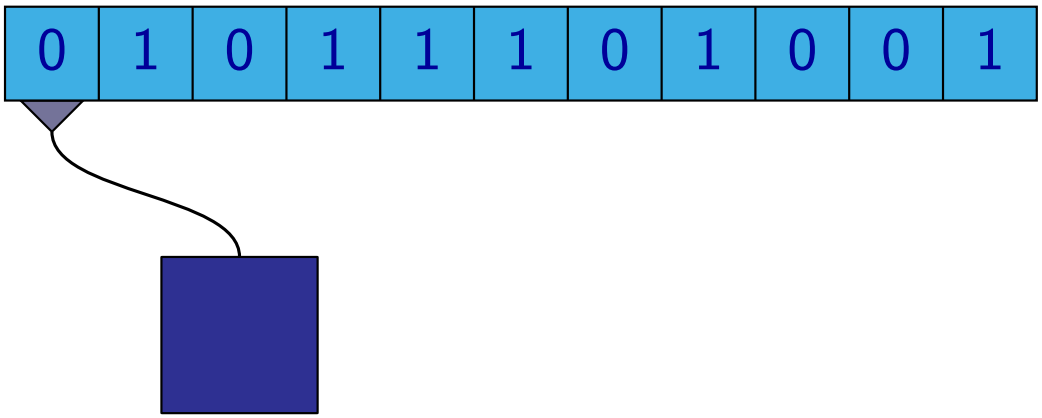
Konečné automaty

Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

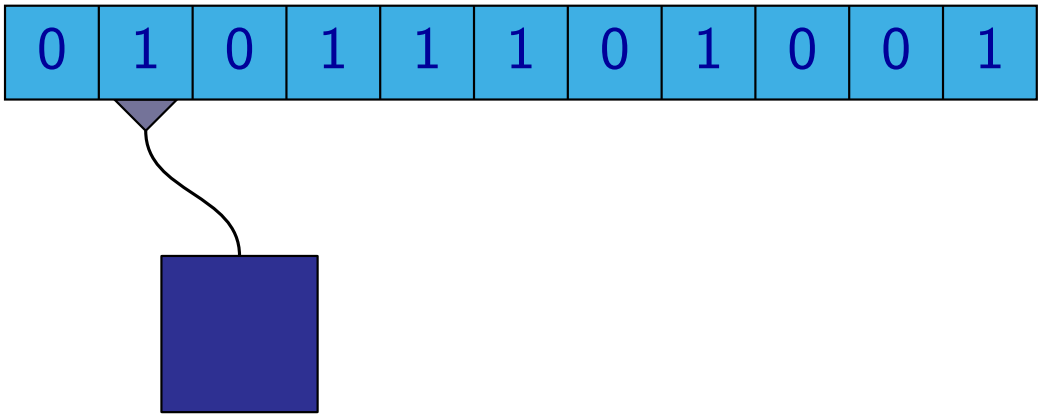


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

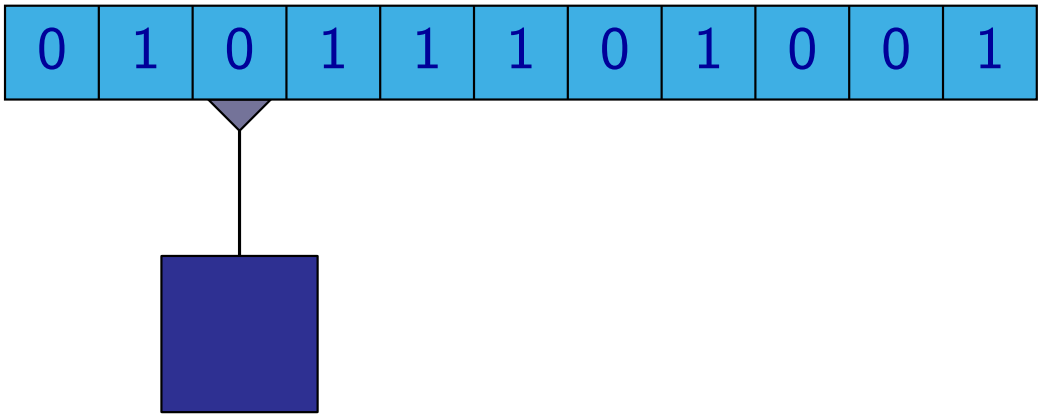


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

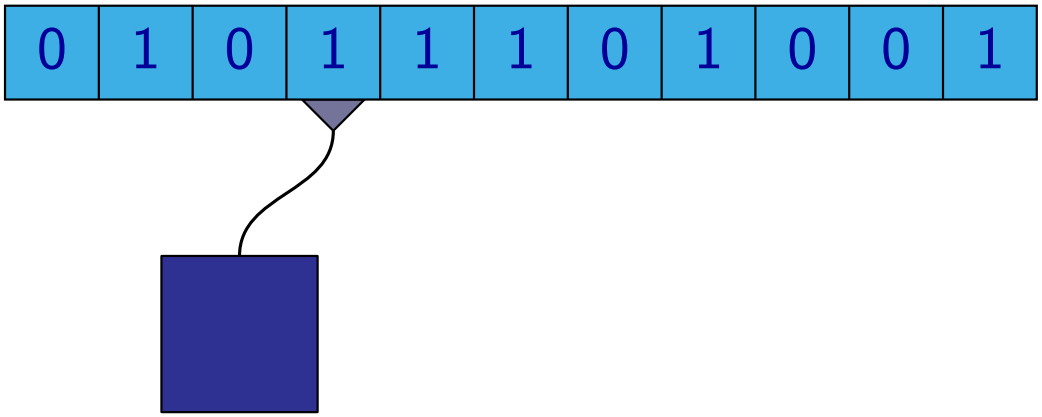


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

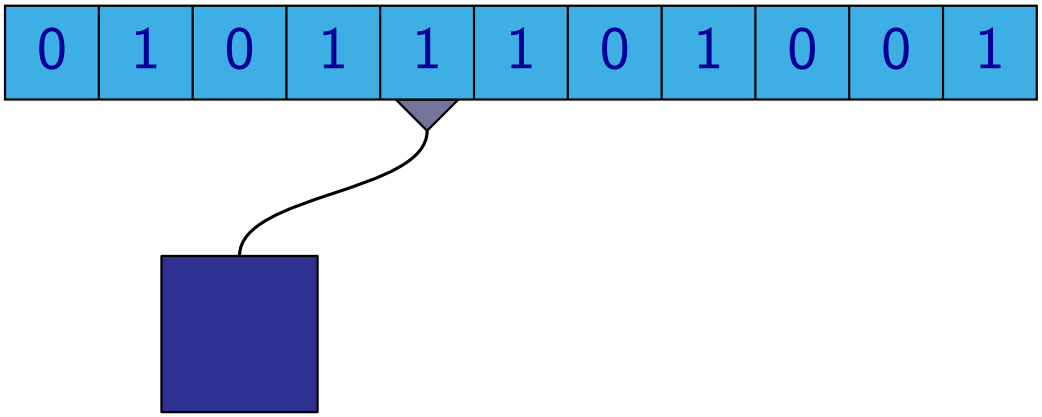


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

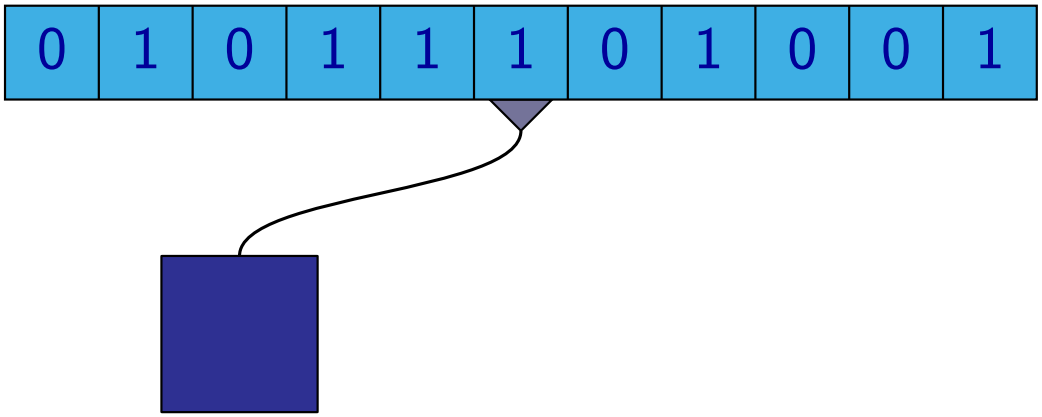


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

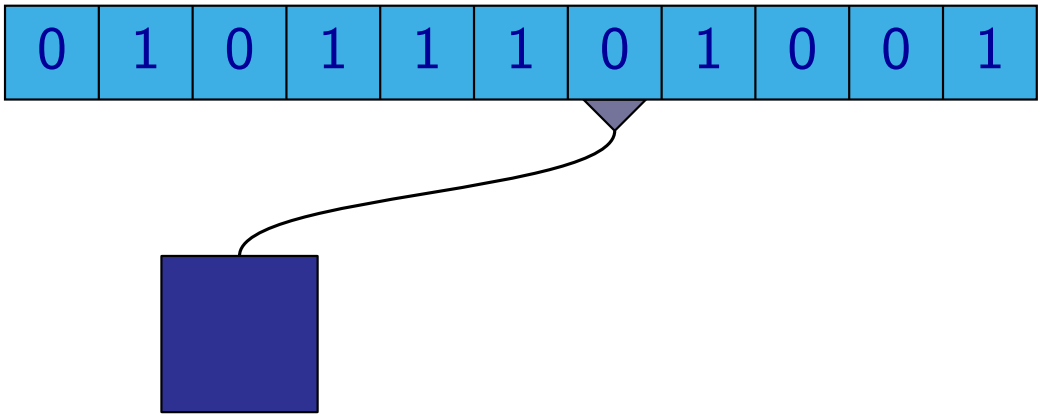


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

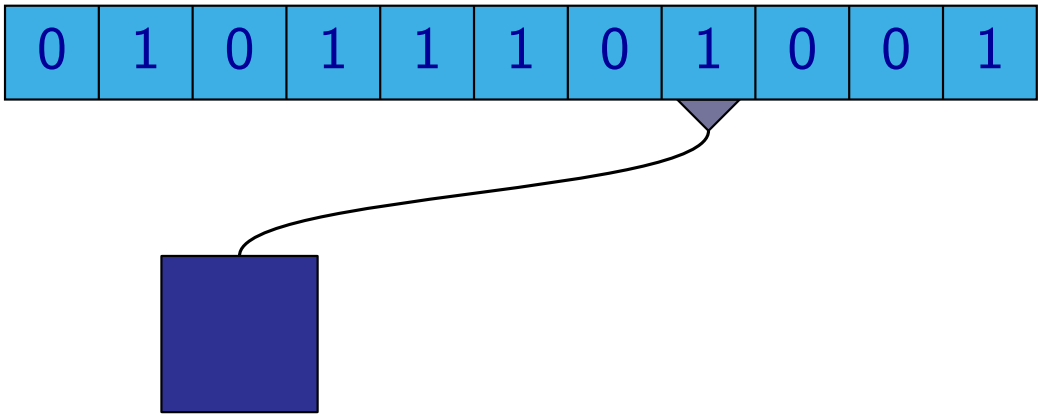


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

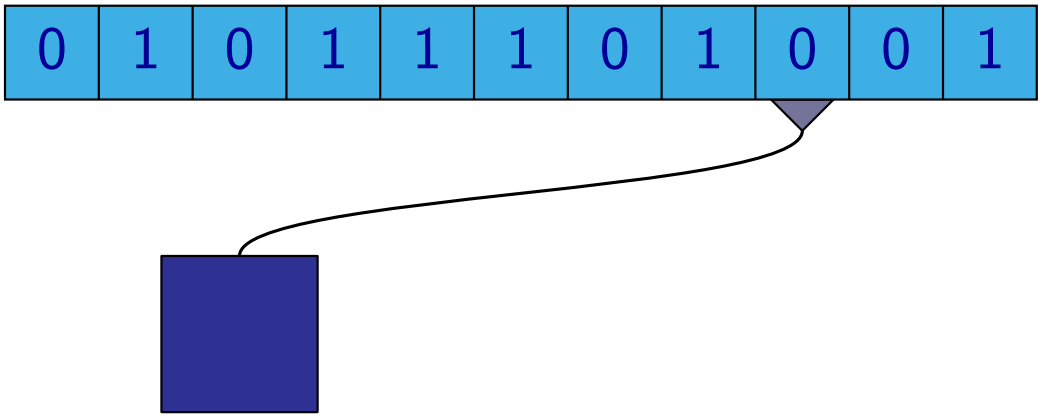


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

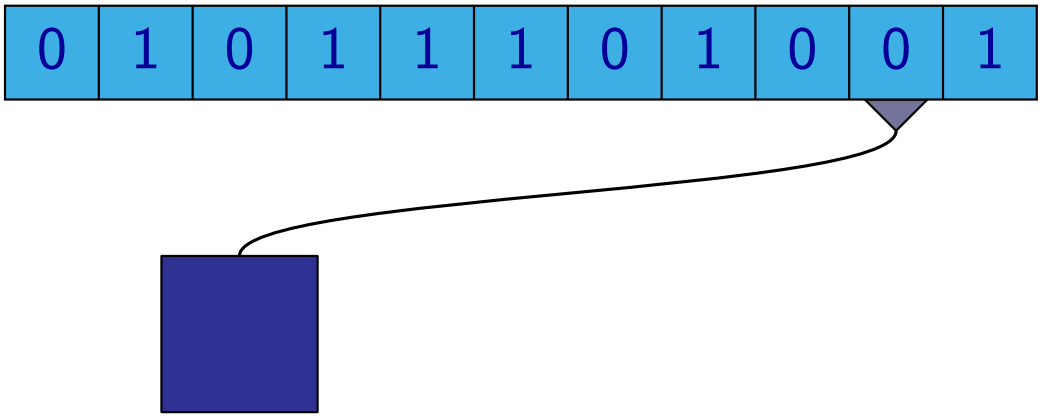


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

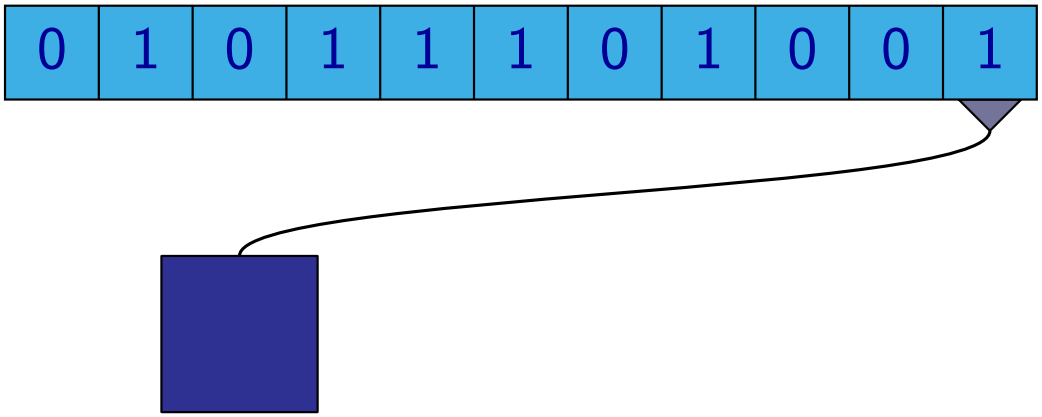


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.

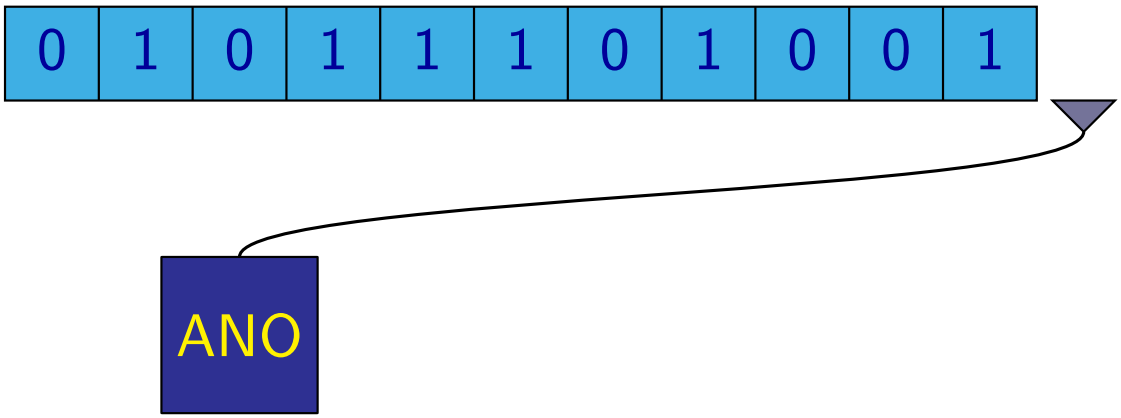


Rozpoznávání jazyka

Příklad: Uvažujme slova nad abecedou $\{0, 1\}$.

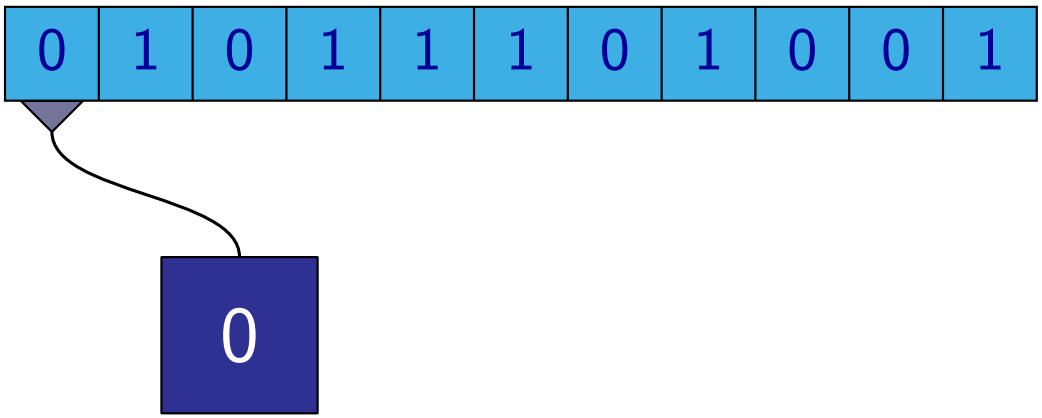
Chtěli bychom rozpoznávat jazyk L, který je tvořen slovy, ve kterých se vyskytuje sudý počet symbolů 1.

Chceme navrhnout zařízení, které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.



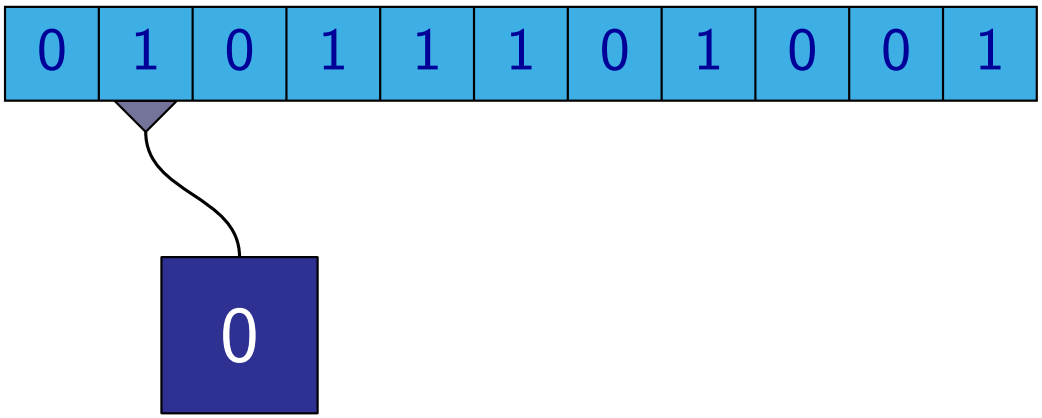
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



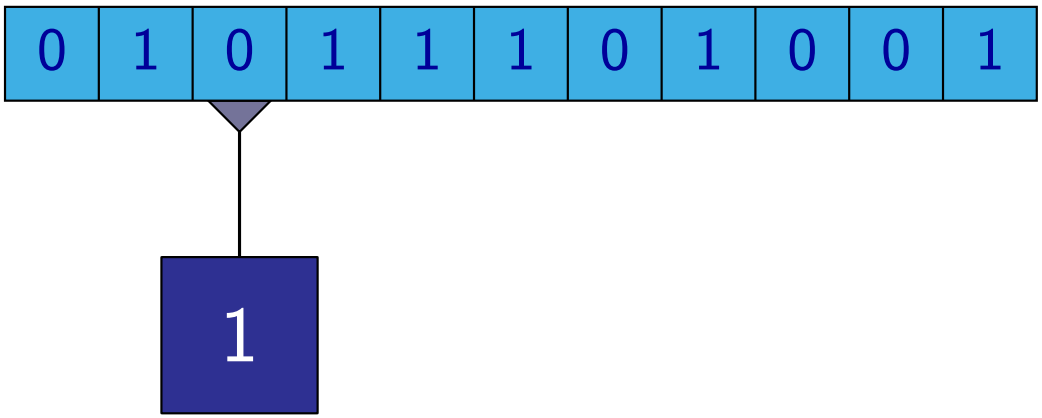
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



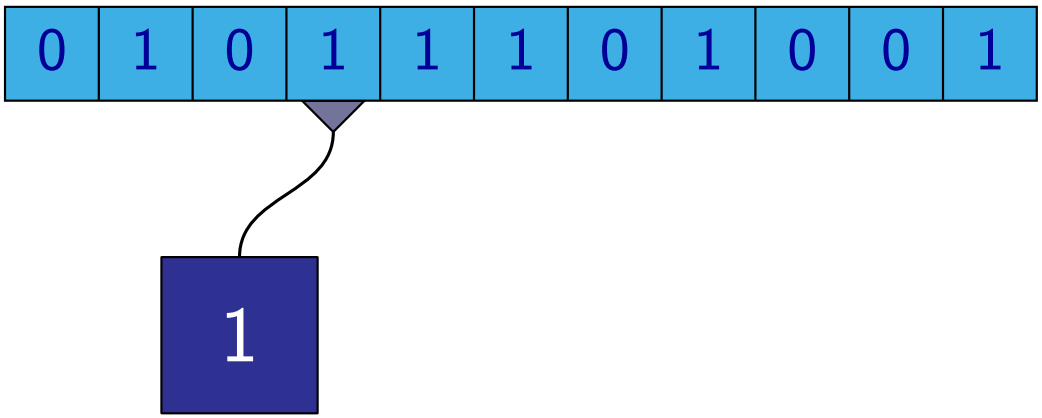
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



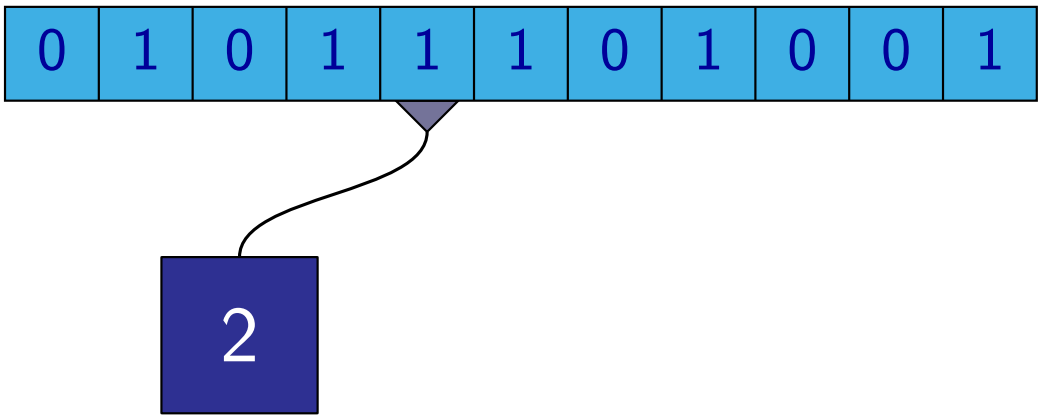
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



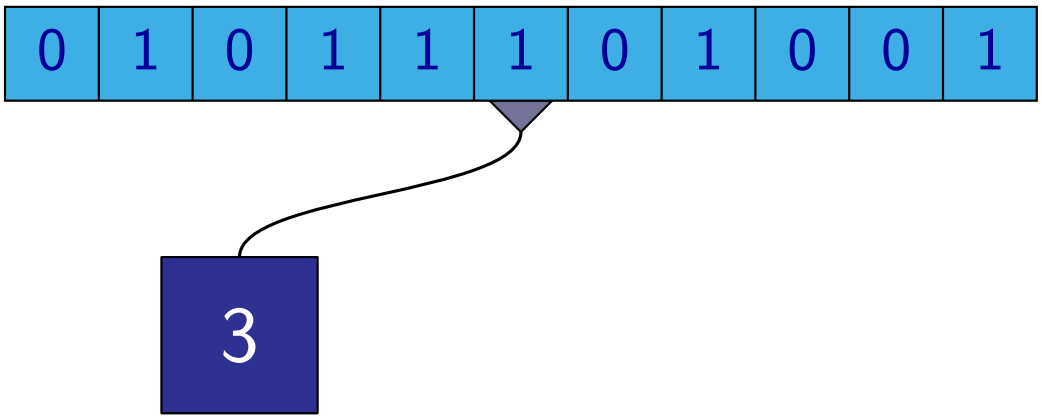
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



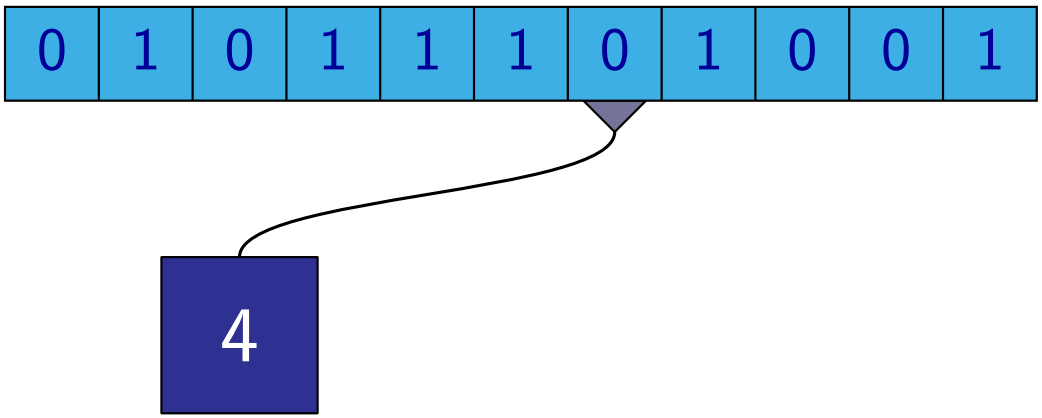
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



