} u_1 &= \varepsilon & u_5 &= ba \\ u_2 &= a & u_6 &= aba \\ u_3 &= b & u_7 &= abb \\ u_4 &= ab & u_8 &= abbb \end{aligned}$$

Neregulární jazyky

Neregulární jazyky

Ne všechny jazyky jsou regulární.

Existují jazyky, pro které neexistuje žádný konečný automat, který by je rozpoznával.

Příklady neregulárních jazyků:

- $L_1 = \{a^n b^n \mid n \geq 0\}$
- $L_2 = \{ww \mid w \in \{a, b\}^*\}$
- $L_3 = \{ww^R \mid w \in \{a, b\}^*\}$

Poznámka: Existence neregulárních jazyků vyplývá již z faktu, že automatů pracujících nad nějakou abecedou Σ je jen spočetně mnoho, zatímco jazyků nad abecedou Σ je nespočetně mnoho.

Neregulární jazyky

Jak dokázat o nějakém jazyce L, že není regulární?

Jazyk není regulární, jestliže neexistuje (tj. není možné sestrojit) konečný automat, který by ho rozpoznával.

Jak ale dokázat, že něco neexistuje?

Neregulární jazyky

Jak dokázat o nějakém jazyce L, že není regulární?

Jazyk není regulární, jestliže neexistuje (tj. není možné sestrojit) konečný automat, který by ho rozpoznával.

Jak ale dokázat, že něco neexistuje?

Odpověď: Stačí ukázat, že jazyk L nemá nějakou vlastnost, kterou má každý jazyk rozpoznávaný nějakým konečným automatem.

Neregulární jazyky

Jak dokázat o nějakém jazyce L, že není regulární?

Jazyk není regulární, jestliže neexistuje (tj. není možné sestrojit) konečný automat, který by ho rozpoznával.

Jak ale dokázat, že něco neexistuje?

Odpověď: Stačí ukázat, že jazyk L nemá nějakou vlastnost, kterou má každý jazyk rozpoznávaný nějakým konečným automatem.

Dva základní postupy dokazování neregularity jazyků:

- využití tzv. pumping lemmatu
- využití Myhillovy-Nerodovy věty (nebudeme se jí dále zabývat)

Neregulární jazyky

Tvrzení

Každý konečný jazyk je regulární.

Důkaz: Konečný jazyk s n slovy můžeme popsat regulárním výrazem $w_1 + w_2 + \ldots + w_n$. Jazyk je tedy regulární.

Neregulární jazyky

Tvrzení

Každý konečný jazyk je regulární.

Důkaz: Konečný jazyk s n slovy můžeme popsat regulárním výrazem $w_1 + w_2 + \ldots + w_n$. Jazyk je tedy regulární.

Tvrzení

Je-li jazyk nekonečný, obsahuje pro každou konstantu $k \in \mathbb{N}$ nějaké slovo w takové, že $|w| > k$

Důkaz: Uvažujeme jen konečnou abecedu. Pro každou konstantu je tedy jen konečně mnoho slov kratších než tato konstanta. Pokud má být jazyk nekonečný, musí být i slovo delší, než jakákoliv konstanta. Ve skutečnosti je slov delších než jakákoliv konstanta nekonečně mnoho.

Pumping Lemma

Předpokládejme, že jazyk L je rozpoznáván nějakým konkrétním deterministickým automatem A, tj. $L = L(A)$.

Vezměme nyní nějaké libovolné slovo $z \in L$, kde $z = a_1 a_2 \cdots a_k$.

Protože automat A slovo z přijímá, musí tomuto slovu odpovídat určitý přijímající výpočet automatu, tj. posloupnost stavů:

$$q_0,\ q_1,\ q_2,\ \ldots,\ q_{k-1},\ q_k$$

délky $k+1$, kde

- q_0 je počáteční stav
- $\delta(q_{i-1}, a_i) = q_i$ pro $\forall i \in \{1, 2, \ldots, k\}$
- q_k je přijímající stav

Pumping Lemma

Předpokládejme, že A má n stavů a že $|z| \geq n$.

Pak máme posloupnost délky minimálně $n+1$, ve které se může vyskytovat maximálně n různých stavů.

Z toho plyne, že musí existovat alespoň jeden stav q, který se v této posloupnosti vyskytuje alespoň dvakrát.

Jde o aplikaci tzv. **holubníkového principu (pigeonhole principle)**.

Holubníkový princip

Jestliže mám $n+1$ holubů rozmístěných do n klecí, pak jsou alespoň v jedné kleci minimálně dva holubi.