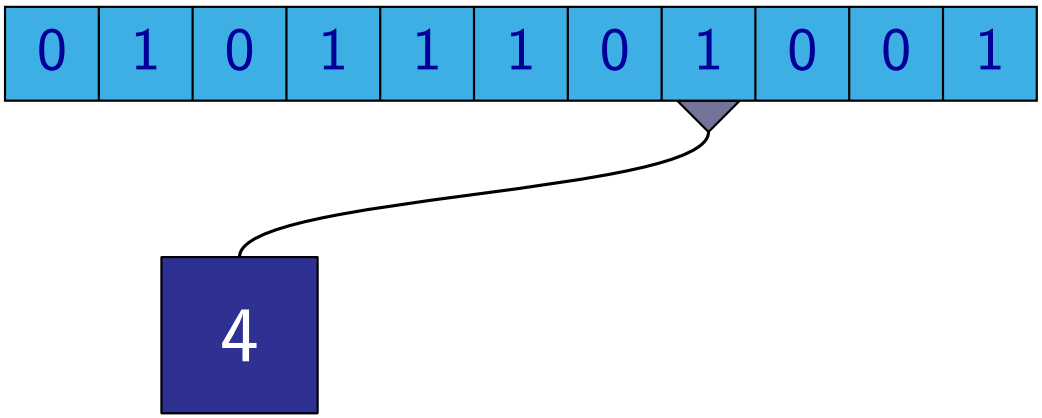
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



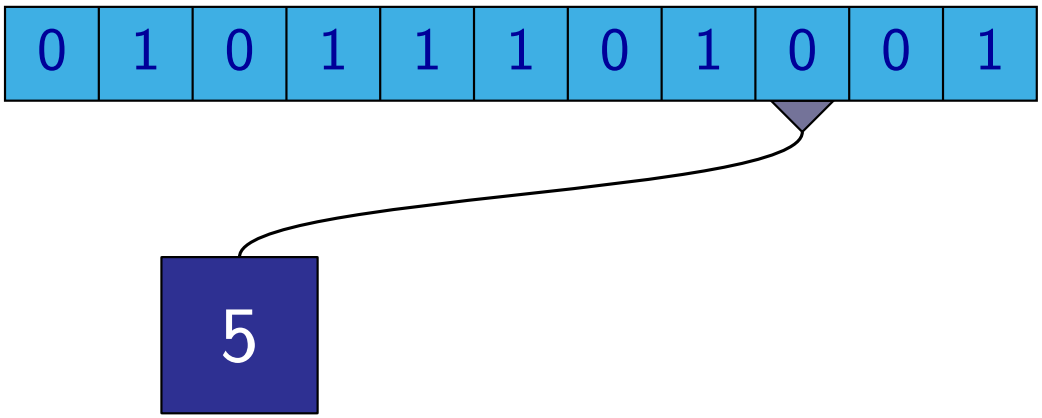
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



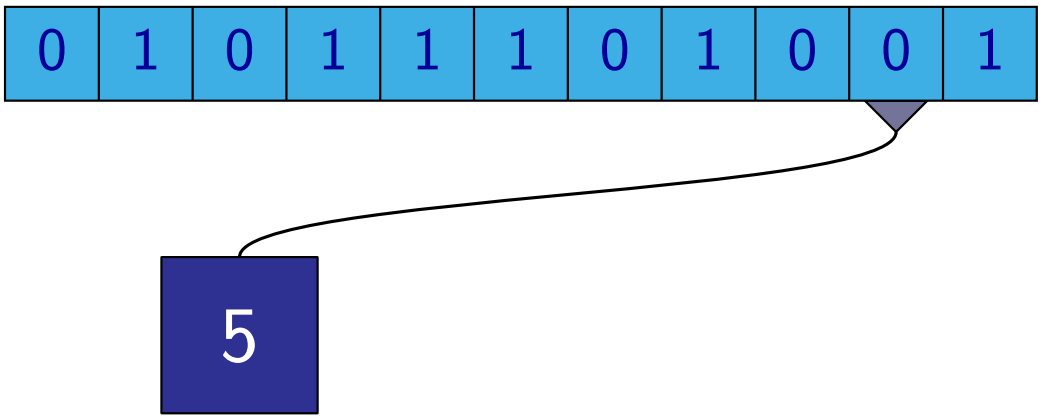
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



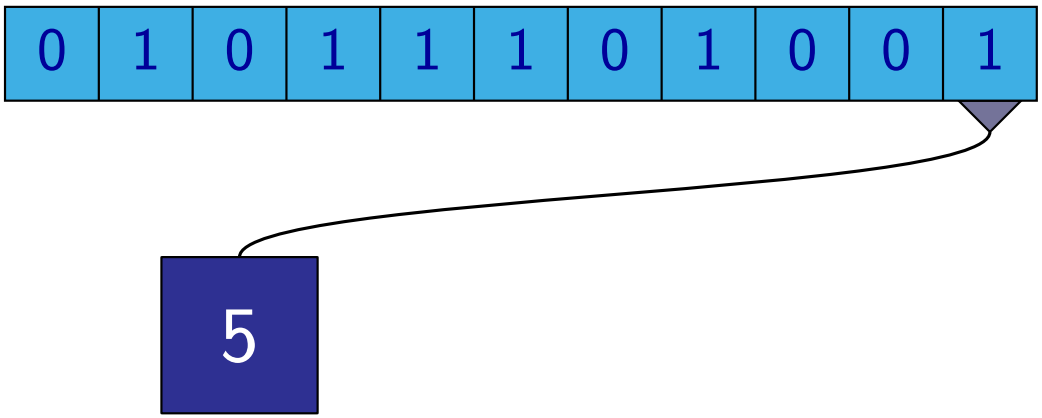
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



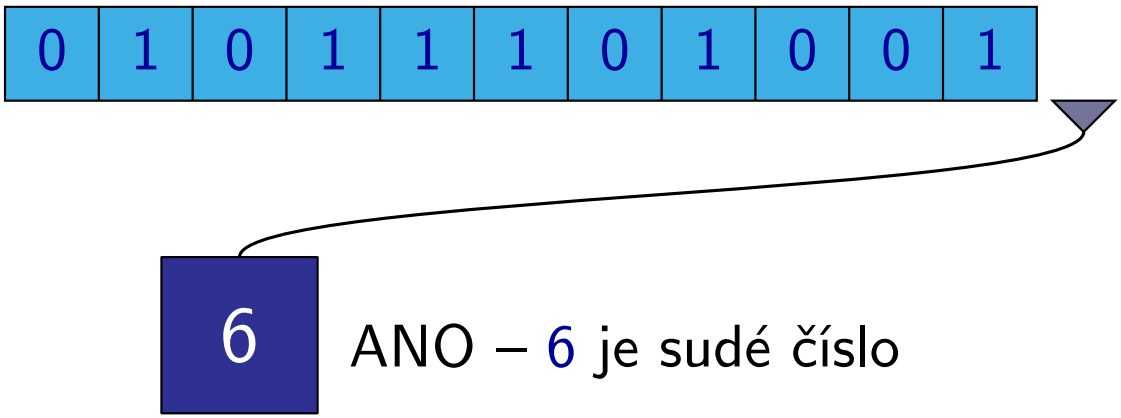
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



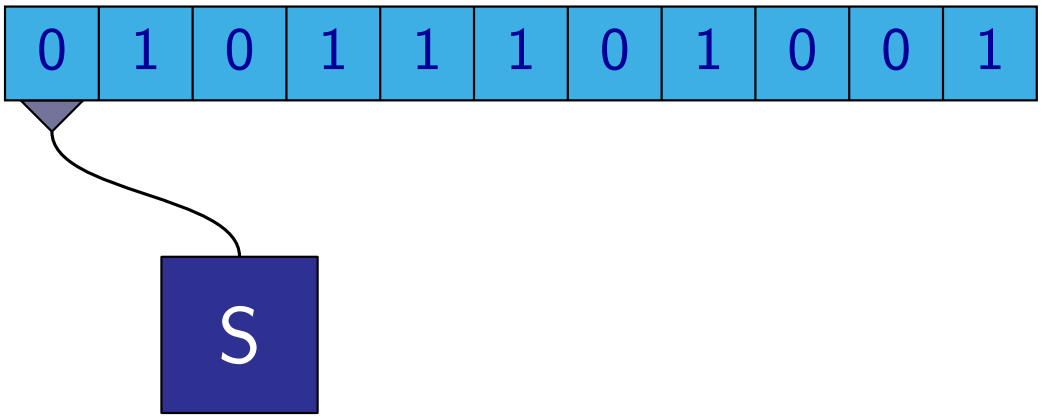
Rozpoznávání jazyka

První nápad: Počítat počet výskytů symbolů 1.



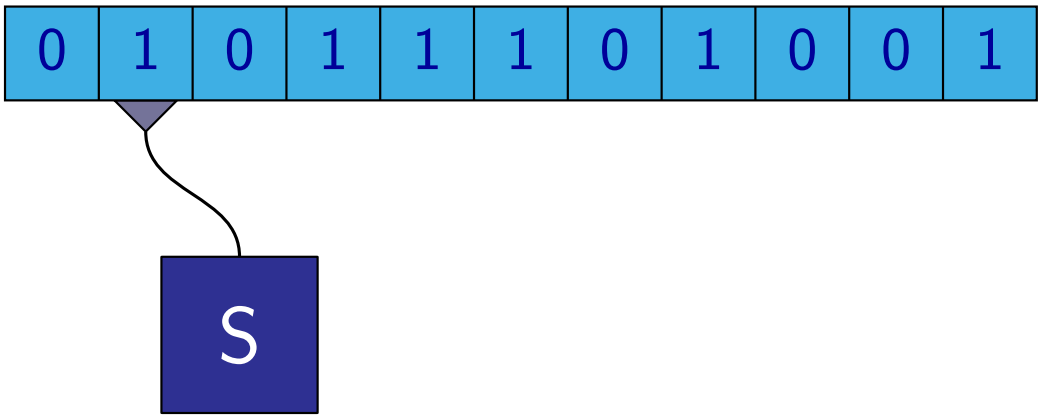
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



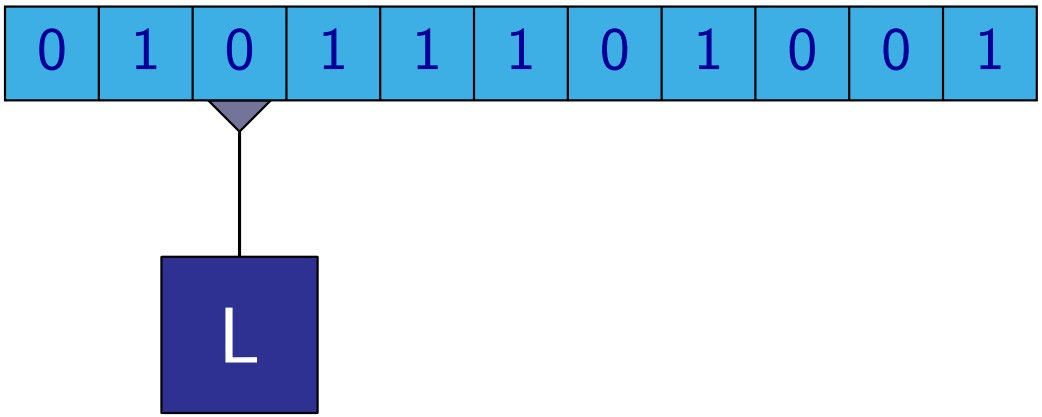
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



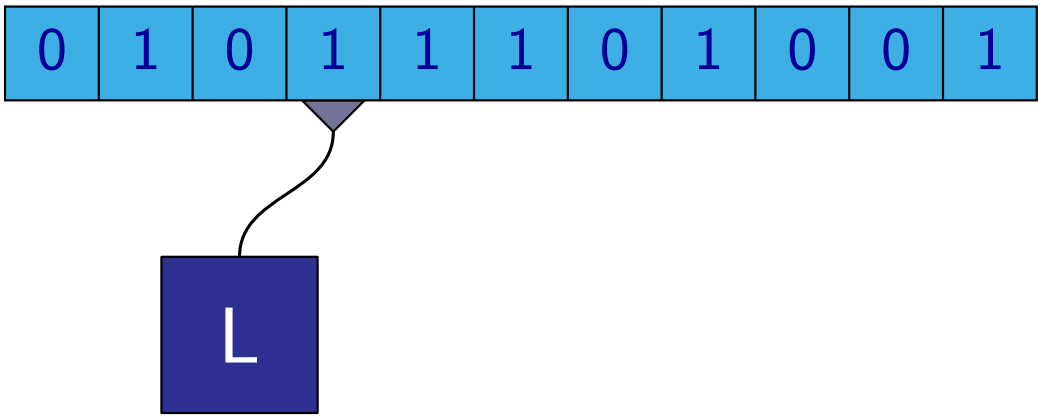
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



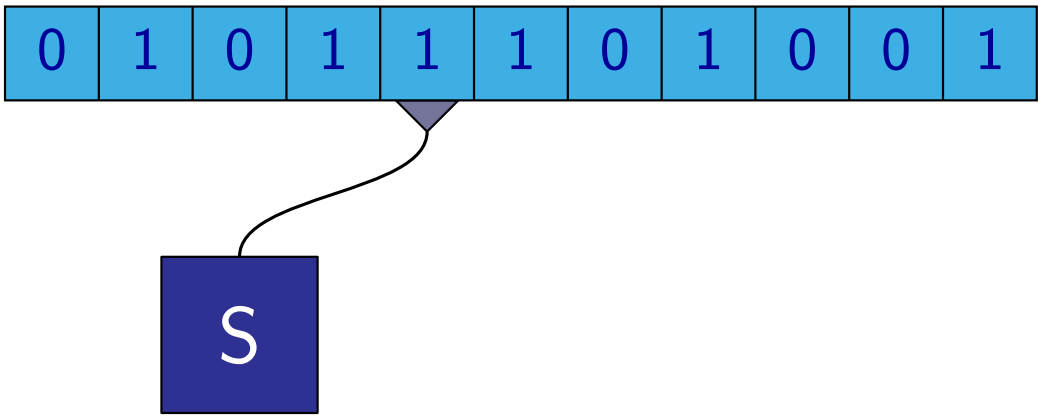
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



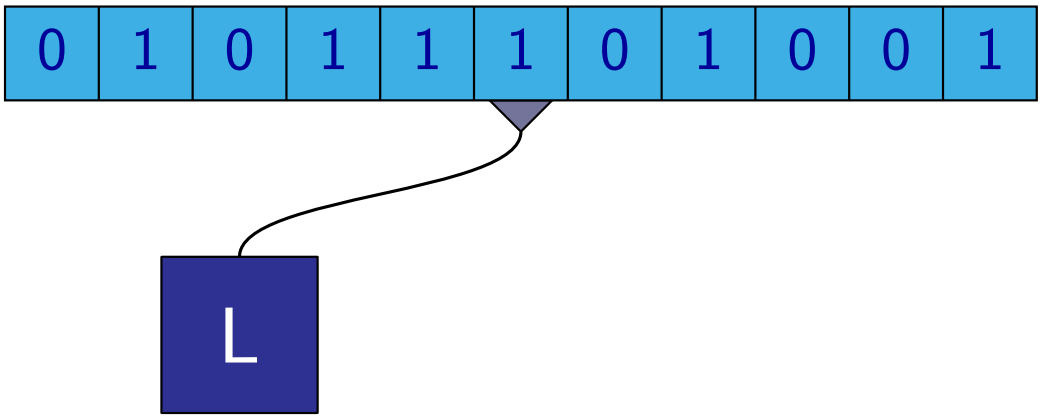
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



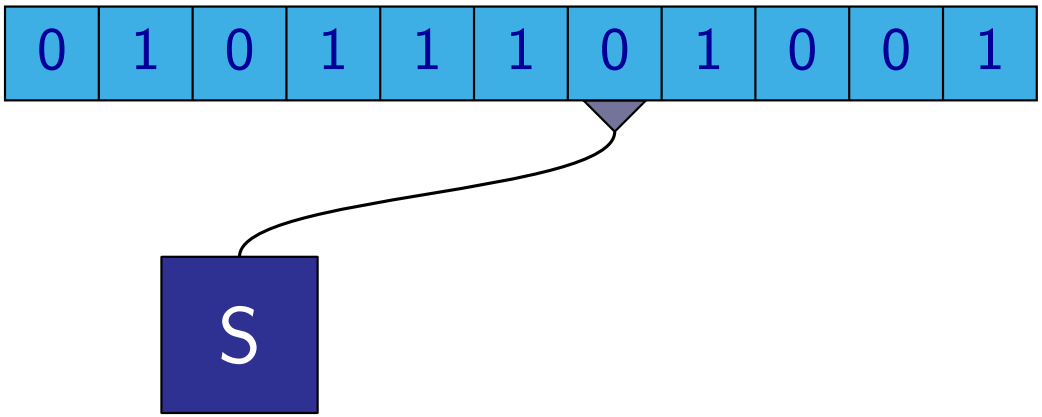
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



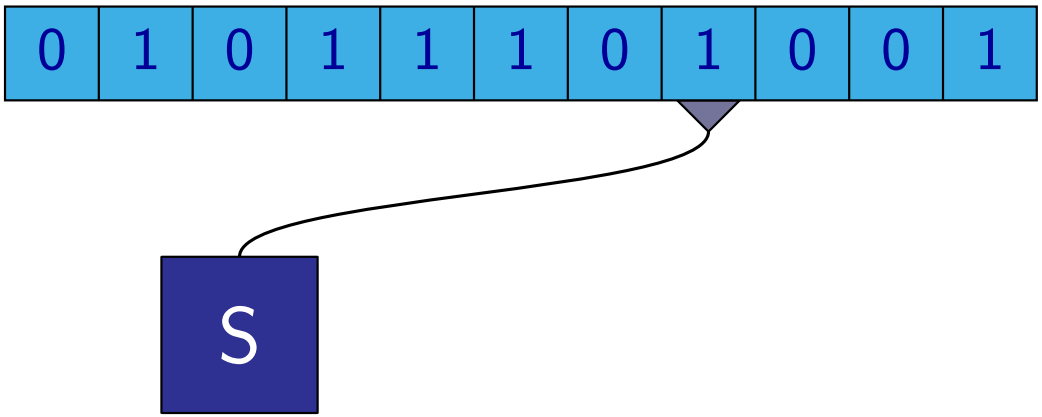
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



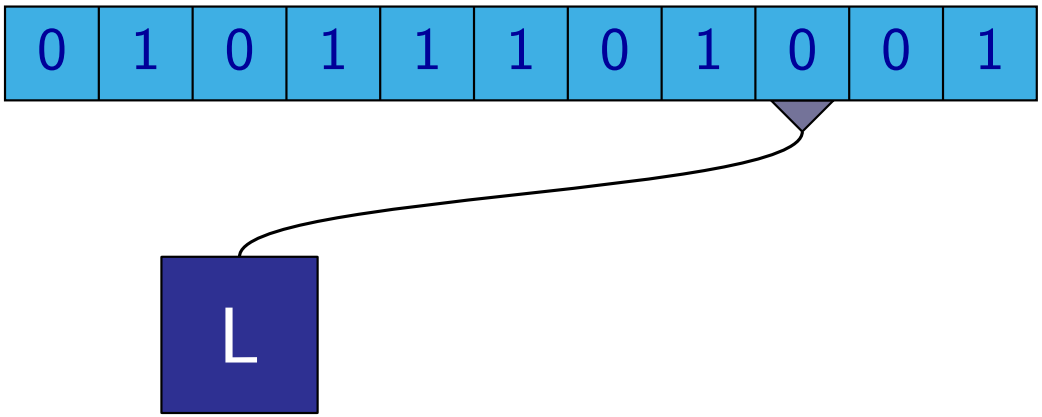
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



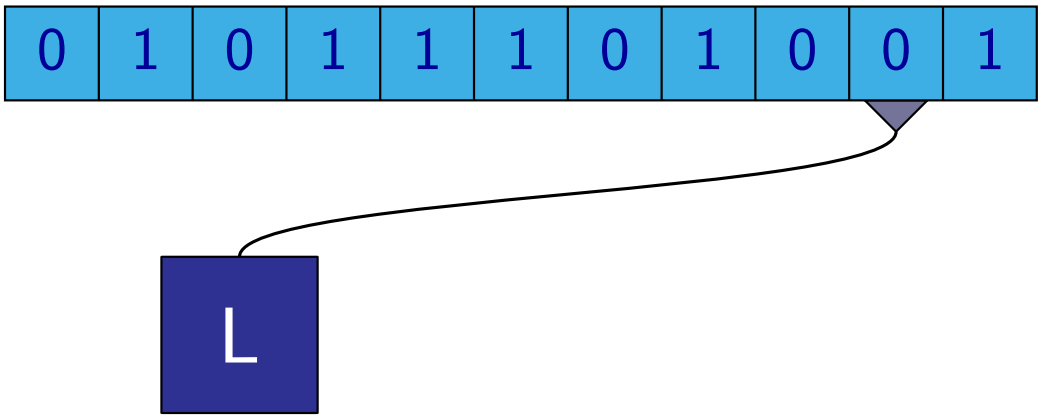
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



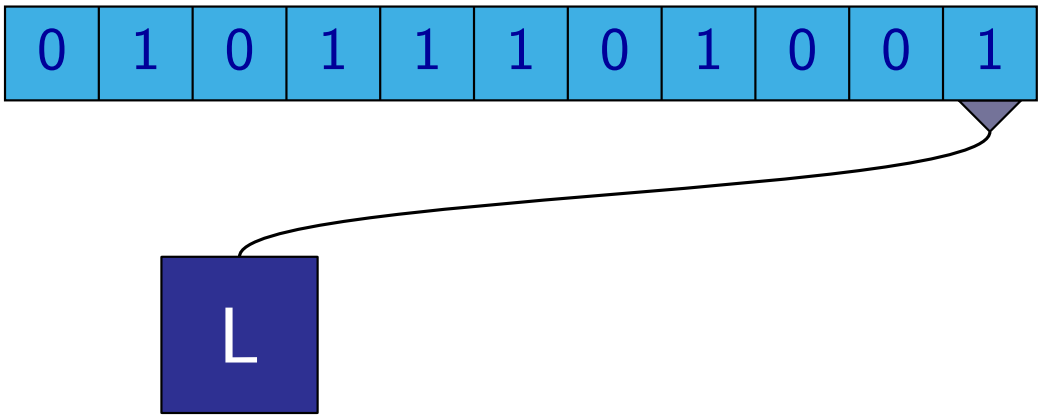
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



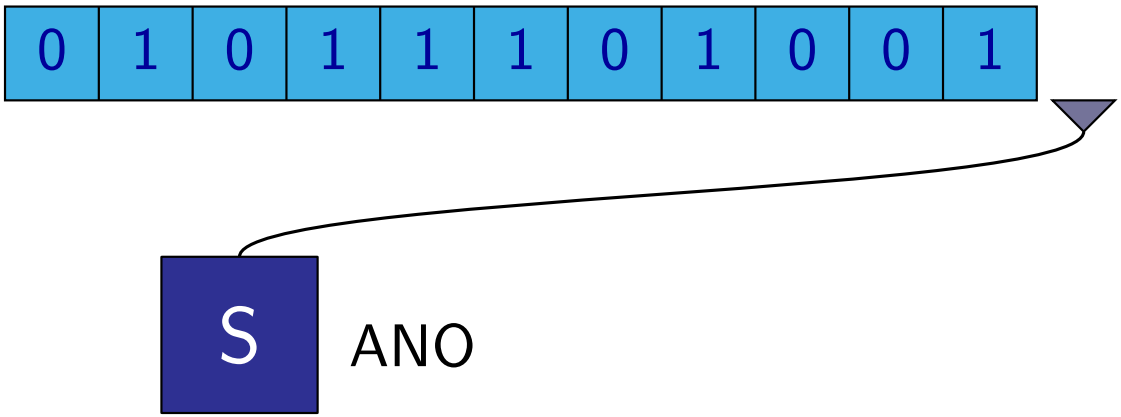
Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



Rozpoznávání jazyka

Druhý nápad: Ve skutečnosti nás zajímá pouze, zda počet dosud přečtených symbolů 1 je sudý nebo lichý (tj. místo čísla si stačí pamatovat jen jeho poslední bit).



Rozpoznávání jazyka

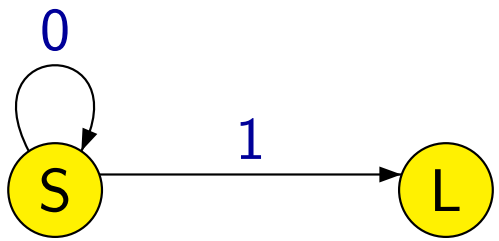
Chovaní tohoto zařízení můžeme popsat grafem:

Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:

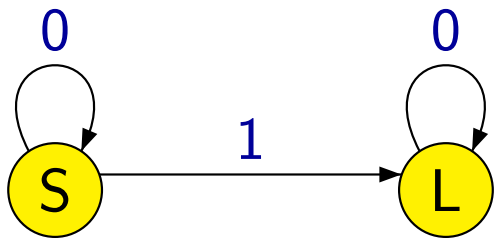
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



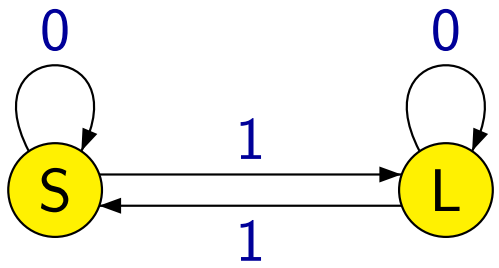
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



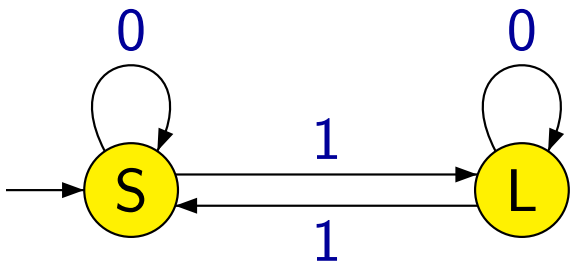
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



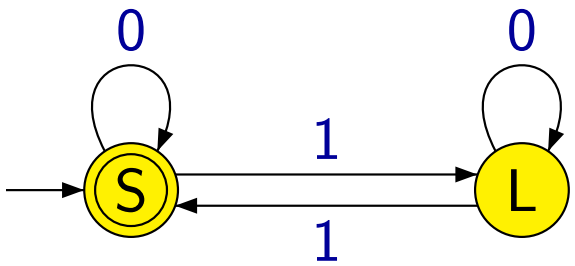
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:



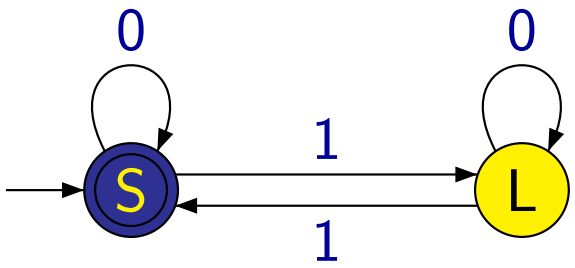
Rozpoznávání jazyka

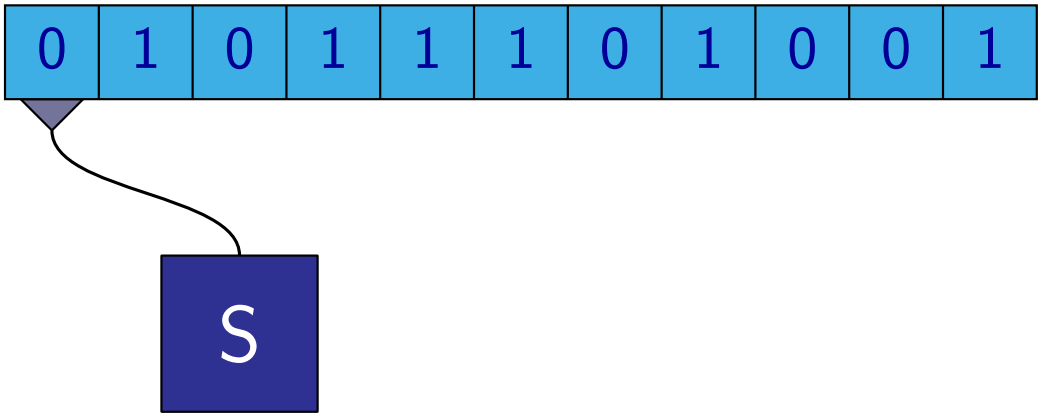
Chovaní tohoto zařízení můžeme popsat grafem:



Rozpoznávání jazyka

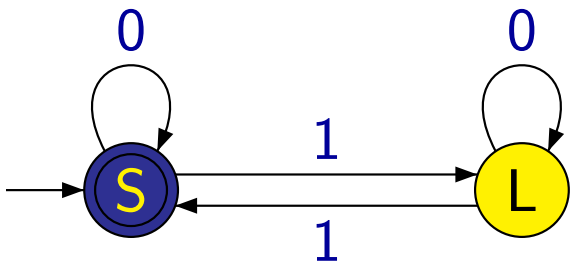
Chovaní tohoto zařízení můžeme popsat grafem:

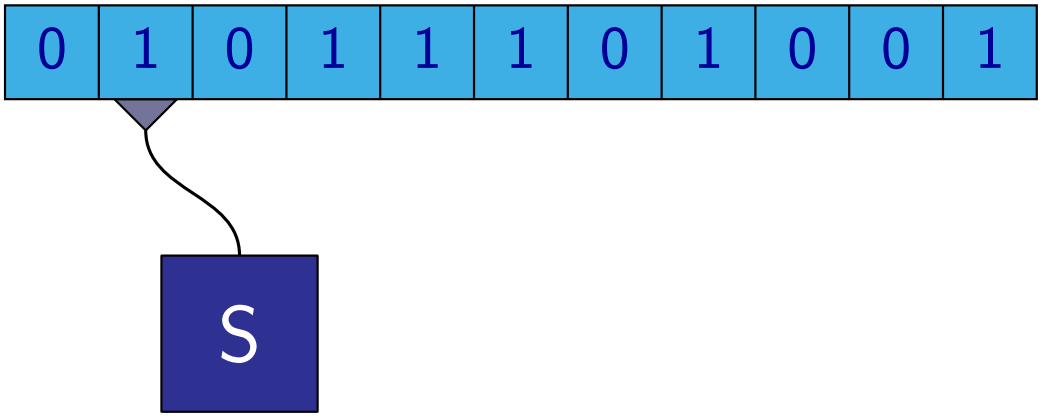




Rozpoznávání jazyka

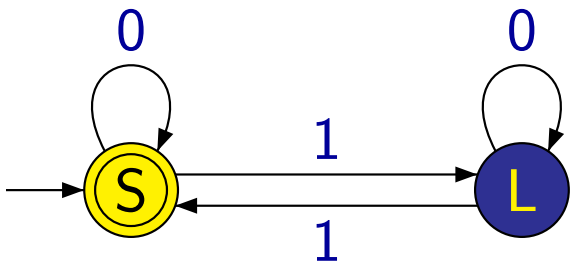
Chovaní tohoto zařízení můžeme popsat grafem:

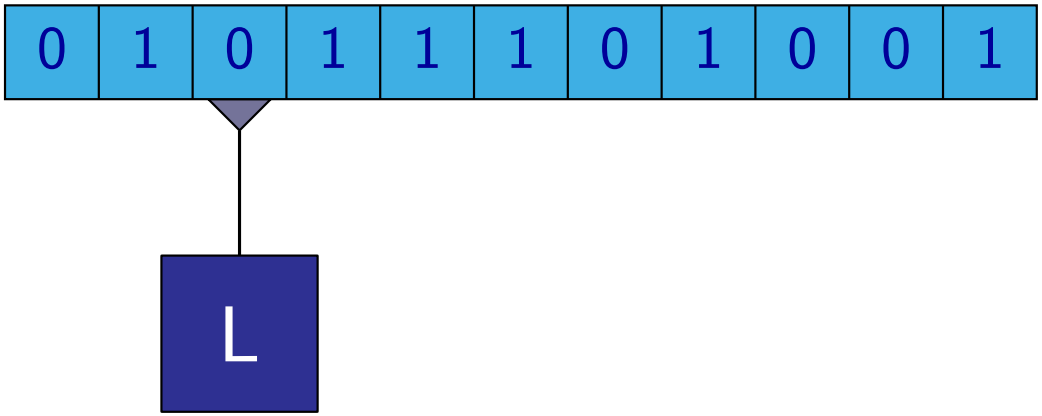




Rozpoznávání jazyka

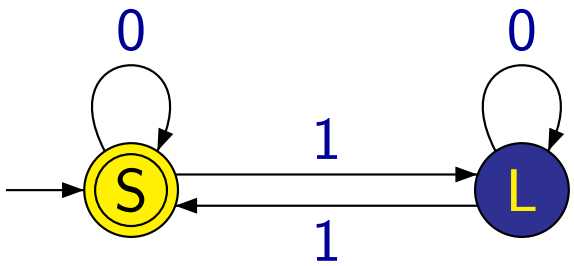
Chovaní tohoto zařízení můžeme popsat grafem:

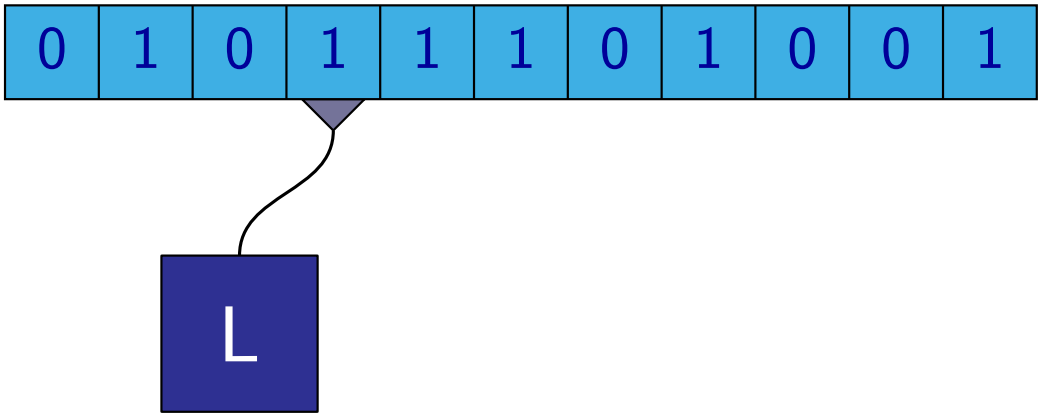




Rozpoznávání jazyka

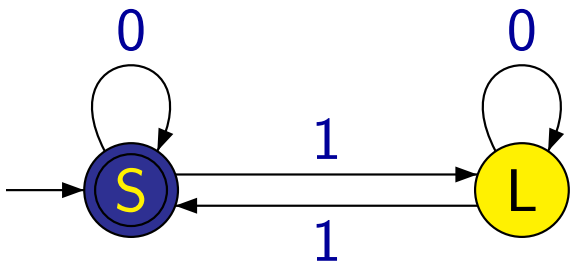
Chovaní tohoto zařízení můžeme popsat grafem:

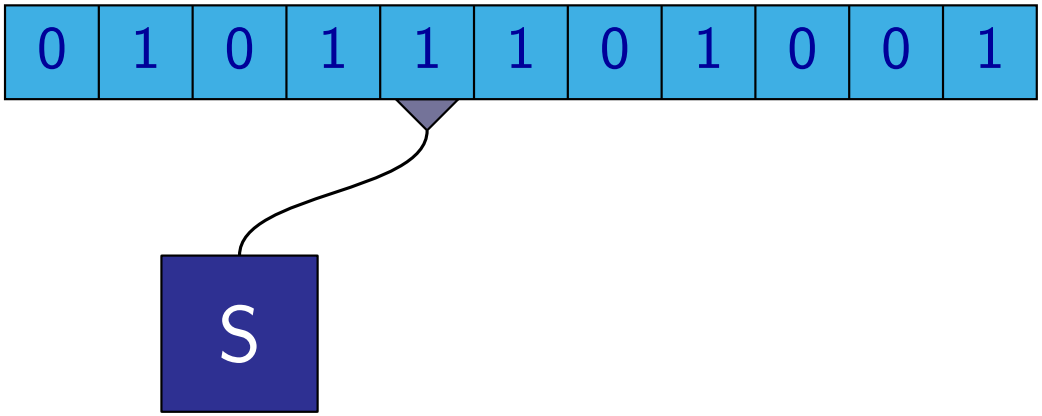




Rozpoznávání jazyka

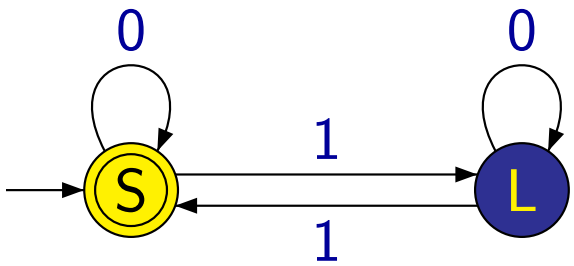
Chovaní tohoto zařízení můžeme popsat grafem:

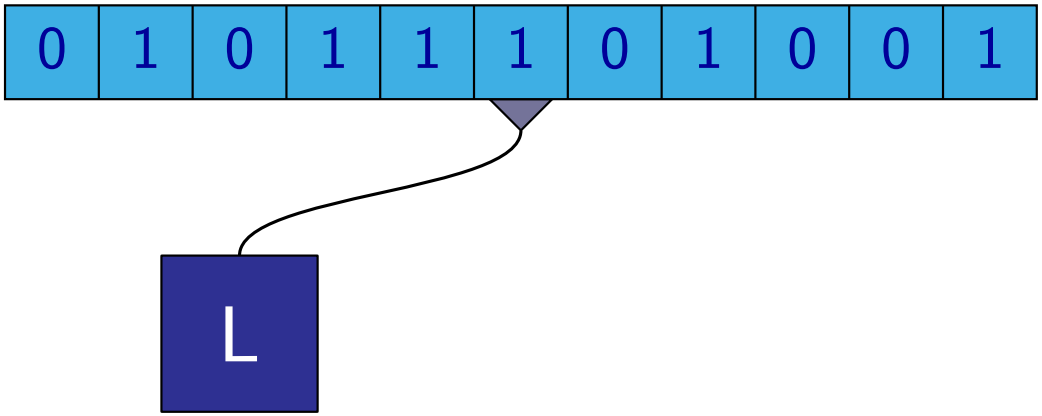




Rozpoznávání jazyka

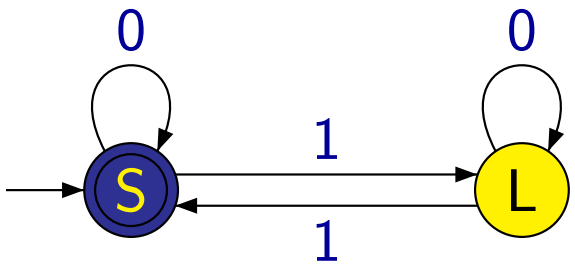
Chovaní tohoto zařízení můžeme popsat grafem:

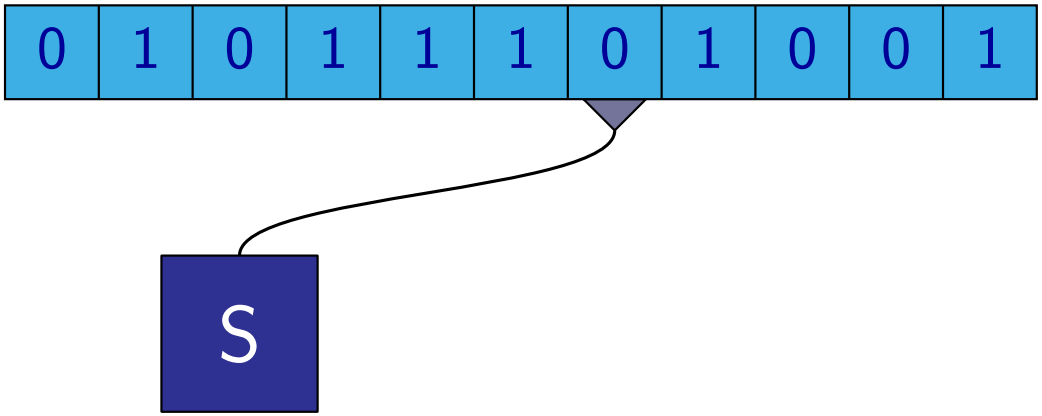




Rozpoznávání jazyka

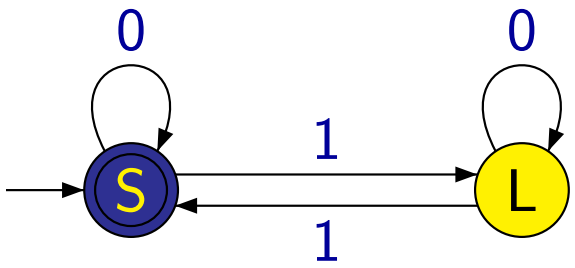
Chovaní tohoto zařízení můžeme popsat grafem:

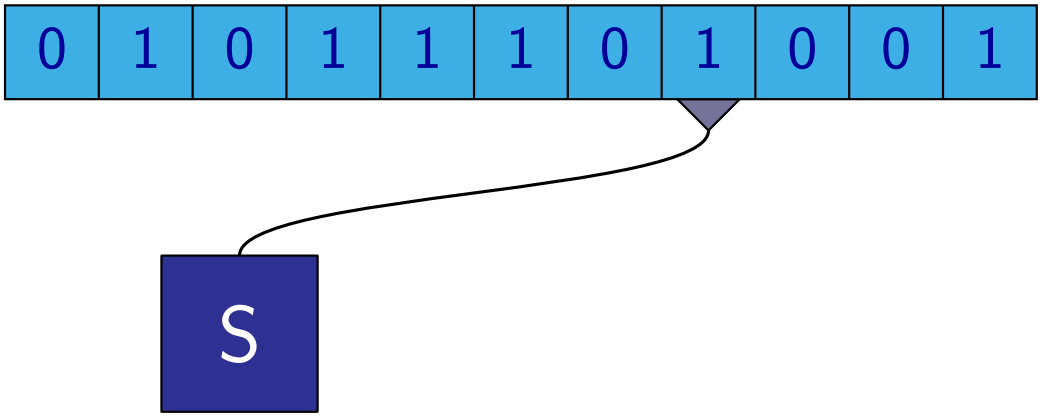




Rozpoznávání jazyka

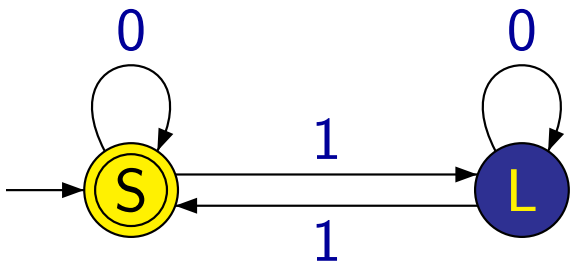
Chovaní tohoto zařízení můžeme popsat grafem:

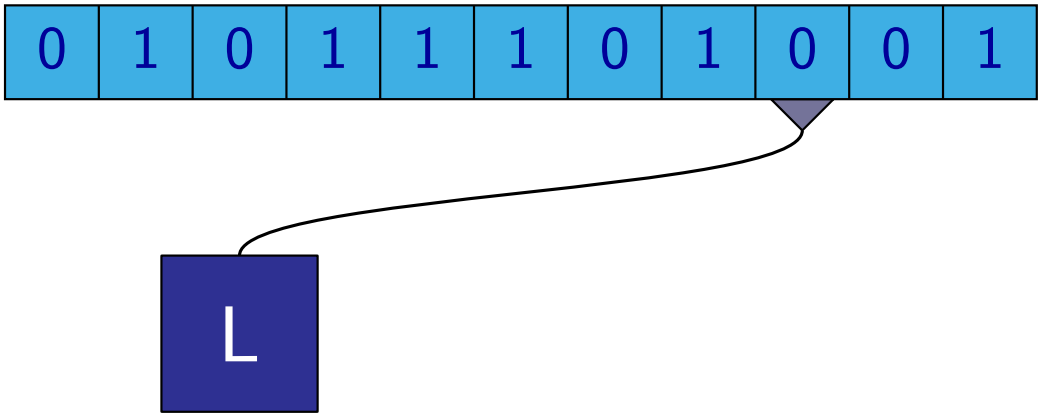




Rozpoznávání jazyka

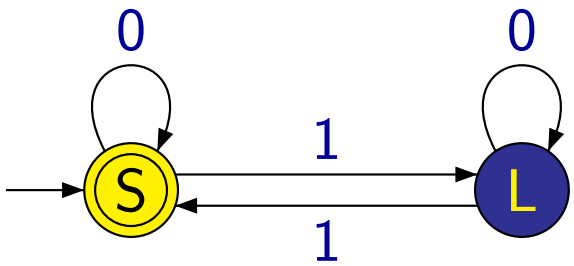
Chovaní tohoto zařízení můžeme popsat grafem:

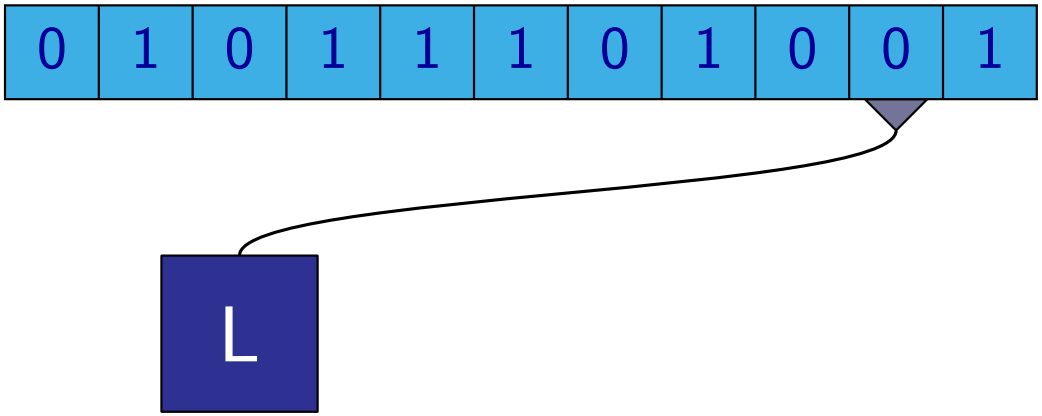




Rozpoznávání jazyka

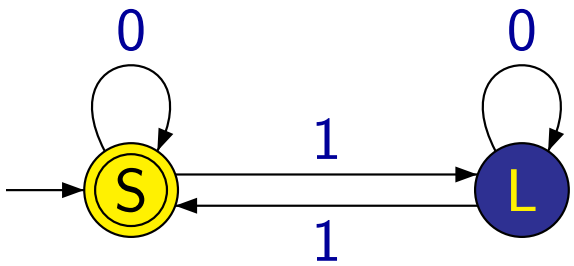
Chovaní tohoto zařízení můžeme popsat grafem:

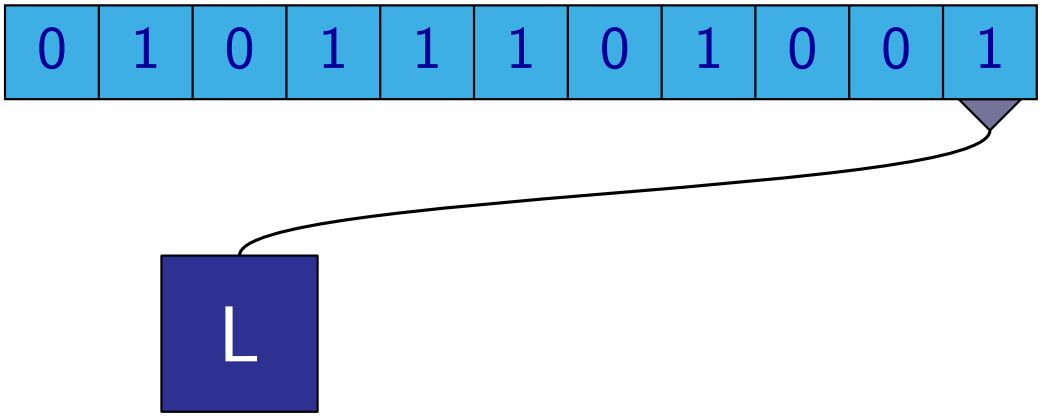




Rozpoznávání jazyka

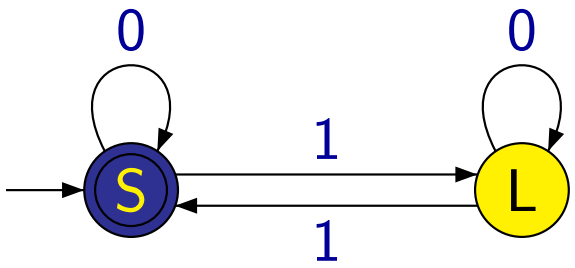
Chovaní tohoto zařízení můžeme popsat grafem:

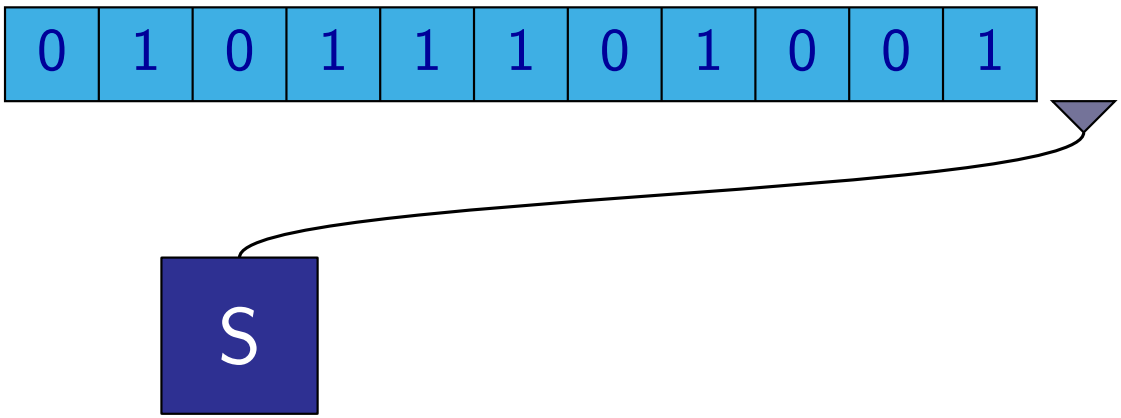




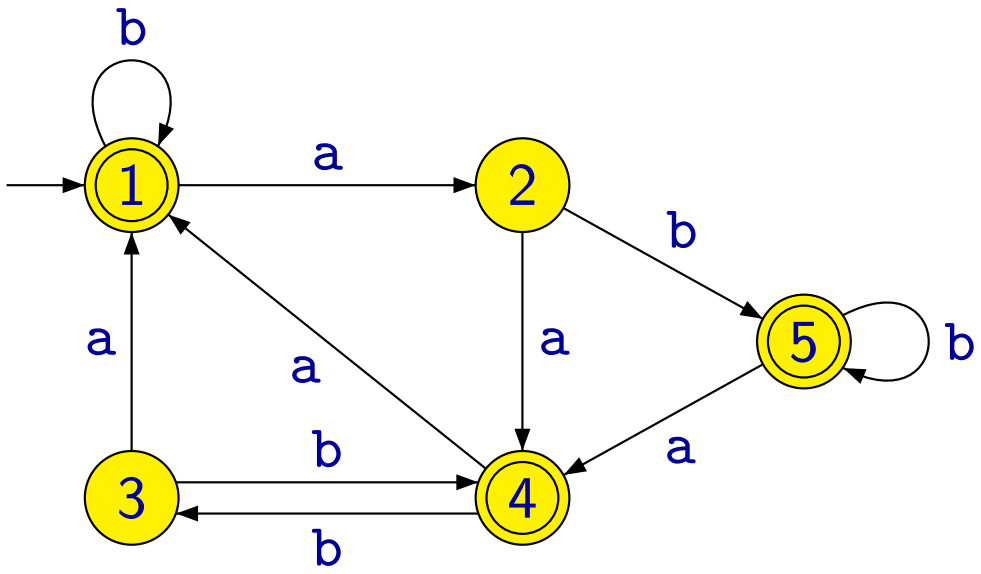
Rozpoznávání jazyka

Chovaní tohoto zařízení můžeme popsat grafem:





Deterministický konečný automat



Deterministický konečný automat se skládá ze **stavů** a **přechodů**. Jeden ze stavů je označen jako **počáteční stav** a některé ze stavů jsou označeny jako **přijímající**.