Pumping Lemma

Řekněme, že opakující stav se vyskytuje na pozicích i, j, tj. $q_i = q_j$, kde $i < j$.

$$q_0, \cdots, q_i, \cdots, q_j, \cdots, q_k$$

Poznámka: Zjevně můžeme najít taková i, j, že $i < j \leq n$

Slovo z můžeme rozdělit na tři části:

$$\underbrace{a_1 \cdots a_i}_{u} \quad \underbrace{a_{i+1} \cdots a_j}_{v} \quad \underbrace{a_{j+1} \cdots a_k}_{w}$$

- $\delta^*(q_0, u) = q_i$
- $\delta^*(q_i, v) = q_j = q_i$
- $\delta^*(q_j, w) = q_k$

Pumping Lemma

Vezměme nyní slova:

$$\underbrace{a_1 \cdots a_i}_{u} \quad \underbrace{a_{j+1} \cdots a_k}_{w}$$

$$\underbrace{a_1 \cdots a_i}_{u} \quad \underbrace{a_{i+1} \cdots a_j}_{v} \quad \underbrace{a_{i+1} \cdots a_j}_{v} \quad \underbrace{a_{j+1} \cdots a_k}_{w}$$

$$\underbrace{a_1 \cdots a_i}_{u} \quad \underbrace{a_{i+1} \cdots a_j}_{v} \quad \underbrace{a_{i+1} \cdots a_j}_{v} \quad \underbrace{a_{i+1} \cdots a_j}_{v} \quad \underbrace{a_{j+1} \cdots a_k}_{w}$$

$$\ldots$$

Je zřejmé, že A přijme každé z nich, vzhledem k tomu, že

- $\delta^*(q_0, u) = q_i$
- $\delta^*(q_i, v) = q_j = q_i$
- $\delta^*(q_j, w) = q_k$, $q_k \in F$

Pumping Lemma

Pumping Lemma

Jestliže jazyk L je regulární, pak existuje n takové, že každé slovo $z \in L$ takové, že $|z| \geq n$, je možné rozdělit na podslova u, v, w taková, že $z = uvw$, $|uv| \leq n$, $|v| \geq 1$ a pro všechna $i \geq 0$ platí $uv^i w \in L$.

Formálně zapsáno:

Jestliže L je regulární, pak

$(\exists n)(\forall z \text{ tž. } z \in L, |z| \geq n)(\exists u, v, w \text{ tž. } z = uvw, |uv| \leq n, |v| \geq 1)$
$(\forall i \geq 0) : uv^i w \in L$

Pumping Lemma

Tvrzení je možné obrátit. ($A \Rightarrow B$ je totéž, co $\neg B \Rightarrow \neg A$.)

Jestliže

$(\forall n)(\exists z \text{ tž. } z \in L, |z| \geq n)(\forall u, v, w \text{ tž. } z = uvw, |uv| \leq n, |v| \geq 1)$
$(\exists i \geq 0) : uv^i w \notin L,$

pak L není regulární.

Pokud tedy chceme ukázat, že jazyk L není regulární, stačí ukázat, že splňuje výše uvedenou podmínku.

Pumping Lemma

Příklad: Uvažujme jazyk $L = \{a^i b^i \mid i \geq 0\}$.

- Předpokládáme, že L je rozpoznáván nějakým automatem s n stavy.
- Zvolme slovo $z = a^n b^n$.
- Uvažujme všechny možnosti, jak může být z rozděleno na podslova u, v, w splňující podmínky $|uv| \leq n$ a $|v| \geq 1$.
  Zjevně slova u a v obsahují pouze symboly a. Pro každé konkrétní rozdělení existují nějaká j a k taková, že $j + k \leq n$, $k \geq 1$ a
  - $u = a^j$
  - $v = a^k$
  - $w = a^{n-(j+k)} b^n$
- Pokud nyní zvolíme $i = 0$, dostáváme $uv^i w = uw = a^{n-k} b^n$. Protože $n - k < n$, zjevně platí $uv^i w \notin L$.

Pumping Lemma

Poznámka: Dokazování platnosti či neplatnosti formule prvého řádu, kde se střídají universální a existenční kvantifikátory, si můžeme představovat jako hru dvou hráčů, hráče A a hráče B, kde se hráč A snaží dokázat, že formule platí, a hráč B, že formule neplatí.

Hráč A vždy volí hodnotu proměnné vázané existenčním kvantifikátorem a naopak hráč B vždy volí hodnotu proměnné vázané universálním kvantifikátorem.

Pokud například chceme platnost formule vyvrátit, stačí nám najít vítěznou strategii hráče B.

Pumping Lemma

$$(\exists n)(\forall z \text{ tž. } z \in L, |z| \geq n)(\exists u, v, w \text{ tž. } z = uvw,\ |uv| \leq n, |v| \geq 1)$$
$$(\forall i \geq 0) : uv^i w \in L$$

Konkrétně pro Pumping Lemma vypadá hra následovně:

1. A zvolí $n > 0$.
2. B zvolí slovo z takové, že $z \in L$ a $|z| \geq n$.
3. A zvolí slova u, v, w taková, že $z = uvw, |uv| \leq n, |v| \geq 1$.
4. B zvolí $i \geq 0$.
5. Je-li $uv^i w \in L$, vyhrává hráč A. Je-li $uv^i w \notin L$, vyhrává hráč B.

Pumping Lemma

Příklad: $L = \{a^i b^i \mid i \geq 0\}$

1. A zvolí $n > 0$.
2. B zvolí $z = a^n b^n$
3. A zvolí slova u, v, w tž. $z = uvw, |uv| \leq n, |v| \geq 1$
4. B zvolí $i = 0$
5. Vyhrál hráč B, protože bez ohledu na to, co volil hráč A, vždy platí $uv^i w \notin L$, protože neprázdné slovo v se nachází v části slova z tvořené pouze symboly a, a jeho vypuštěním dostaneme slovo tvaru $a^k b^n$, kde $k < n$, a tedy nepatřící do L.