Deterministický konečný automat

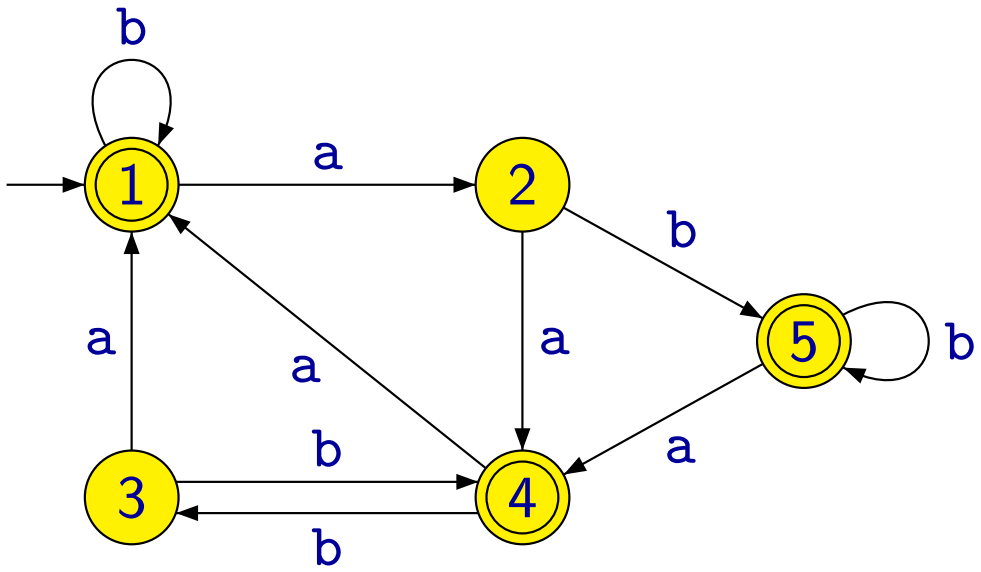
Formálně je **deterministický konečný automat** definován jako pětice

$$(Q, \Sigma, \delta, q_0, F)$$

kde:

- Q je neprázdná konečná množina **stavů**
- Σ je **abeceda** (neprázdná konečná množina symbolů)
- $\delta : Q \times \Sigma \to Q$ je **přechodová funkce**
- $q_0 \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **přijímajících stavů**

Deterministický konečný automat



- $Q = \{1, 2, 3, 4, 5\}$
- $\Sigma = \{\mathtt{a}, \mathtt{b}\}$
- $q_0 = 1$
- $F = \{1, 4, 5\}$

$$\delta(1, \mathtt{a}) = 2 \qquad \delta(1, \mathtt{b}) = 1$$
$$\delta(2, \mathtt{a}) = 4 \qquad \delta(2, \mathtt{b}) = 5$$
$$\delta(3, \mathtt{a}) = 1 \qquad \delta(3, \mathtt{b}) = 4$$
$$\delta(4, \mathtt{a}) = 1 \qquad \delta(4, \mathtt{b}) = 3$$
$$\delta(5, \mathtt{a}) = 4 \qquad \delta(5, \mathtt{b}) = 5$$

Deterministický konečný automat

Místo zápisu

$$\begin{array}{ll} \delta(1,\mathtt{a}) = 2 & \delta(1,\mathtt{b}) = 1 \\ \delta(2,\mathtt{a}) = 4 & \delta(2,\mathtt{b}) = 5 \\ \delta(3,\mathtt{a}) = 1 & \delta(3,\mathtt{b}) = 4 \\ \delta(4,\mathtt{a}) = 1 & \delta(4,\mathtt{b}) = 3 \\ \delta(5,\mathtt{a}) = 4 & \delta(5,\mathtt{b}) = 5 \end{array}$$

budeme raději používat stručnější tabulku nebo grafické znázornění:

δ	a	b
1	2	1
2	4	5
3	1	4
4	1	3
5	4	5

Deterministický konečný automat

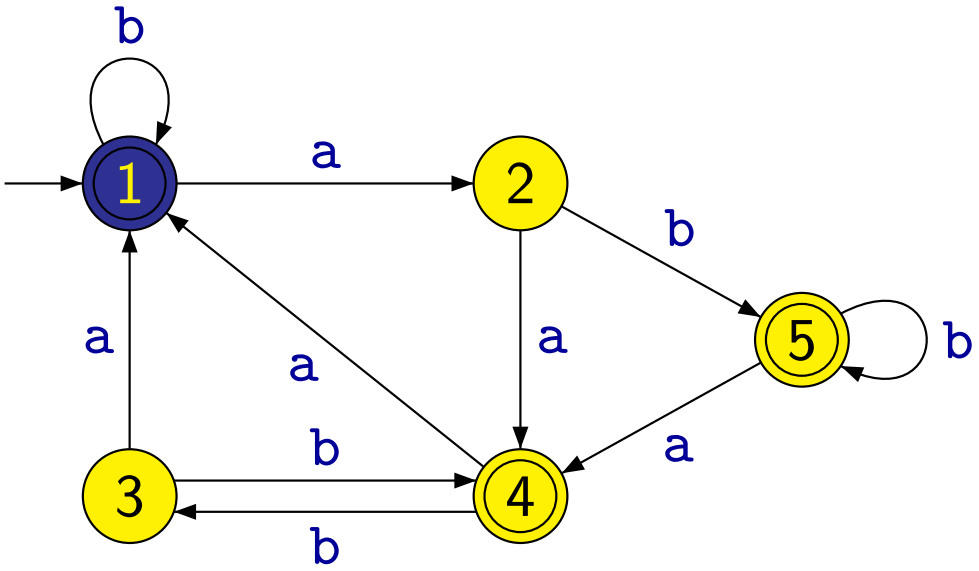
Místo zápisu

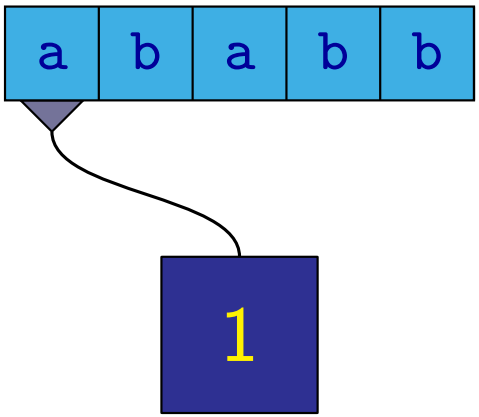
$$\begin{array}{ll} \delta(1,\mathtt{a}) = 2 & \delta(1,\mathtt{b}) = 1 \\ \delta(2,\mathtt{a}) = 4 & \delta(2,\mathtt{b}) = 5 \\ \delta(3,\mathtt{a}) = 1 & \delta(3,\mathtt{b}) = 4 \\ \delta(4,\mathtt{a}) = 1 & \delta(4,\mathtt{b}) = 3 \\ \delta(5,\mathtt{a}) = 4 & \delta(5,\mathtt{b}) = 5 \end{array}$$

budeme raději používat stručnější tabulku nebo grafické znázornění:

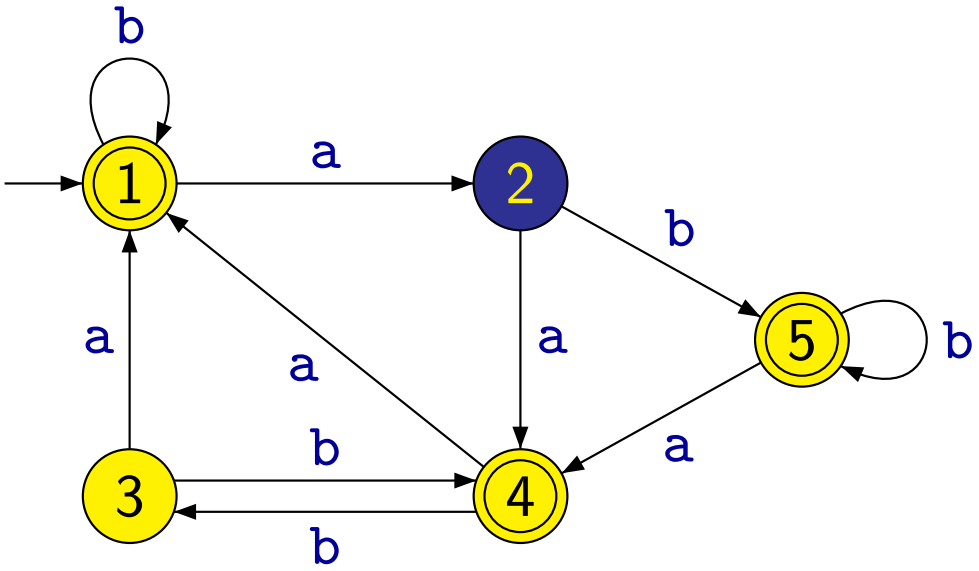
δ	a	b
$\leftrightarrow 1$	2	1
2	4	5
3	1	4
$\leftarrow 4$	1	3
$\leftarrow 5$	4	5

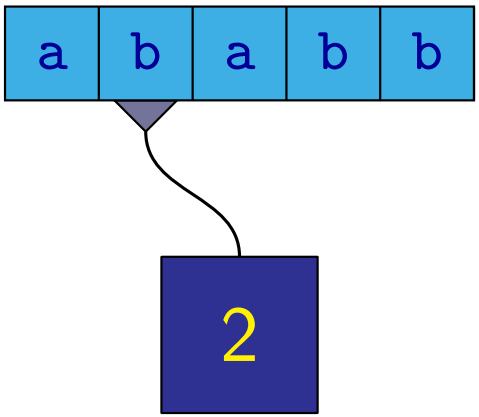
Deterministický konečný automat



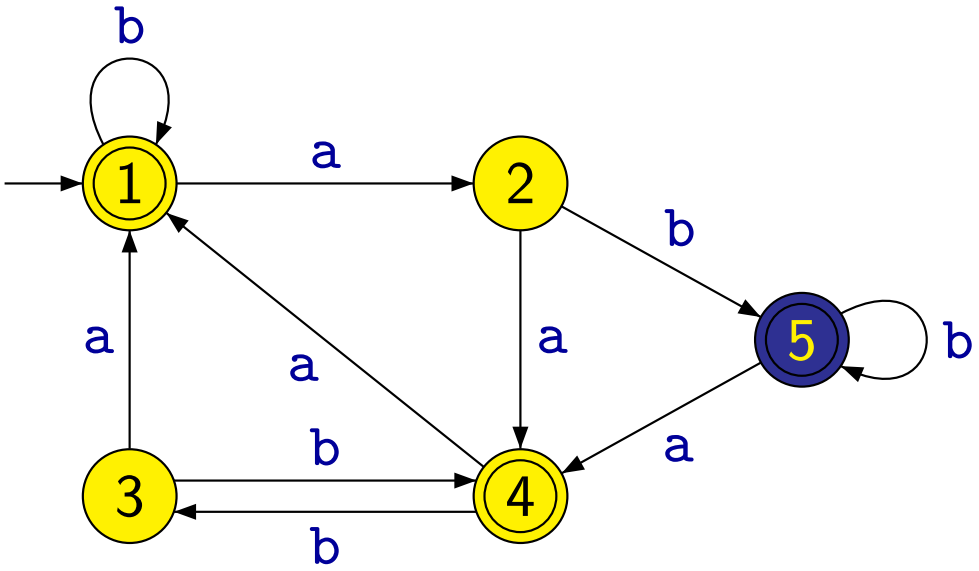


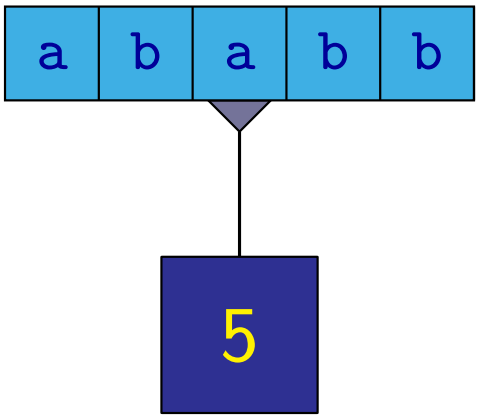
Deterministický konečný automat



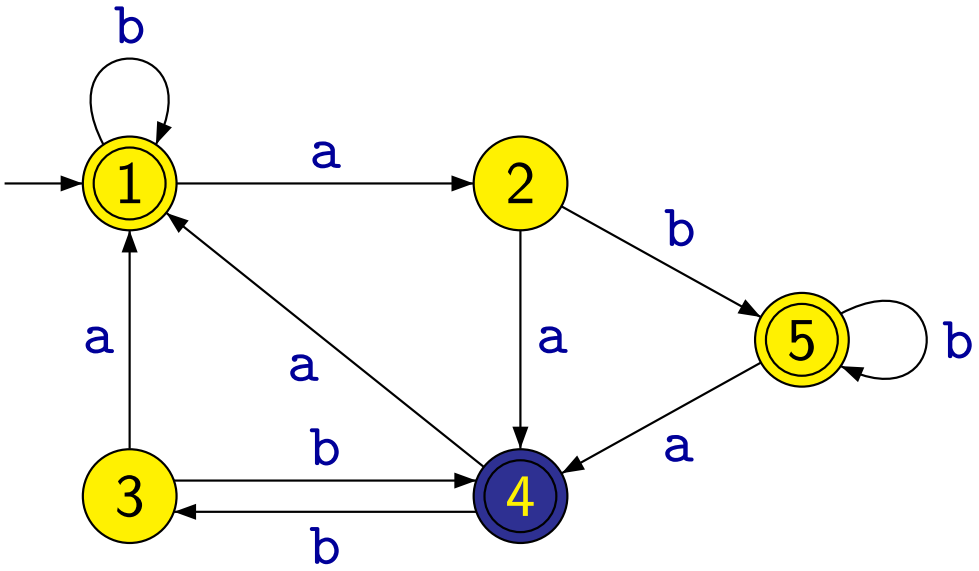


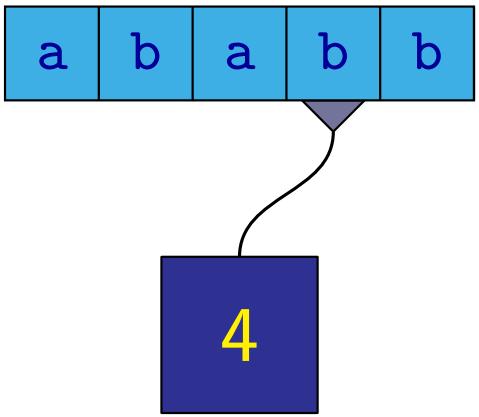
Deterministický konečný automat



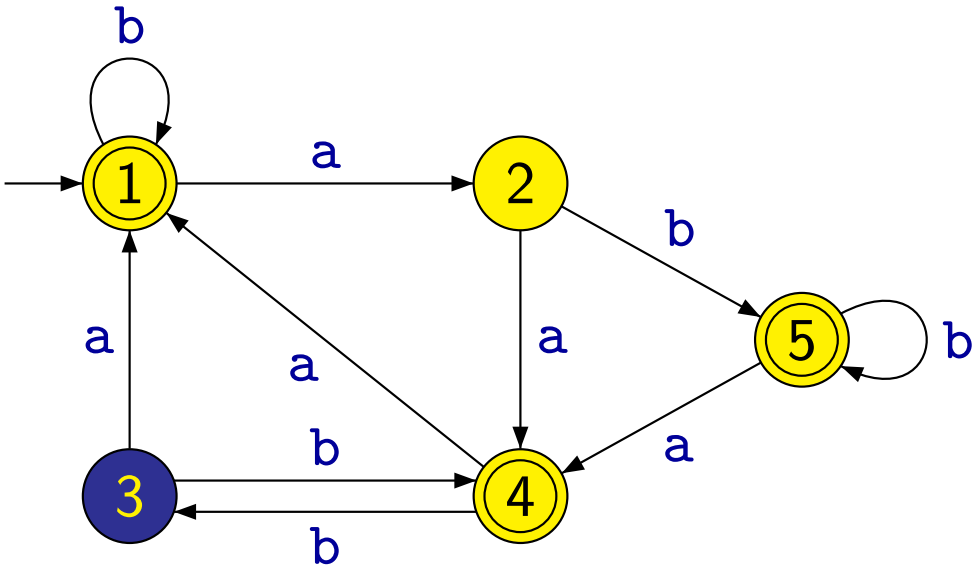


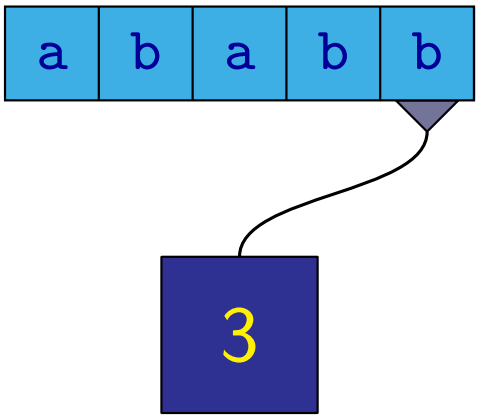
Deterministický konečný automat



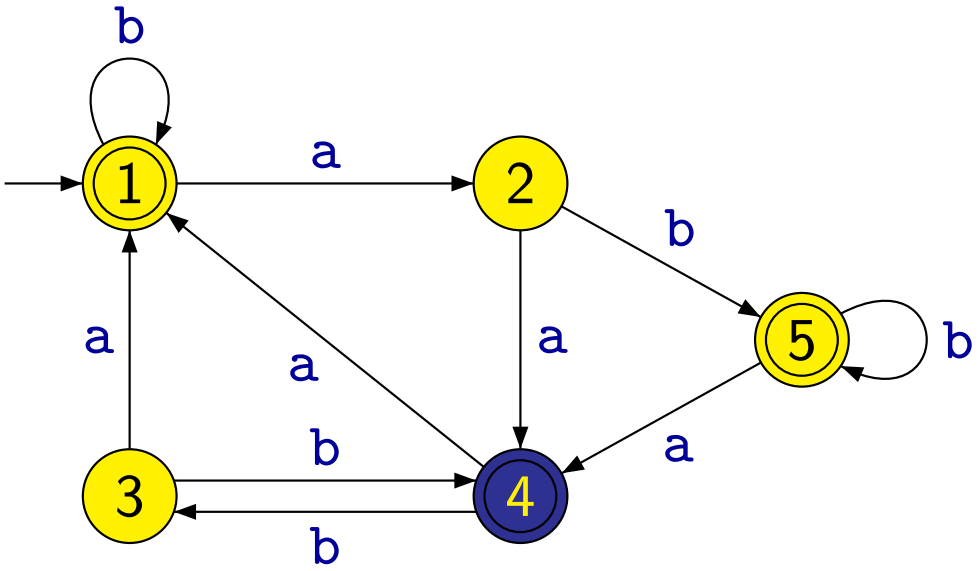


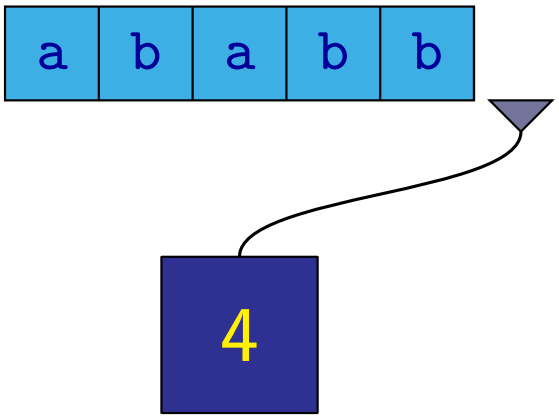
Deterministický konečný automat





Deterministický konečný automat





Deterministický konečný automat

Konfigurace konečného automatu A je dána stavem jeho řídící jednotky a dosud nepřečteným obsahem pásky.

Množina všech konfigurací $Conf$ je $Q \times \Sigma^*$.

Příklad: $(2, \mathtt{babb}) \in Conf$

Pro daný automat A můžeme na množině všech konfigurací definovat binární relaci $\vdash \subseteq Conf \times Conf$ s následujícím významem:
$C_1 \vdash C_2$ znamená, že automat A může přejít jedním krokem z konfigurace C_1 do konfigurace C_2.

Příklad:

$$(2, \mathtt{babb}) \vdash (5, \mathtt{abb})$$

Formálně platí, že $(q, w) \vdash (q', w')$ právě tehdy, když $w = aw'$ a $q' = \delta(q, a)$ pro nějaké $a \in \Sigma$.

Deterministický konečný automat

Konfigurace (q, w) se nazývá **počáteční konfigurace**, jestliže $q = q_0$.

Příklad: $(1, \texttt{ababb})$ je počáteční konfigurace.

Konfigurace (q, w) se nazývá **koncová konfigurace**, jestliže $w = \varepsilon$.

Příklad: $(4, \varepsilon)$ je koncová konfigurace.

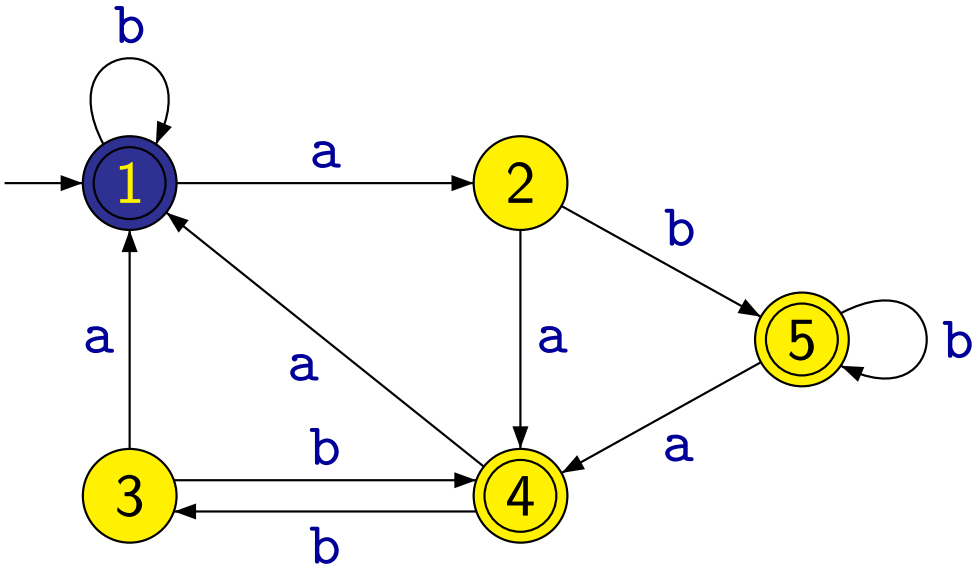
Definice

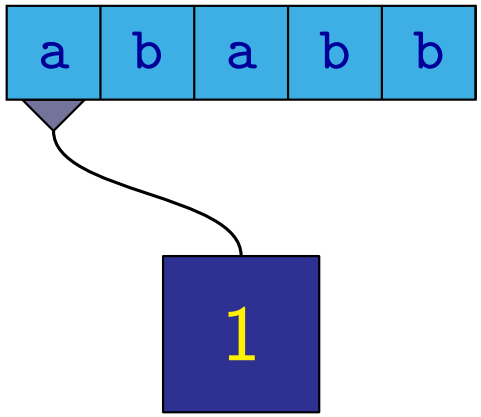
Výpočet automatu je posloupnost konfigurací

$$C_0, C_1, C_2, \cdots, C_k$$

kde C_i jsou konfigurace, C_0 je počáteční konfigurace, C_k je koncová konfigurace a pro všechna $i \in \{1, 2, \ldots, k\}$ platí, že $C_{i-1} \vdash C_i$.

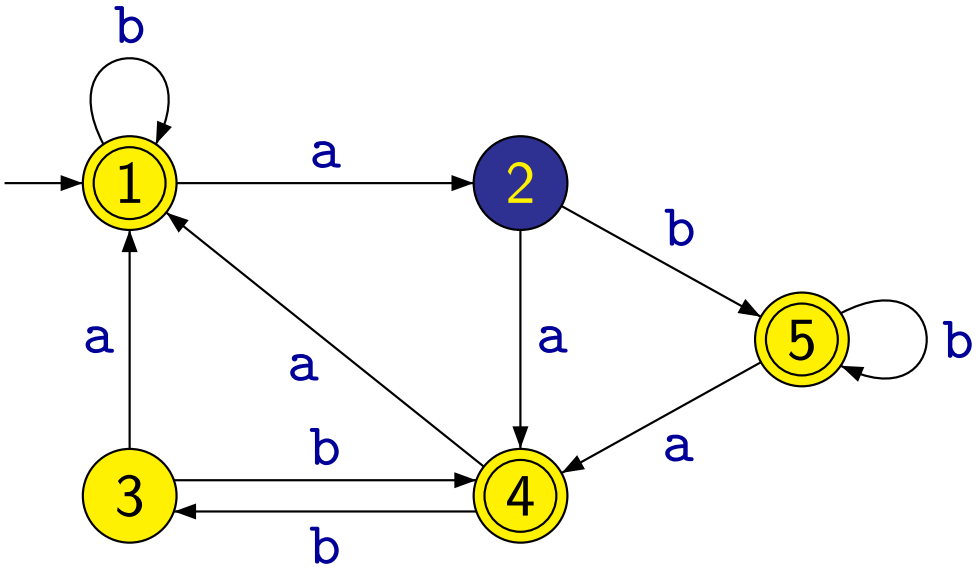
Deterministický konečný automat

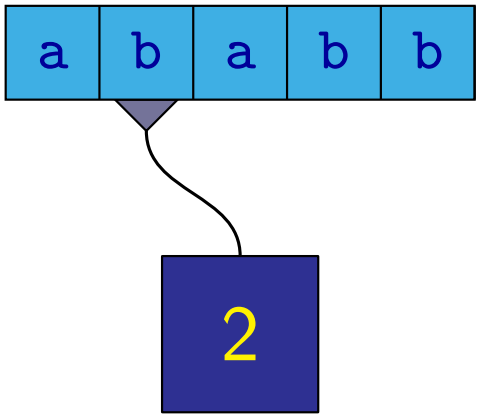




$(1, \mathtt{ababb})$

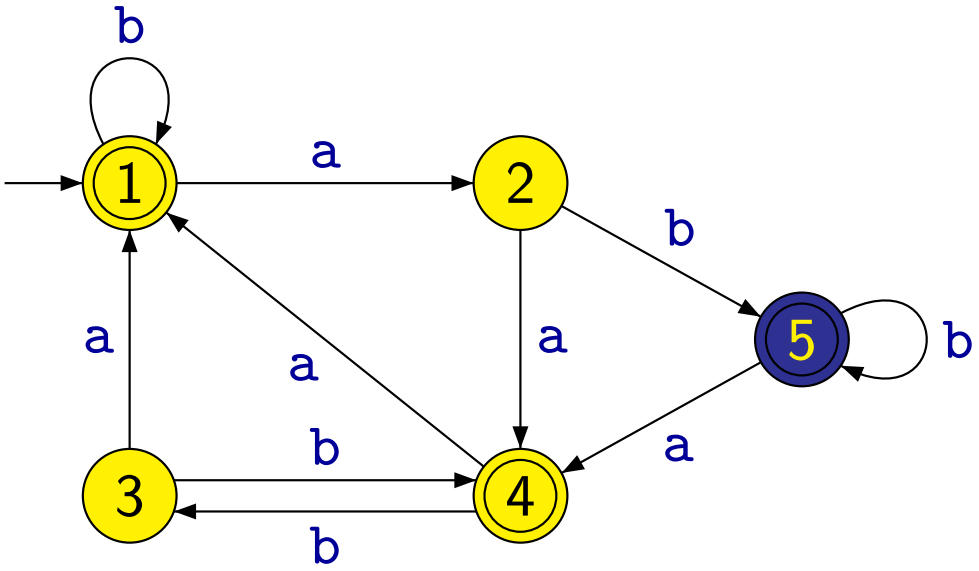
Deterministický konečný automat

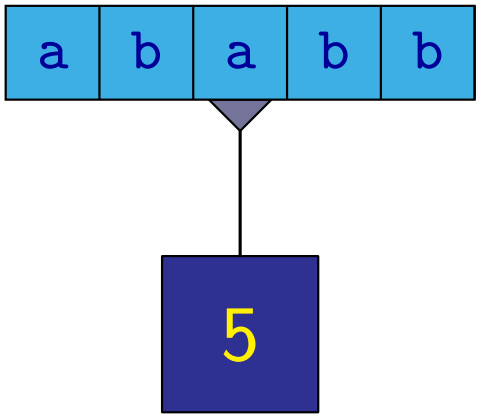




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb})$

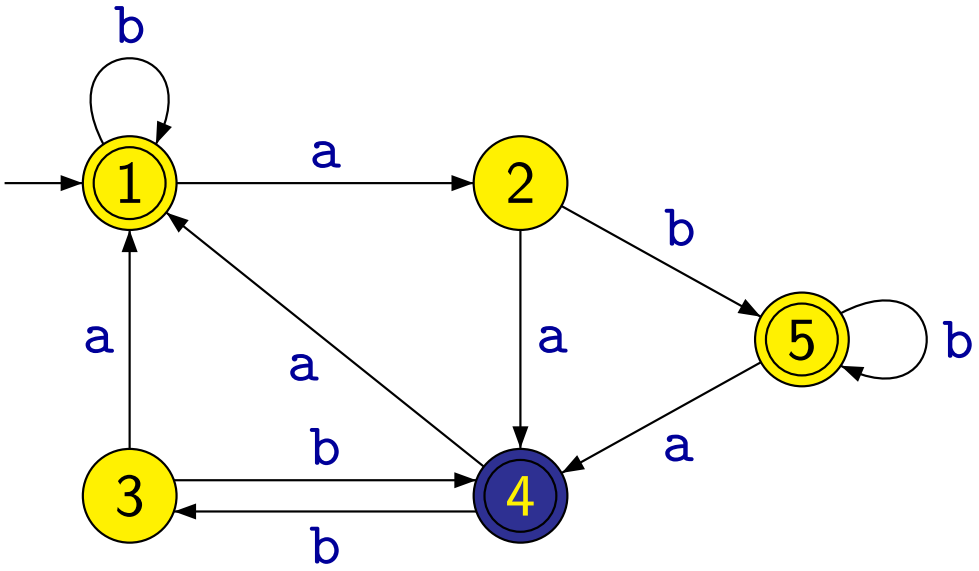
Deterministický konečný automat

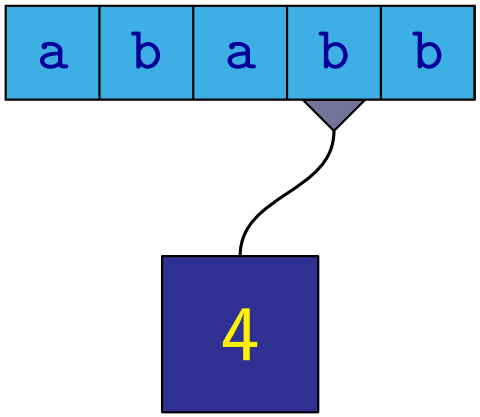




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash (5, \mathtt{abb})$

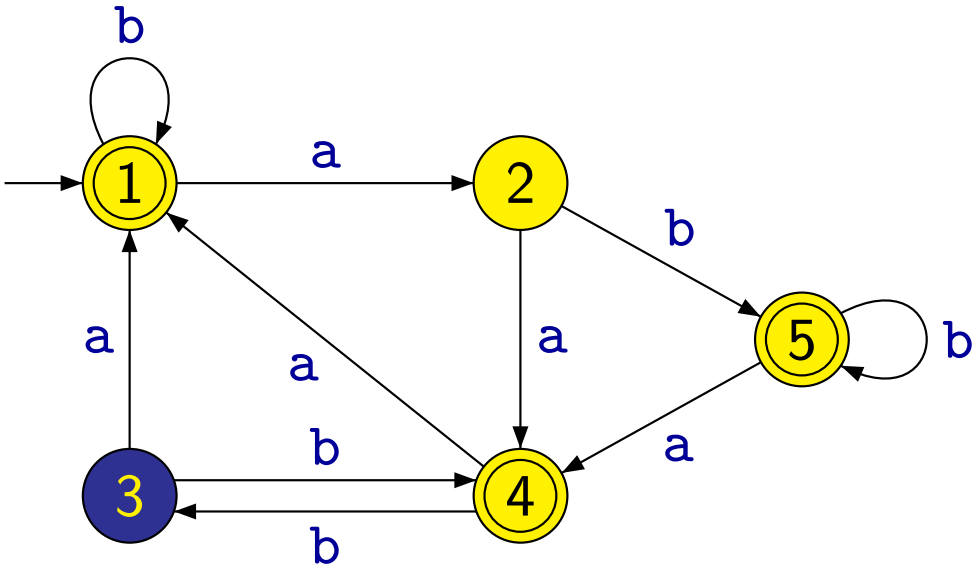
Deterministický konečný automat

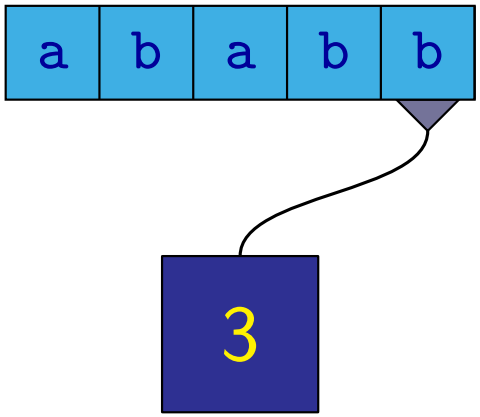




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash$
$(5, \mathtt{abb}) \vdash (4, \mathtt{bb})$

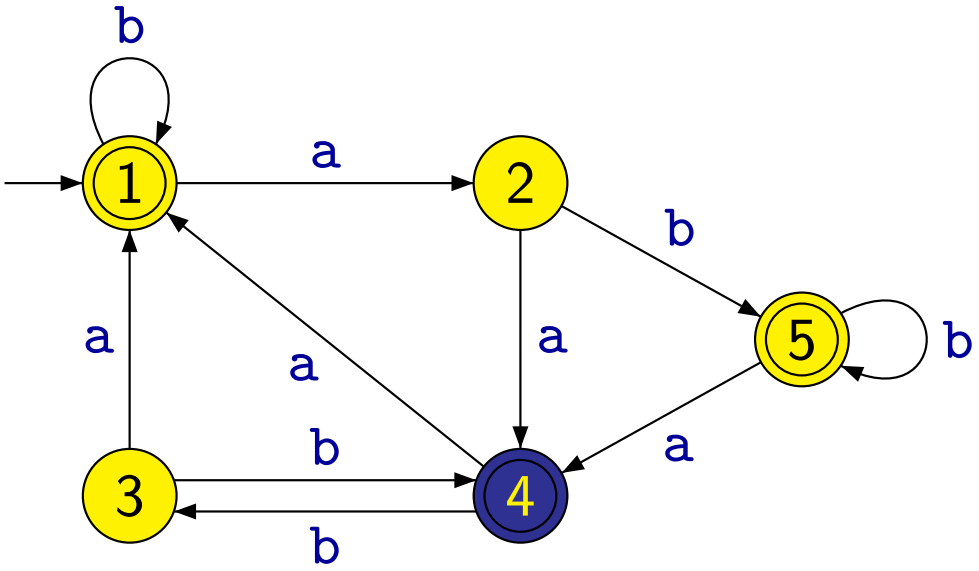
Deterministický konečný automat

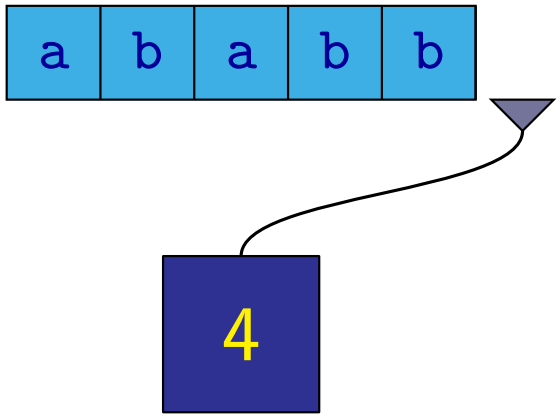




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash$
$(5, \mathtt{abb}) \vdash (4, \mathtt{bb}) \vdash$
$(3, \mathtt{b})$

Deterministický konečný automat





$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash$
$(5, \mathtt{abb}) \vdash (4, \mathtt{bb}) \vdash$
$(3, \mathtt{b}) \vdash (4, \varepsilon)$

Deterministický konečný automat

Dále můžeme definovat pro daný automat A relaci

$$\vdash^* \subseteq \mathit{Conf} \times \mathit{Conf},$$

takovou, že $C \vdash^* C'$ platí právě tehdy, když automat může přejít nějakým libovolným (i nulovým) počtem kroků z konfigurace C do konfigurace C'.

Přesněji řečeno, $C \vdash^* C'$ platí právě tehdy, když existuje posloupnost konfigurací

$$C_0,\ C_1,\ C_2,\ \ldots,\ C_k$$

kde $C_0 = C$, $C_k = C'$ a pro všechna $i \in \{1, 2, \ldots, k\}$ platí, že $C_{i-1} \vdash C_i$.

Deterministický konečný automat

Alternativně můžeme relaci $\vdash^*$ definovat jako nejmenší relaci splňující následující podmínky:

- Pro každé $C \in \mathit{Conf}$ platí $C \vdash^* C$.
- Pro každé $C, C' \in \mathit{Conf}$ platí, že pokud $C \vdash C'$, pak $C \vdash^* C'$.
- Pro každé $C, C', C'' \in \mathit{Conf}$ platí, že pokud $C \vdash^* C'$ a $C' \vdash^* C''$, pak $C \vdash^* C''$.

Poznámka: Relace $\vdash^*$ je **reflexivním a trazitivním uzávěrem** relace $\vdash$, tj. je to nejmenší reflexivní a trazitivní relace obsahující relaci $\vdash$.

Deterministický konečný automat

Definice

Koncová konfigurace (q, ε) je **přijímající**, jestliže $q \in F$.

Definice

Automat $A = (Q, \Sigma, \delta, q_0, F)$ **přijímá** slovo $w \in \Sigma^*$ právě tehdy, když $(q_0, w) \vdash^* (q, \varepsilon)$ pro nějaké $q \in F$.

Tj. automat A přijímá slovo w právě tehdy, když výpočet automatu A začínající v počáteční konfiguraci (q_0, w) skončí v přijímající koncové konfiguraci.

Deterministický konečný automat

Definice

Jazyk rozpoznávaný (přijímaný) daným deterministickým konečným automatem $A = (Q, \Sigma, \delta, q_0, F)$, označovaný $L(A)$, je množina všech slov přijímaných tímto automatem, tj.

$$L(A) = \{w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \varepsilon),\ q \in F\}$$

Definice

Jazyk L je **regulární** právě tehdy, když existuje nějaký konečný automat, který jej přijímá.

Deterministický konečný automat

Definice

Mějme DKA $A = (Q, \Sigma, \delta, q_0, F)$.

Zápisem $q \xrightarrow{w} q'$, kde $q, q' \in Q$ a $w \in \Sigma^*$, budeme označovat to, že pokud je automat ve stavu q, tak přečtením slova w přejde do stavu q'.

Poznámka: $\longrightarrow \subseteq Q \times \Sigma^* \times Q$ je ternární relace.

Místo $(q, w, q') \in \longrightarrow$ píšeme $q \xrightarrow{w} q'$.

Tuto relaci můžeme formálně definovat následující induktivní definicí:

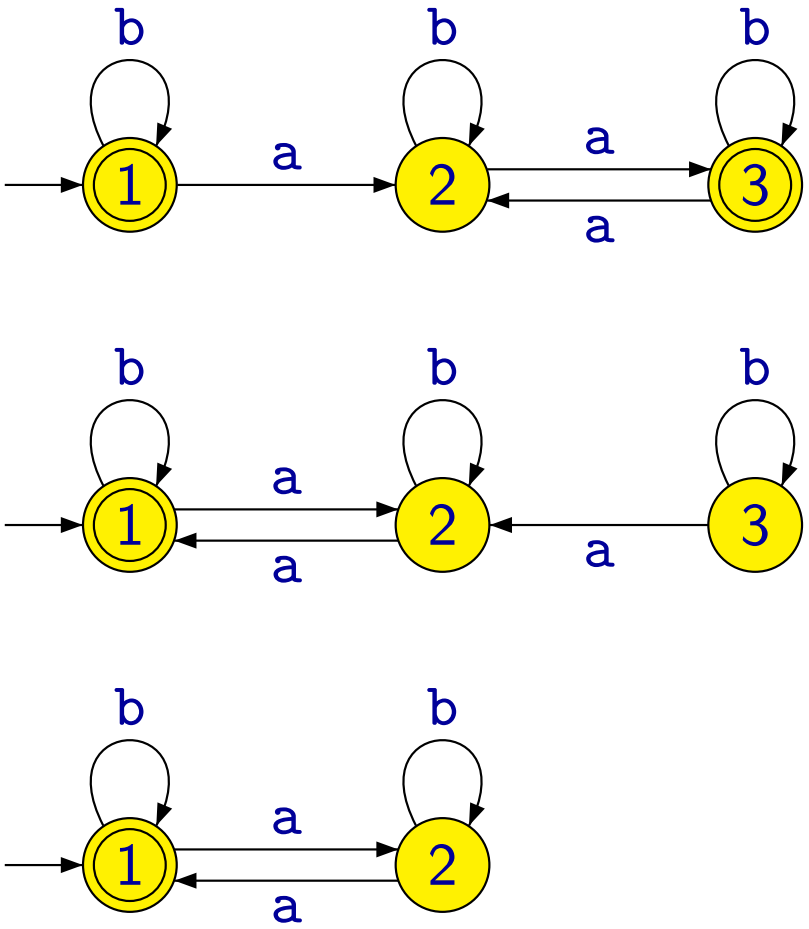
- $q \xrightarrow{\varepsilon} q$ pro libovolné $q \in Q$
- Pro $w \in \Sigma^*$ a $a \in \Sigma$:
  $q \xrightarrow{wa} q'$ právě tehdy, když existuje $q'' \in Q$ takové, že $q \xrightarrow{w} q''$ a $\delta(q'', a) = q'$.

Deterministický konečný automat

Jazyk rozpoznávaný daným DKA $A = (Q, \Sigma, \delta, q_0, F)$ pak můžeme definovat (alternativně) takto:

$$L(A) = \{w \in \Sigma^* \mid \exists q \in F : q_0 \xrightarrow{w} q\}$$

Ekvivalence automatů



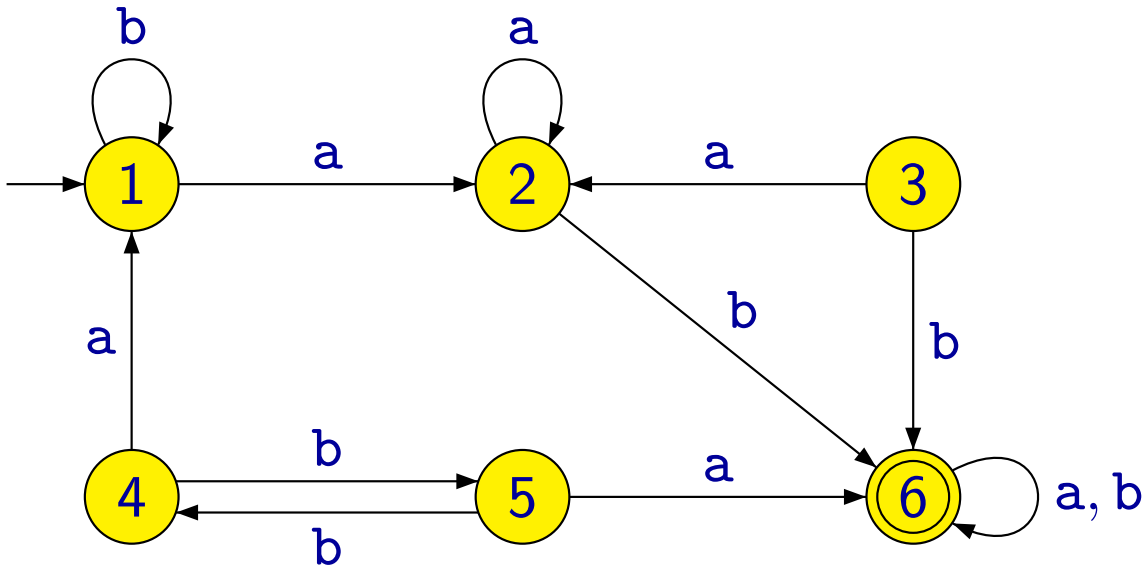
- Všechny 3 automaty přijímají jazyk všech slov se sudým počtem *a*.
- Nejvýhodnější je pro nás poslední z nich – má nejméně stavů.

Ekvivalence automatů

Definice

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

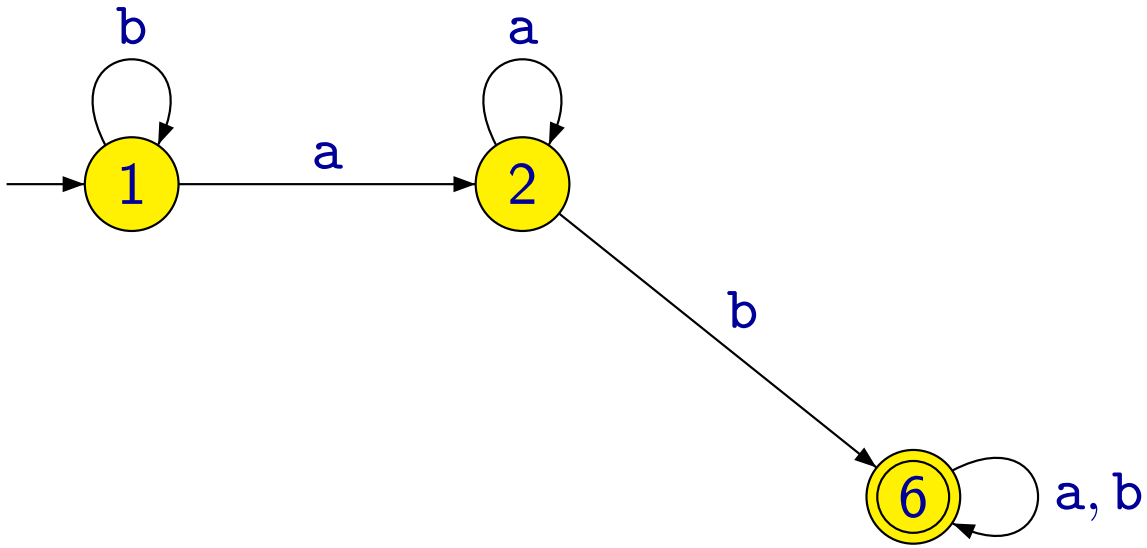
Nedosažitelné stavy automatu



- 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.

Nedosažitelné stavy automatu

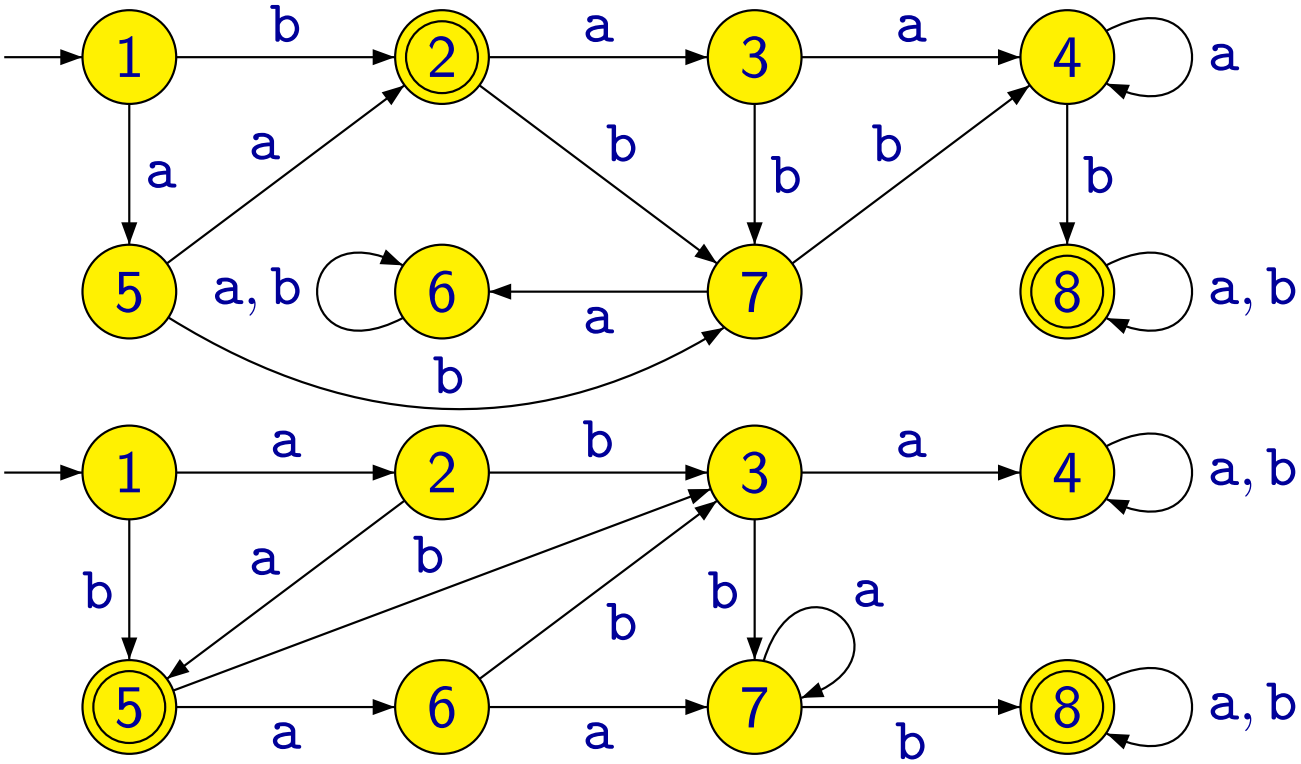
Definice

Stav q konečného automatu $A = (Q, \Sigma, \delta, q_0, F)$ je **dosažitelný** pokud existuje nějaké slovo w takové, že $q_0 \xrightarrow{w} q$.

V opačném případě stav nazýváme **nedosažitelný**.

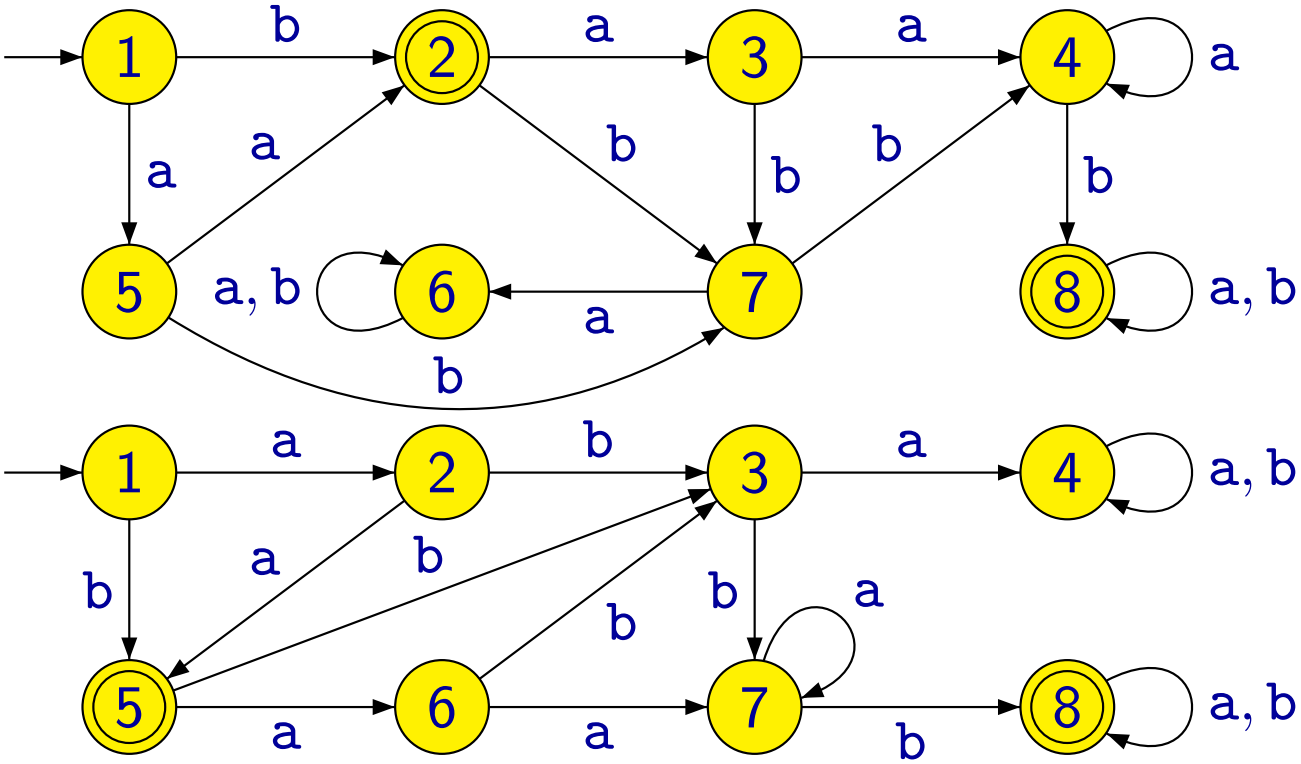
- Do nedosažitelných stavů nevede v grafu automatu žádná orientovaná cesta z počátečního stavu.
- Nedosažitelné stavy můžeme z automatu odstranit (spolu se všemi přechody vedoucími do nich a z nich). Jazyk přijímaný automatem se nezmění.

Normovaný tvar automatu



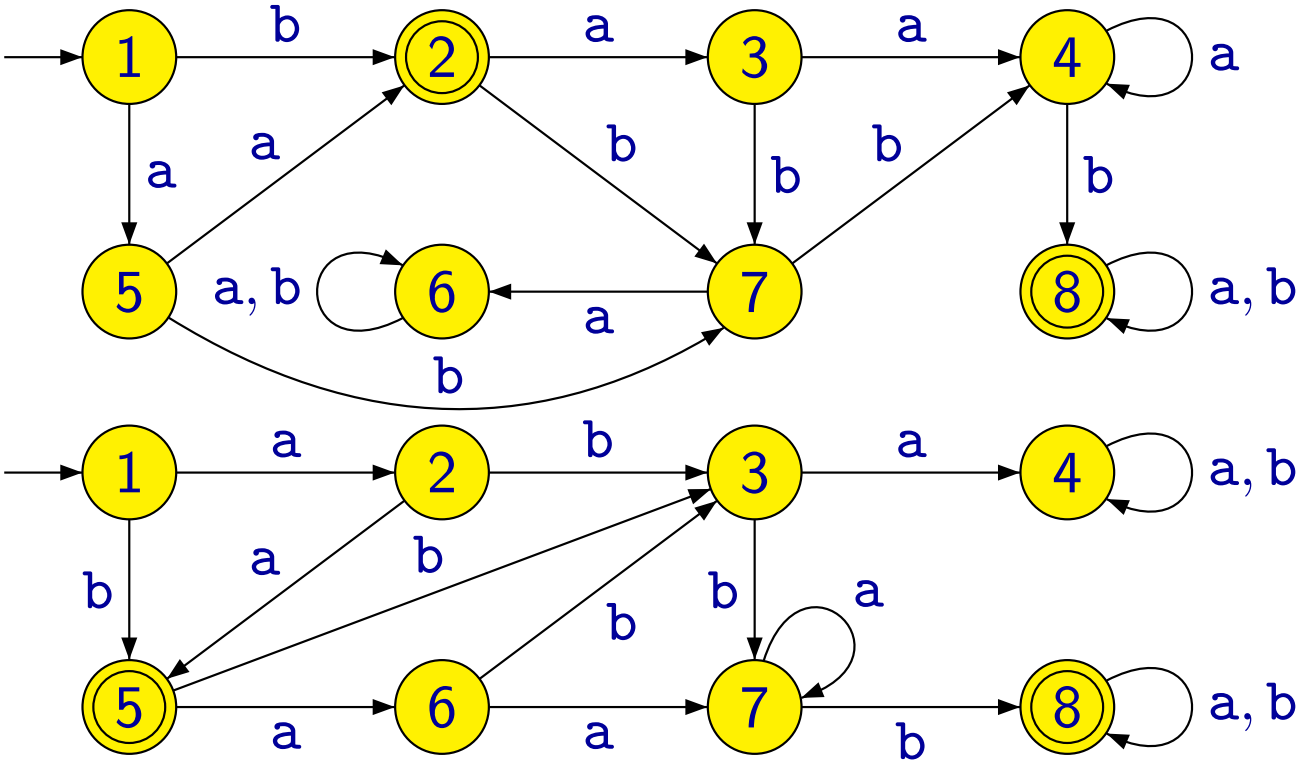
$\mathcal{A}_1$	a	b
$\rightarrow 1$	5	2
$\leftarrow 2$	3	7
3	4	7
4	4	8
5	2	7
6	6	6
7	6	4
$\leftarrow 8$	8	8

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

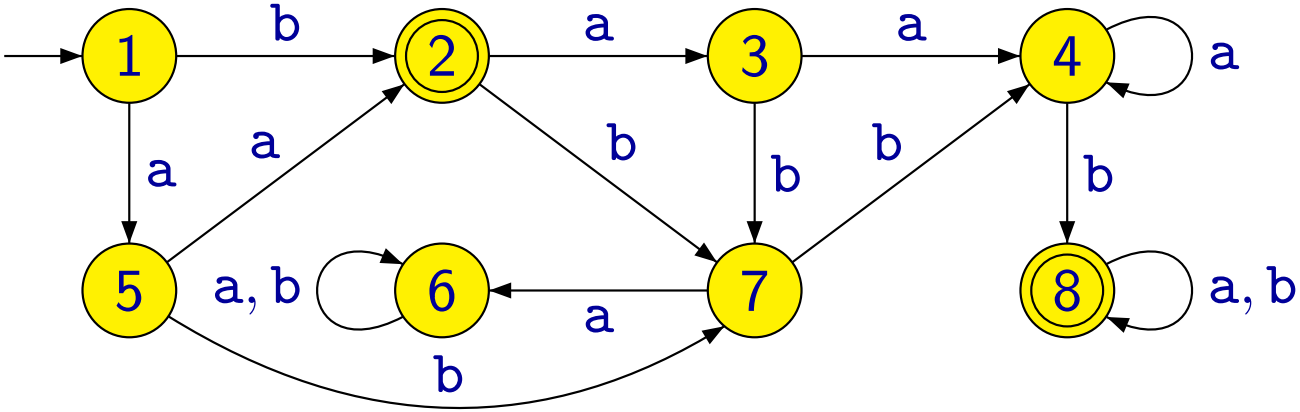
Normovaný tvar automatu



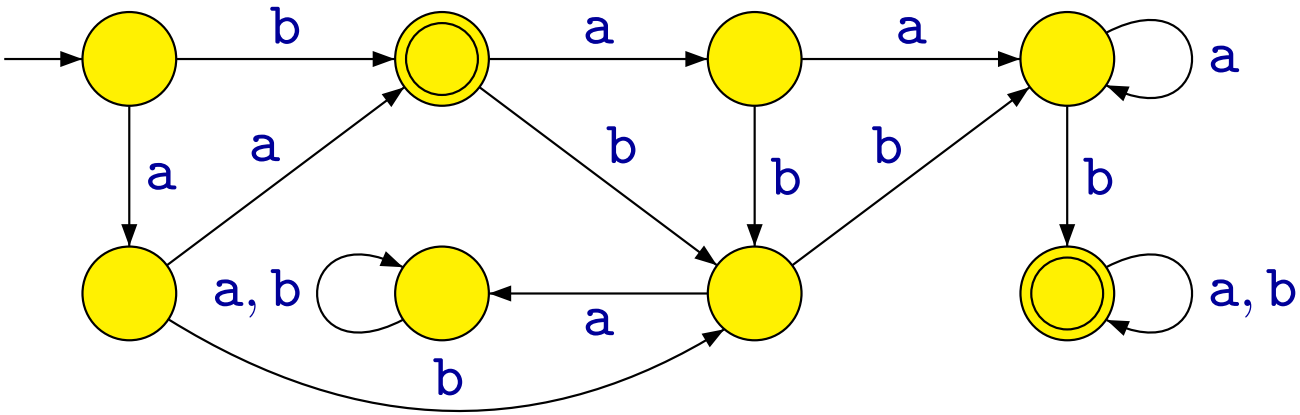
$\mathcal{A}_2$	a	b
$\rightarrow 1$	2	5
2	5	3
3	4	7
4	4	4
$\leftarrow 5$	6	3
6	7	3
7	7	8
$\leftarrow 8$	8	8

- Pokud se automaty liší jen pojmenováním stavů, pak jsou zcela jistě ekvivalentní.
- Automaty na obrázku můžeme čísly 1 až 8 pojmenovat 8! způsoby.
- Chtěli bychom jednoznačně vybrat jedno z možných pojmenování.

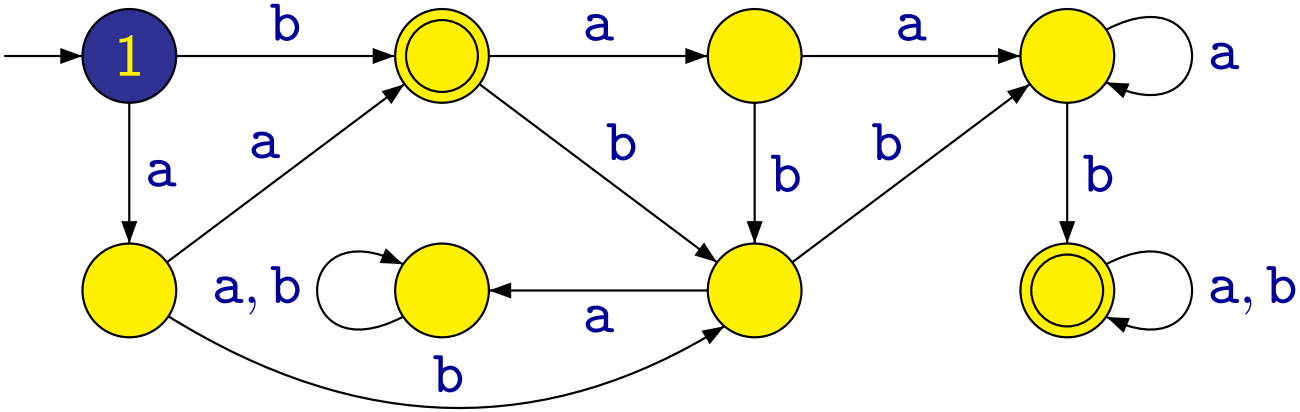
Normovaný tvar automatu



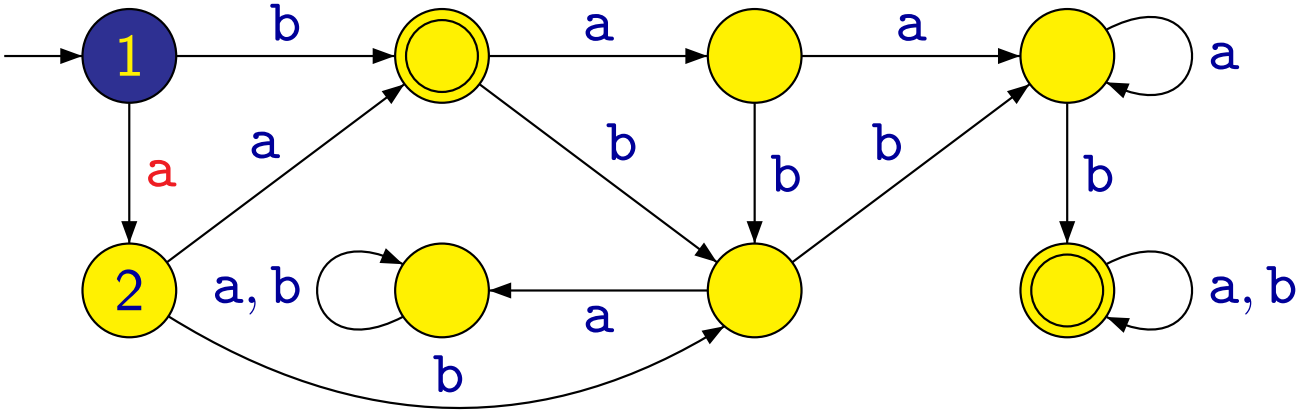
Normovaný tvar automatu



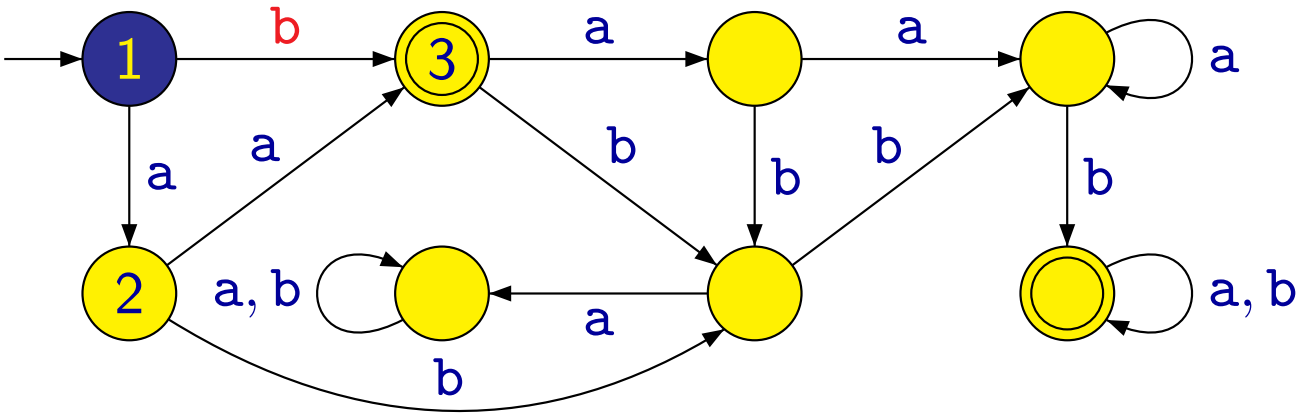
Normovaný tvar automatu



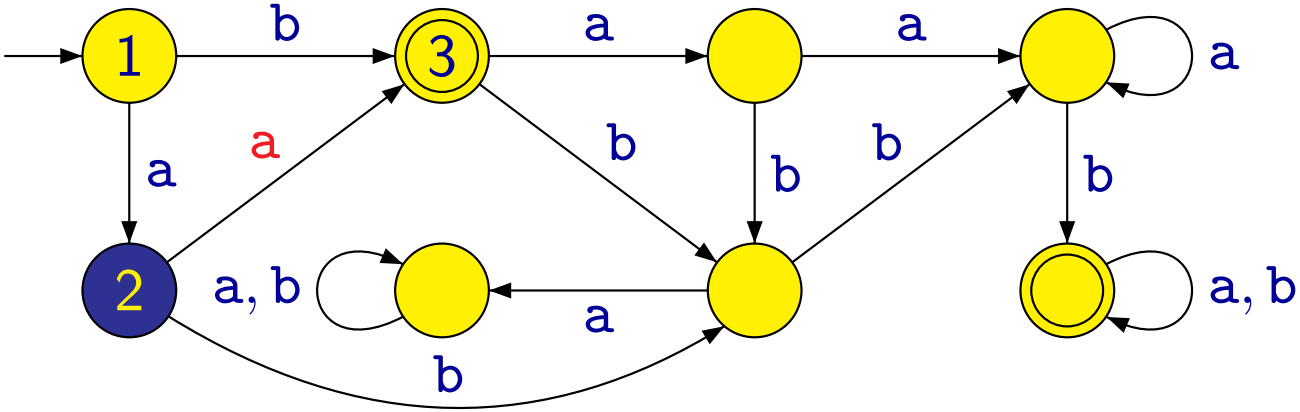
Normovaný tvar automatu



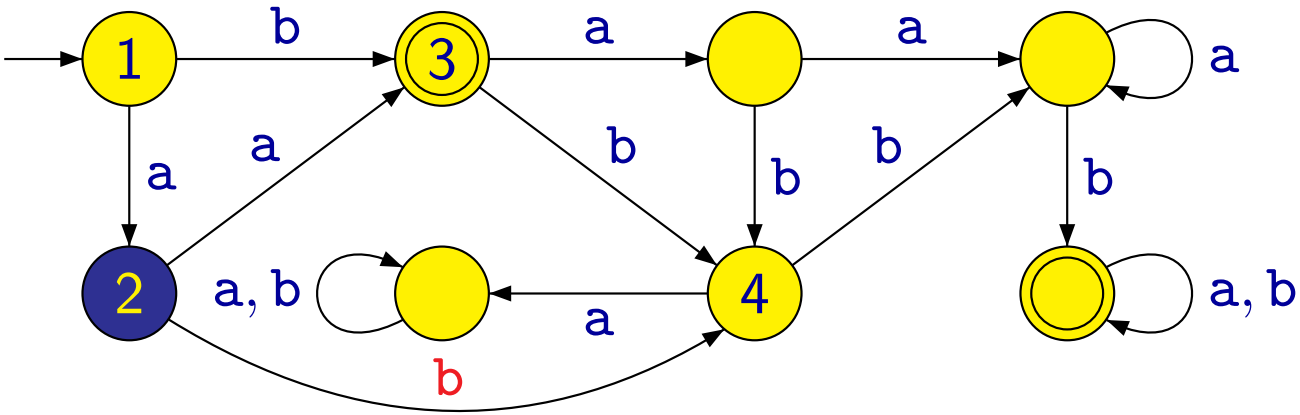
Normovaný tvar automatu



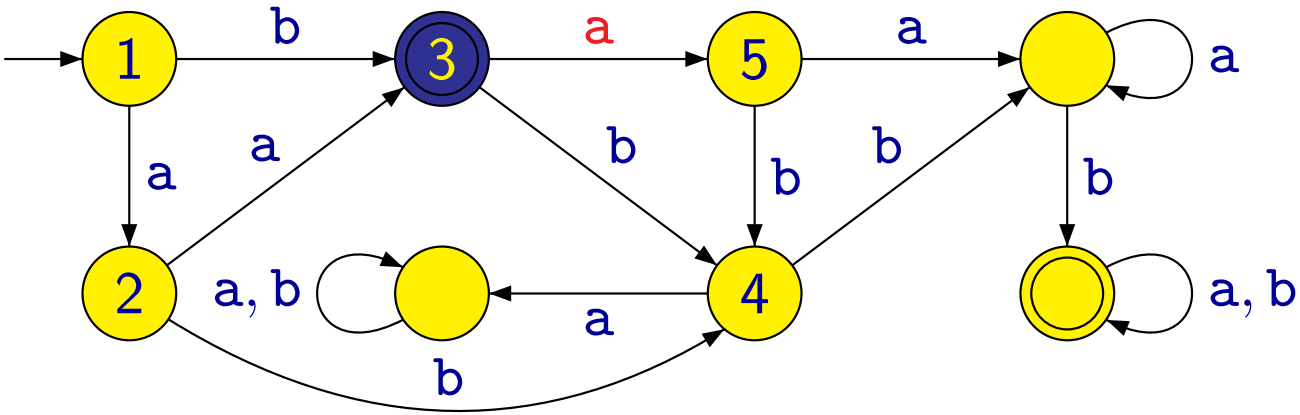
Normovaný tvar automatu



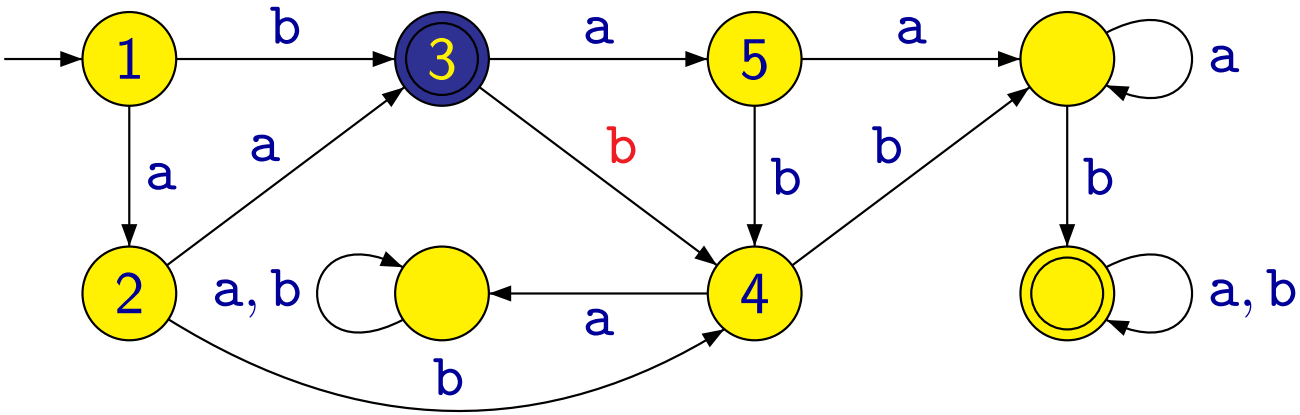
Normovaný tvar automatu



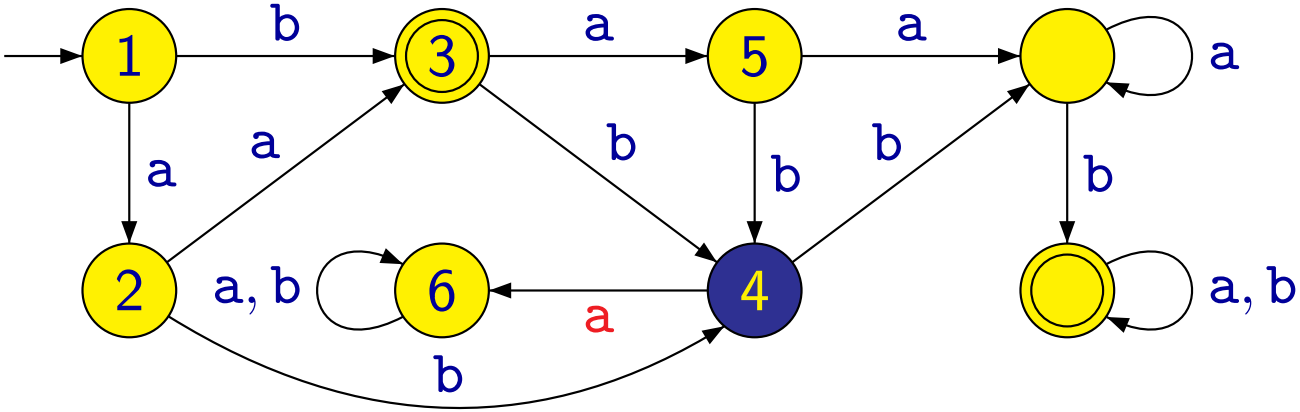
Normovaný tvar automatu



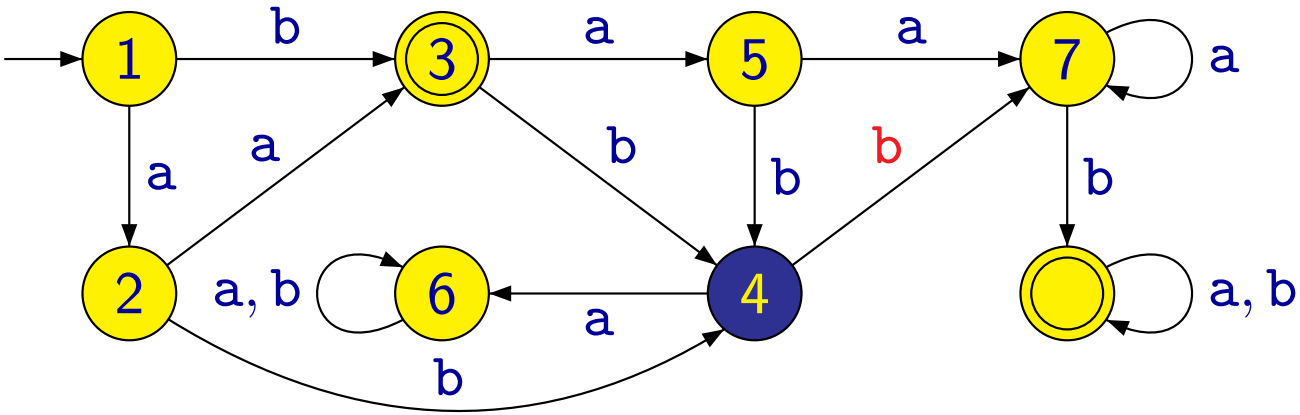
Normovaný tvar automatu



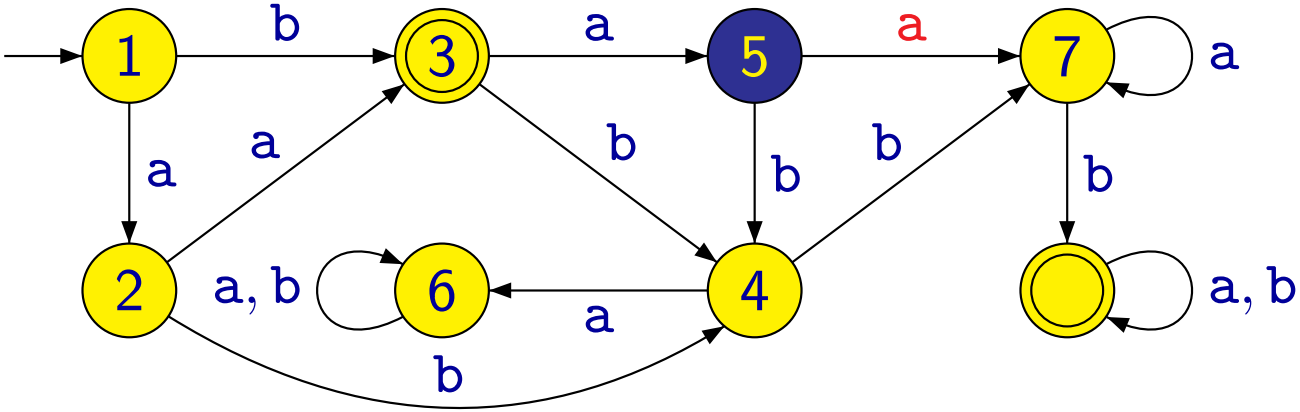
Normovaný tvar automatu



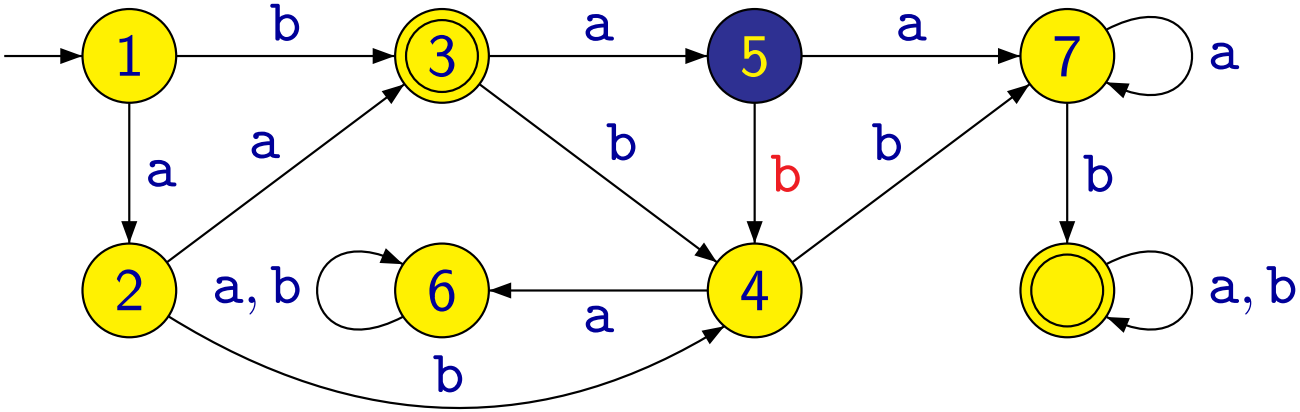
Normovaný tvar automatu



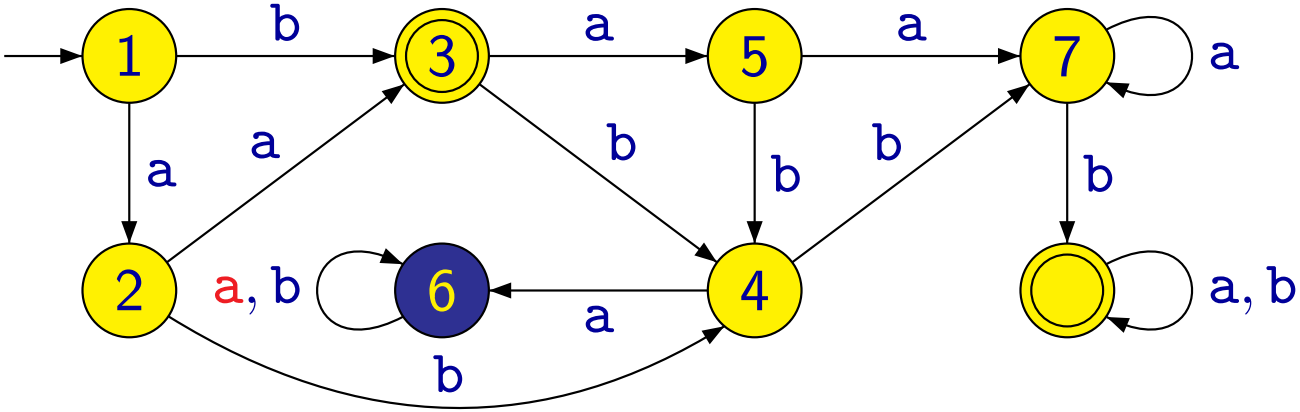
Normovaný tvar automatu



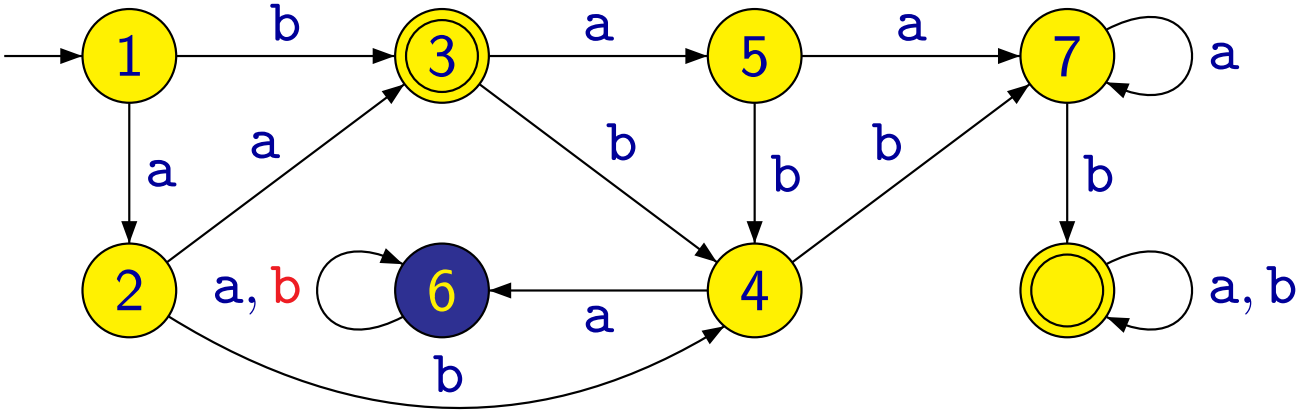
Normovaný tvar automatu



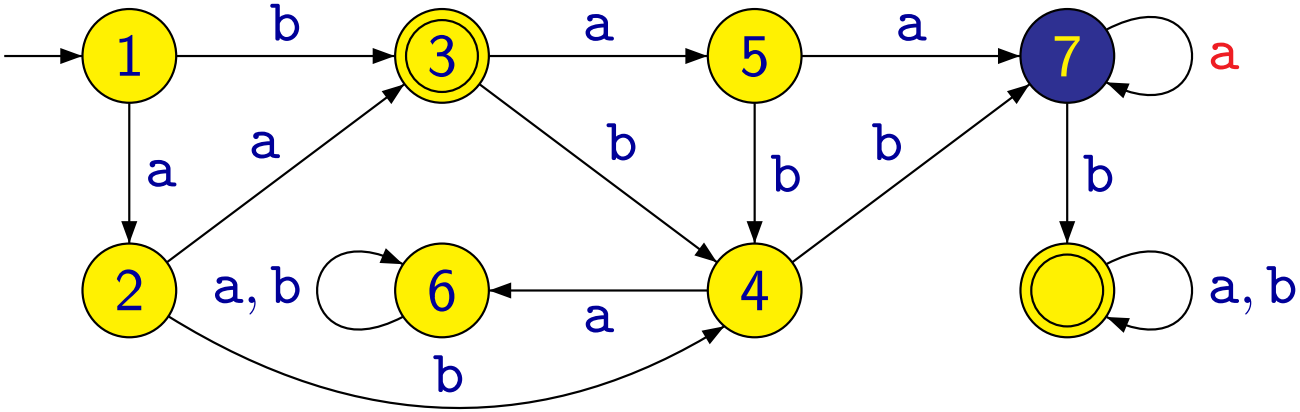
Normovaný tvar automatu



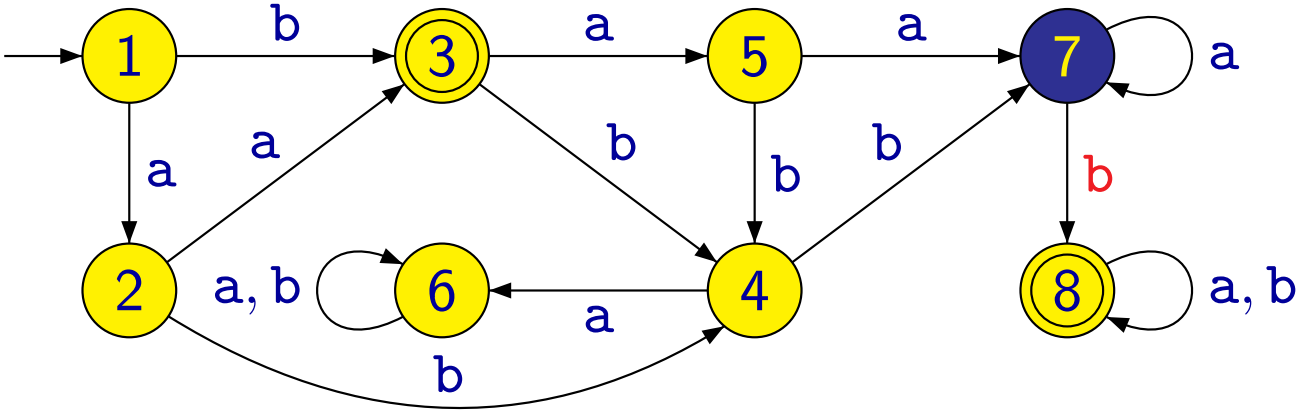
Normovaný tvar automatu



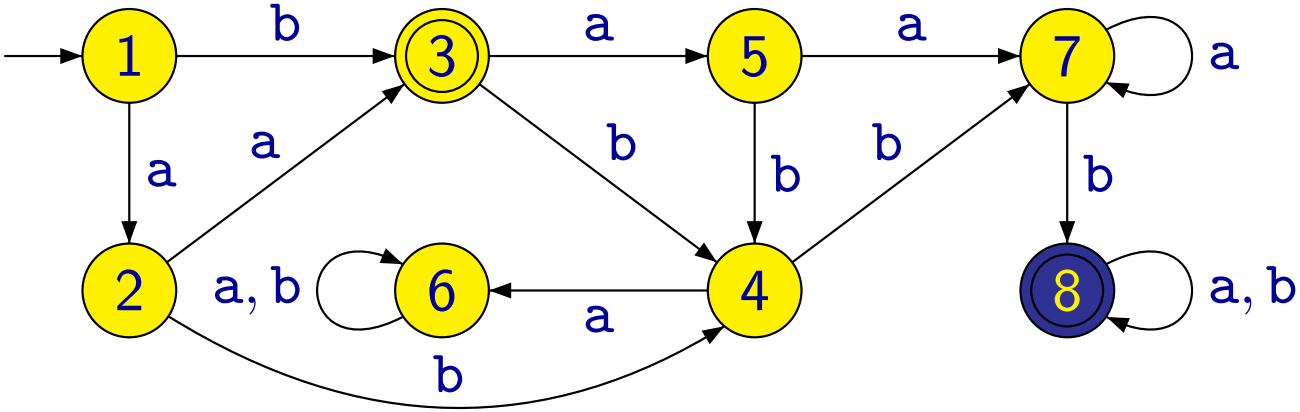
Normovaný tvar automatu



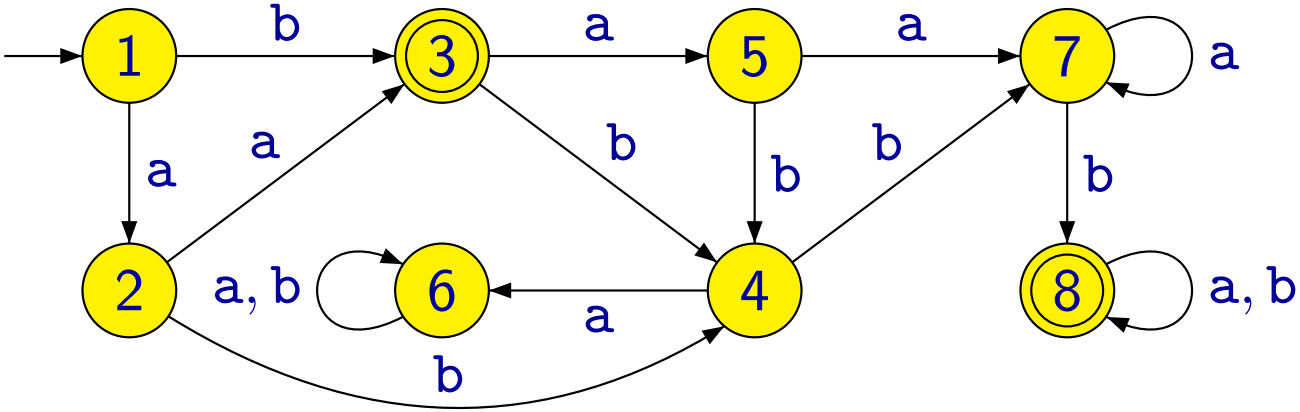
Normovaný tvar automatu



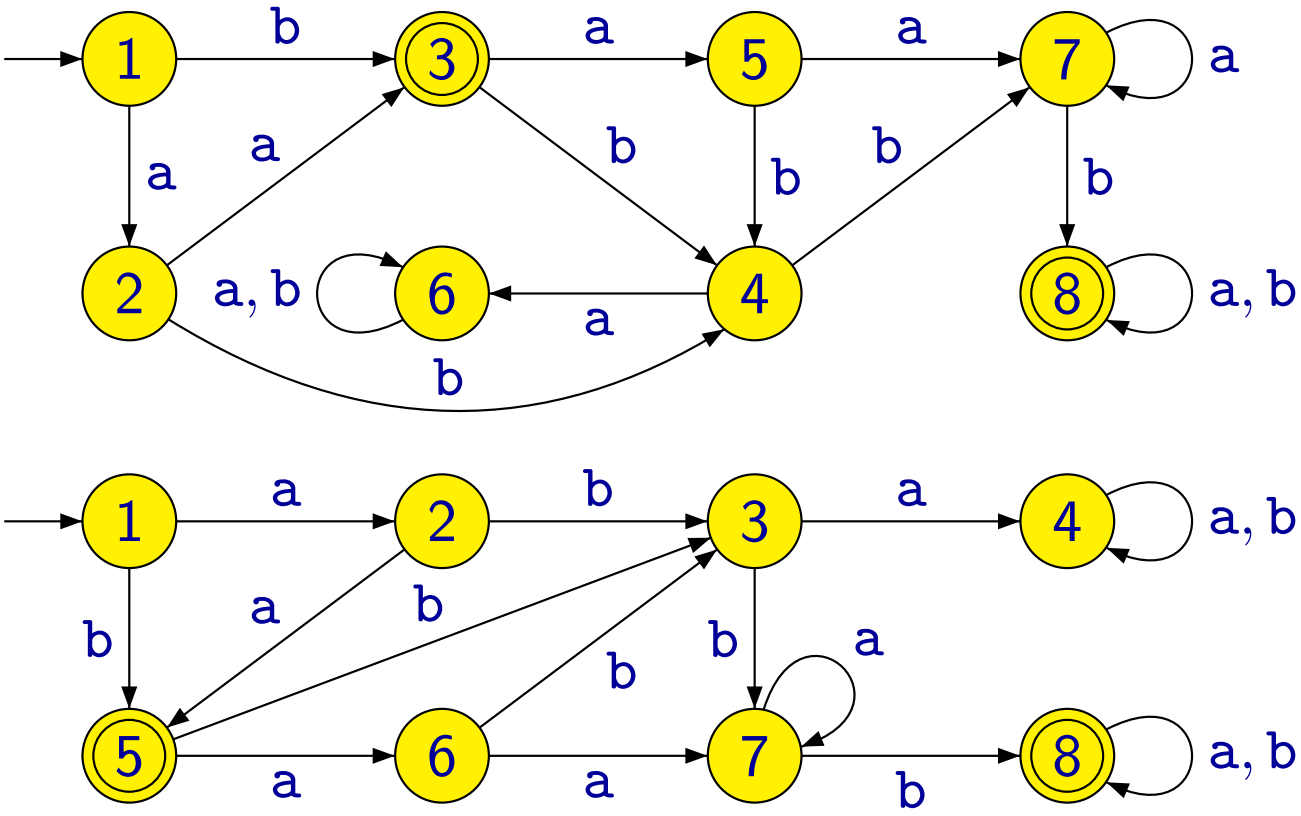
Normovaný tvar automatu



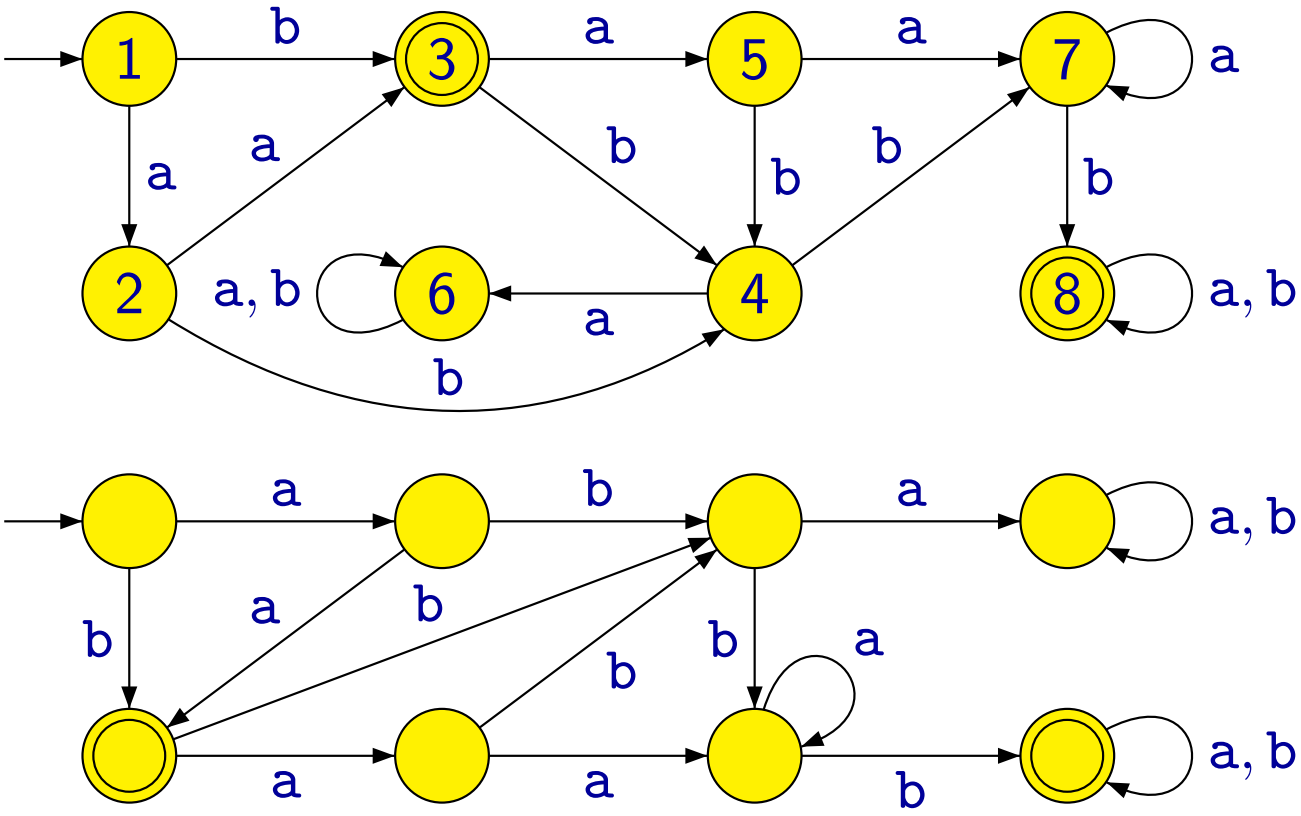
Normovaný tvar automatu



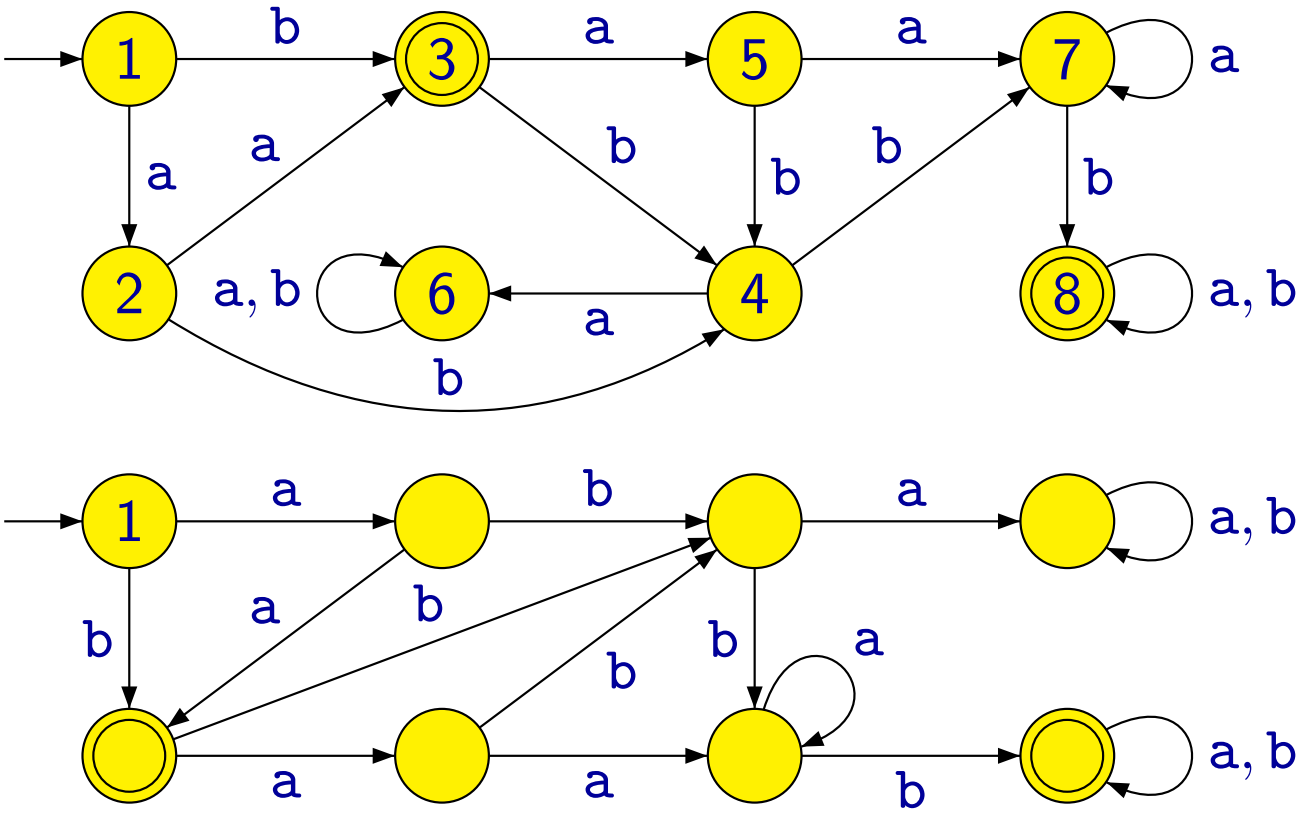
Normovaný tvar automatu



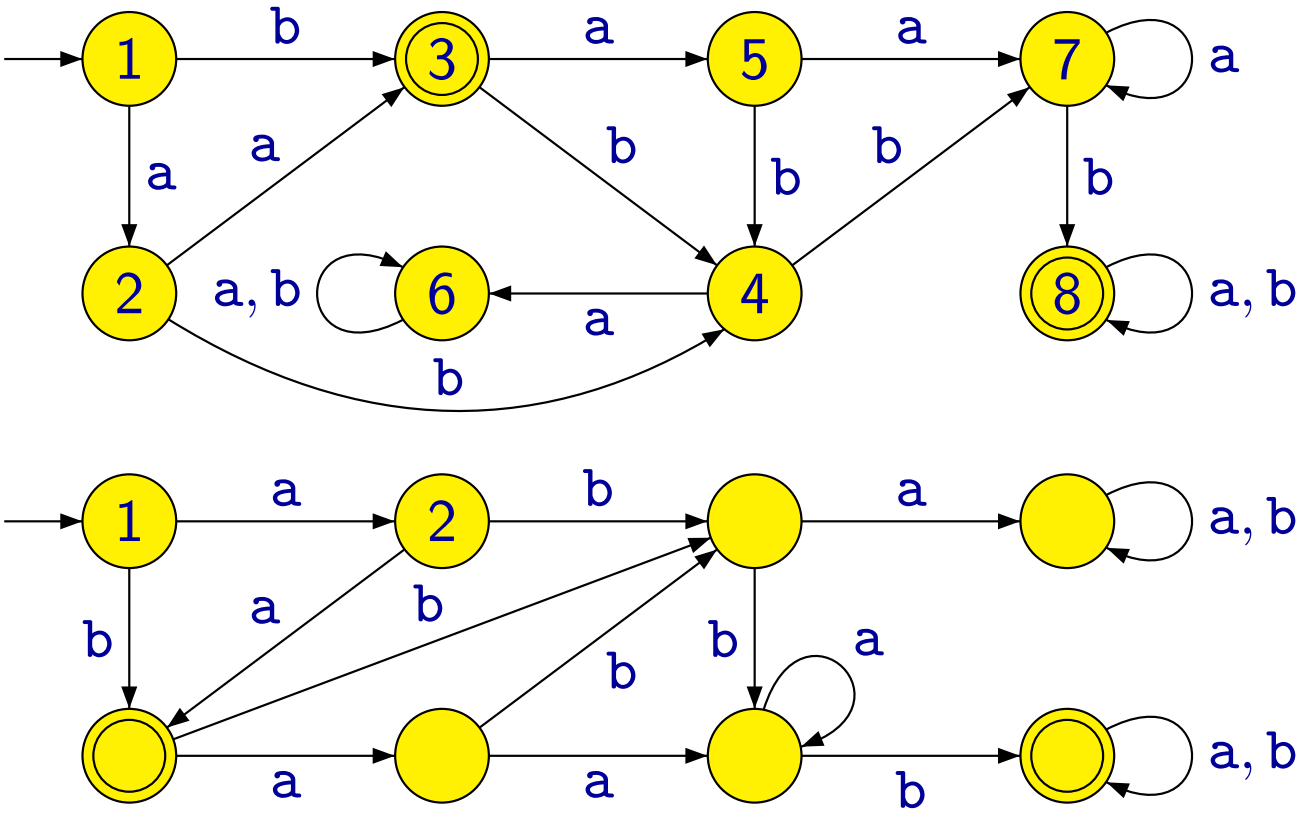
Normovaný tvar automatu



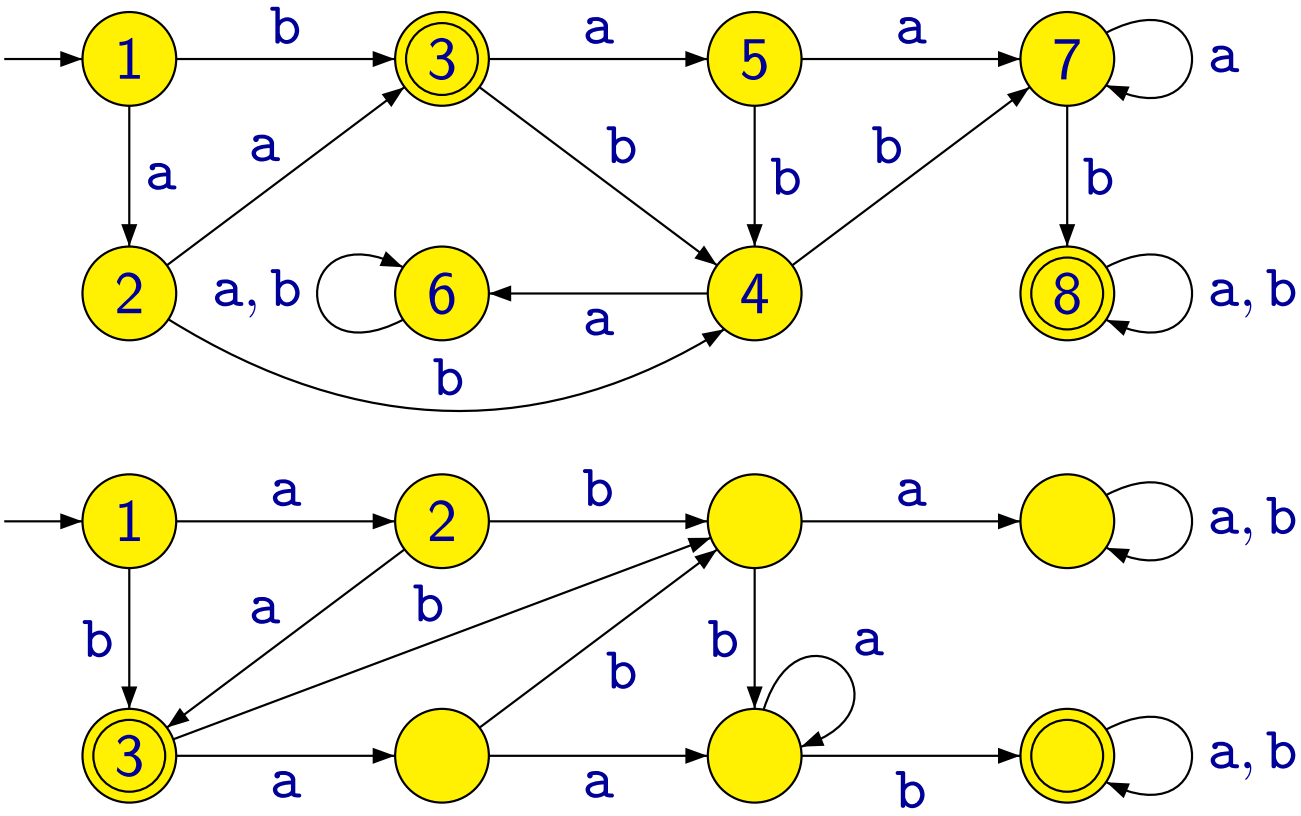
Normovaný tvar automatu



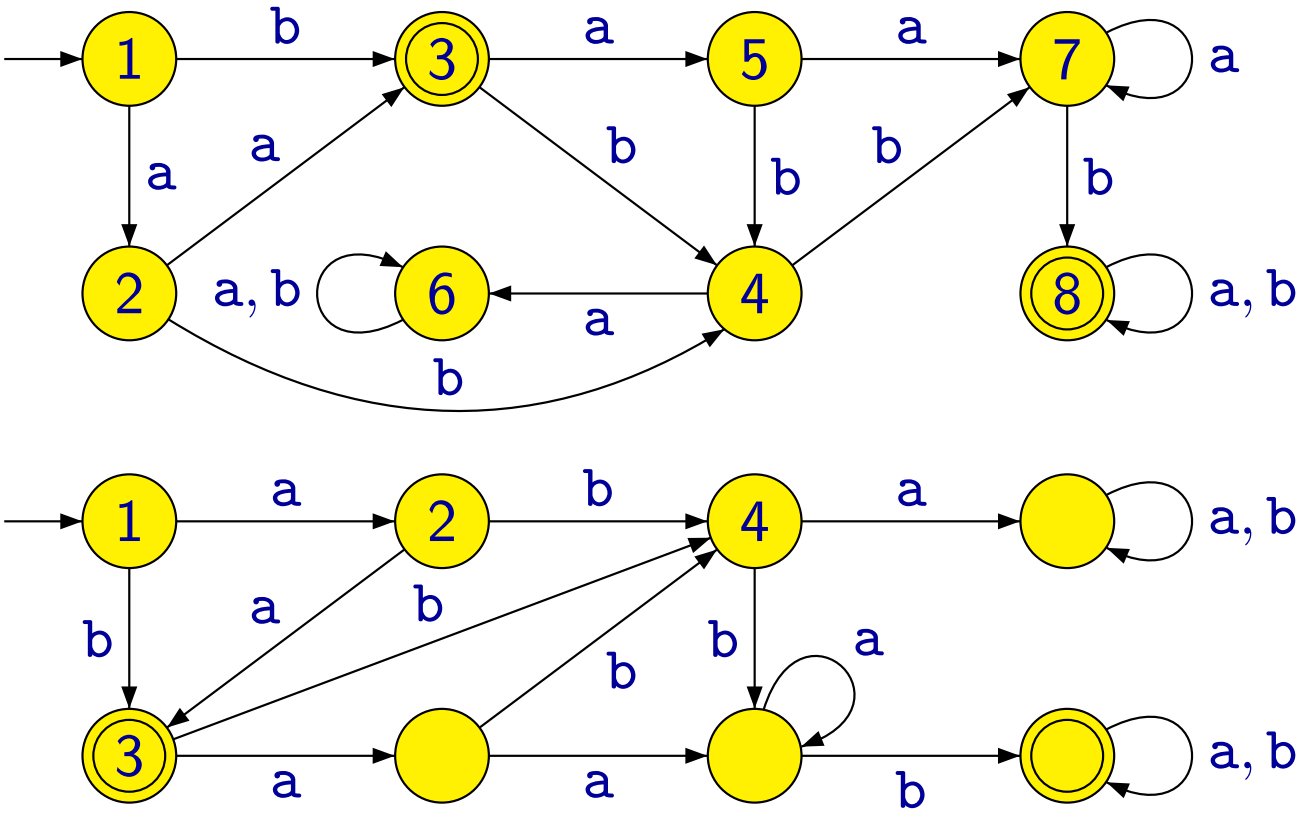
Normovaný tvar automatu



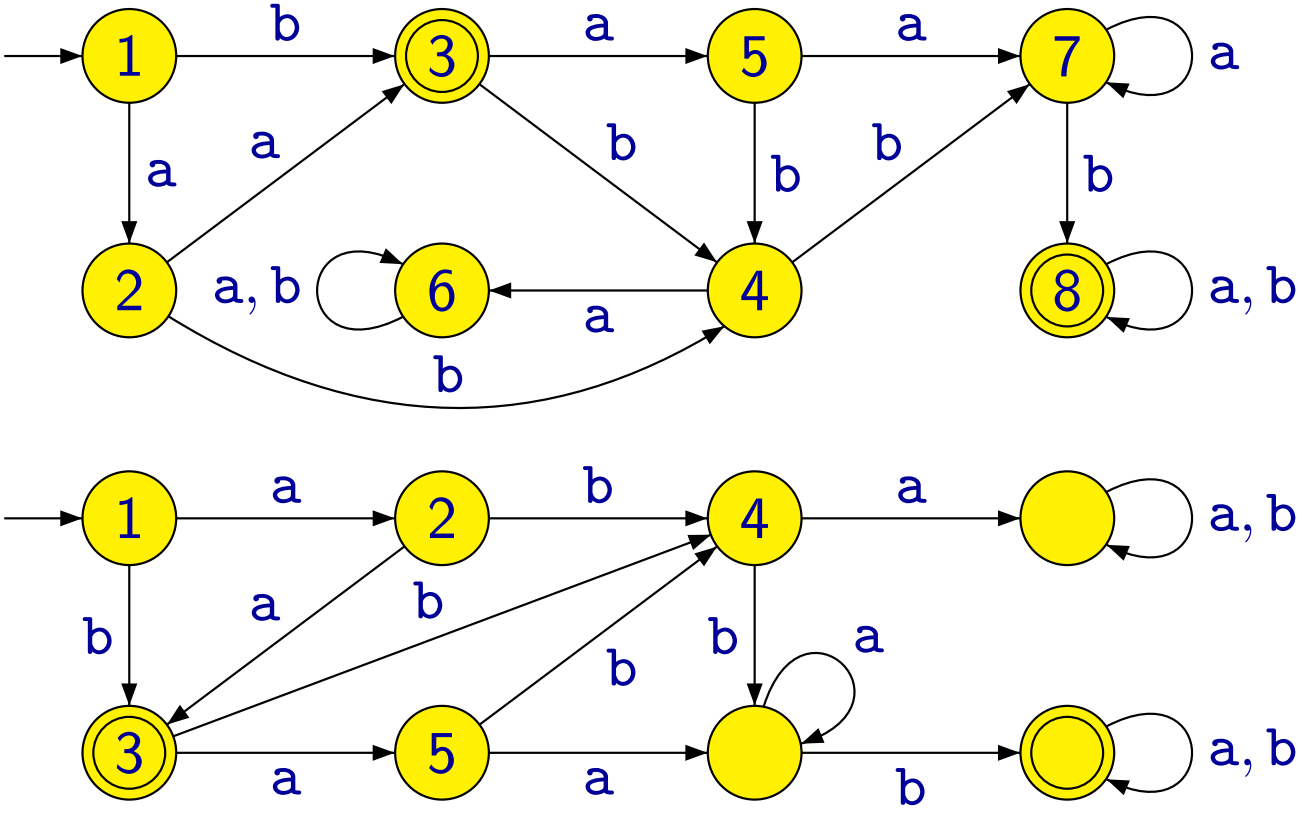
Normovaný tvar automatu



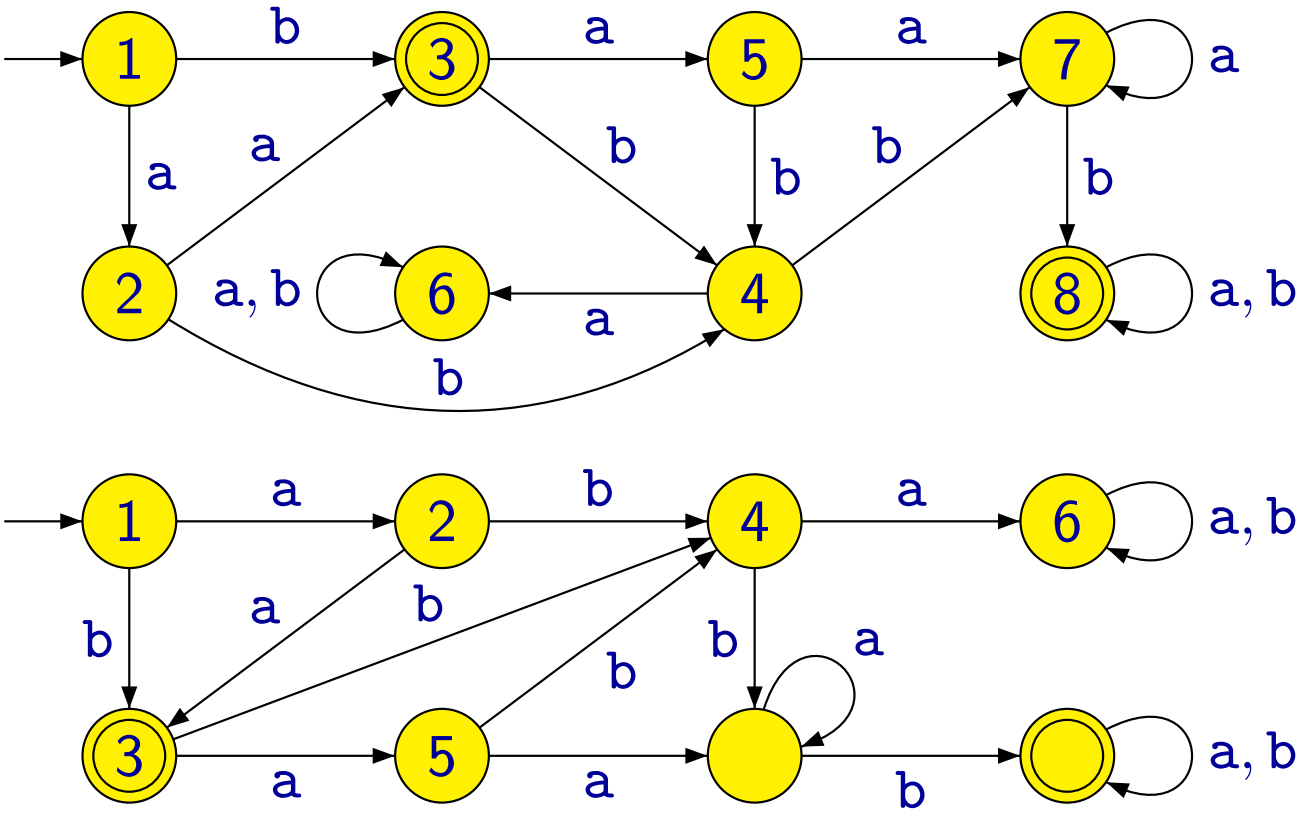
Normovaný tvar automatu



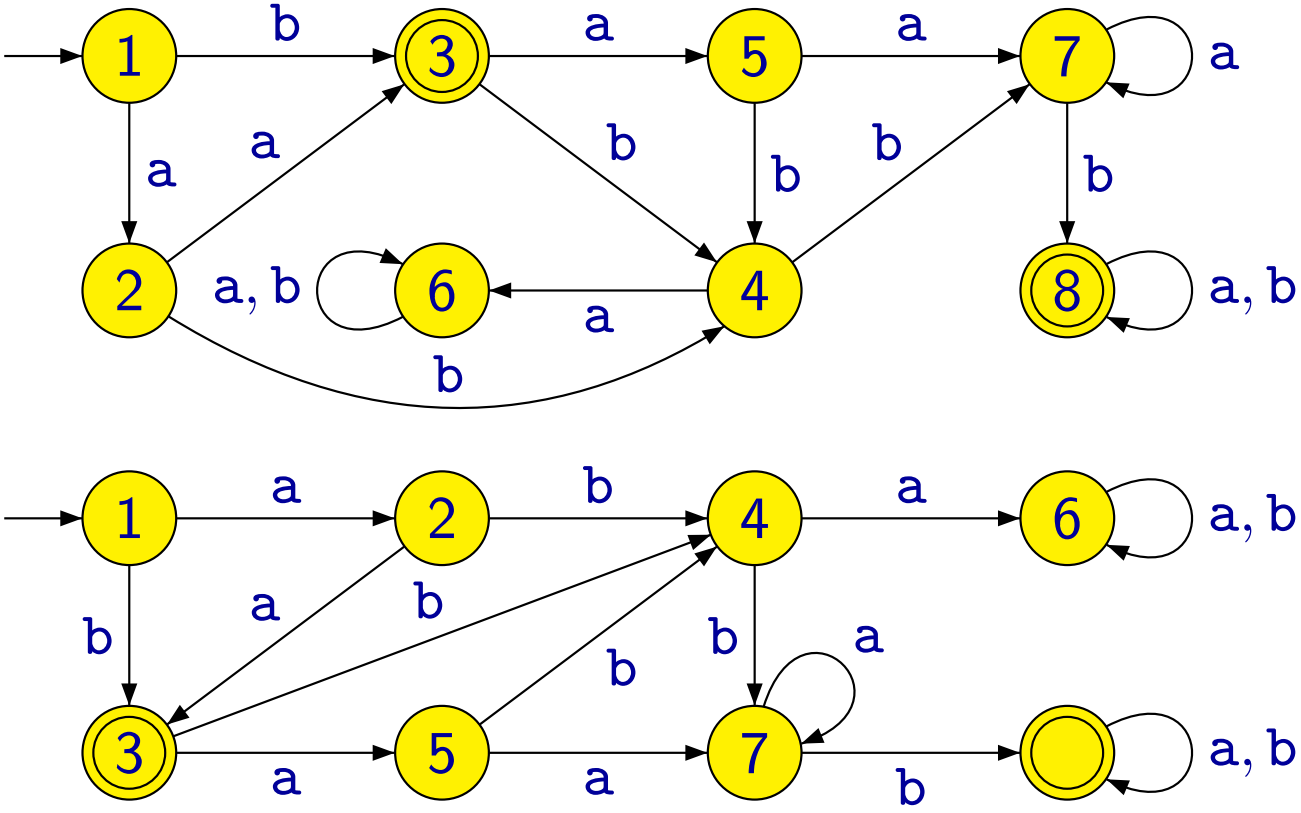
Normovaný tvar automatu



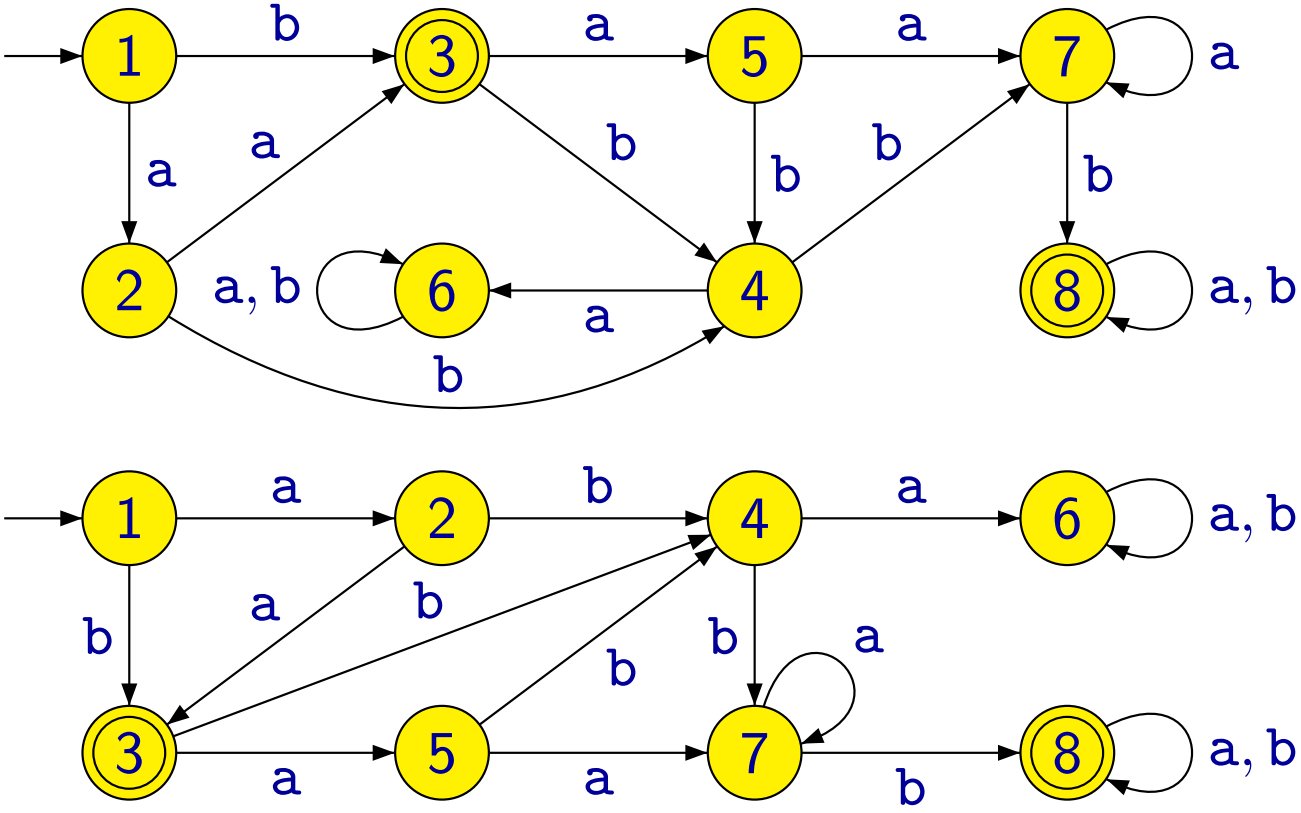
Normovaný tvar automatu



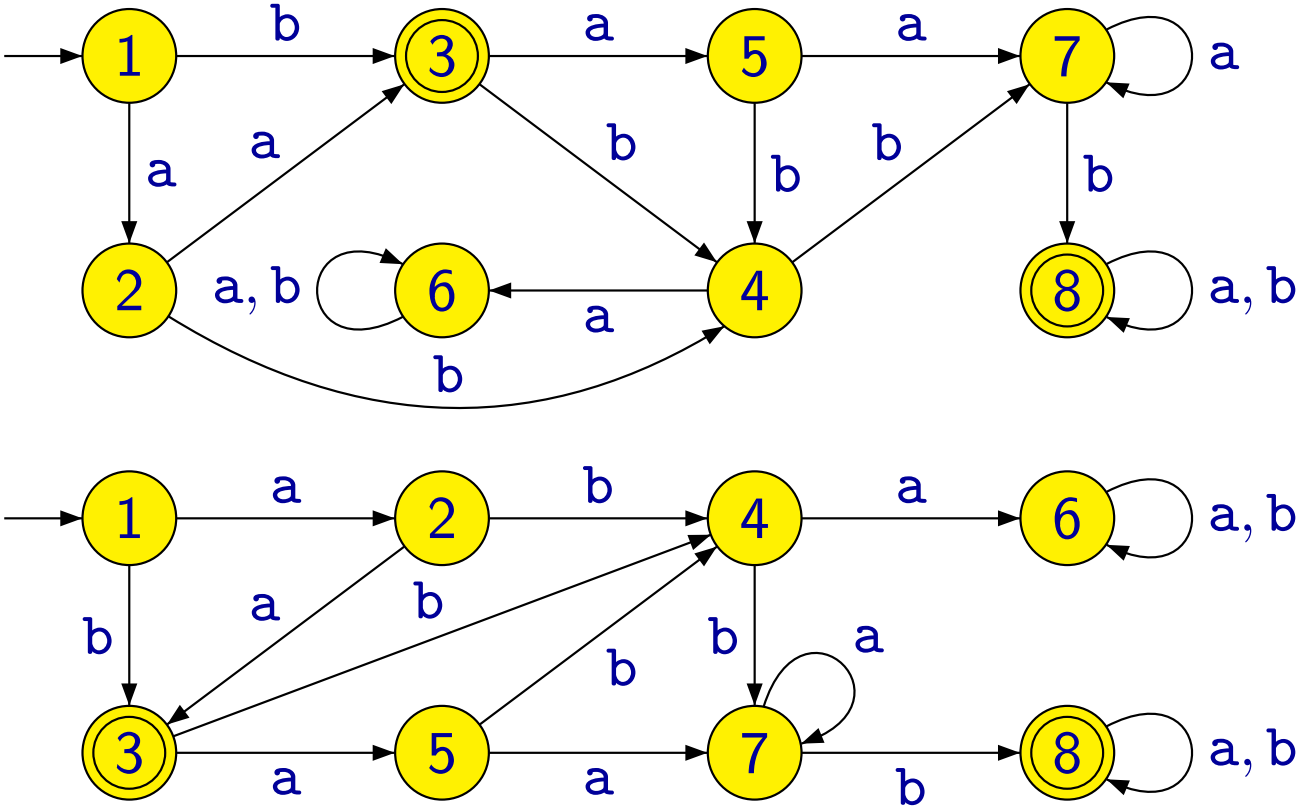
Normovaný tvar automatu



Normovaný tvar automatu

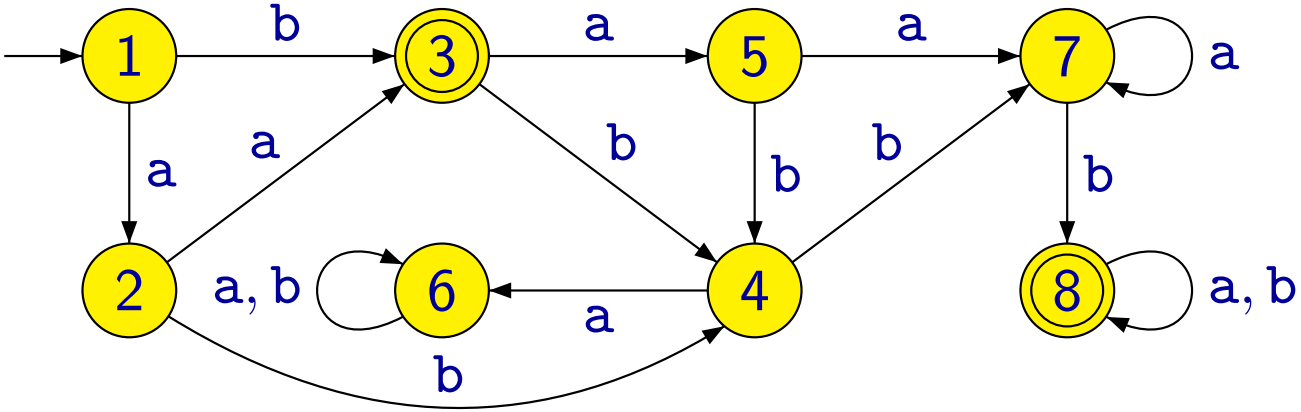


Normovaný tvar automatu

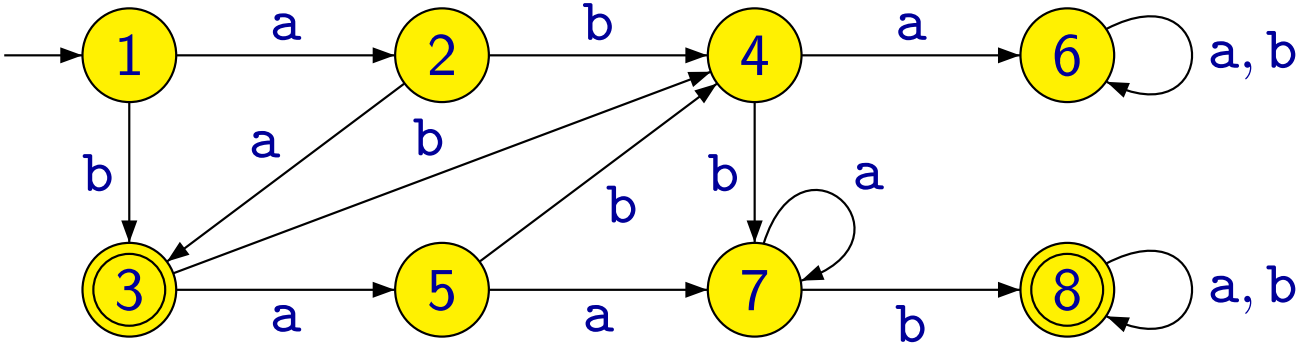


	a	b
$\rightarrow$ 1	2	3
2	3	4
$\leftarrow$ 3	5	4
4	6	7
5	7	4
6	6	6
7	7	8
$\leftarrow$ 8	8	8

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 počáteční a přijímající stavy.
- Automaty jsou tedy nejen ekvivalentní, ale dokonce stejné.
- Říkáme, že automaty jsou v normovaném tvaru.

Normovaný tvar automatu

Mějme deterministický konečný automat $A = (Q, \Sigma, \delta, q_0, F)$ bez nedosažitelných stavů.

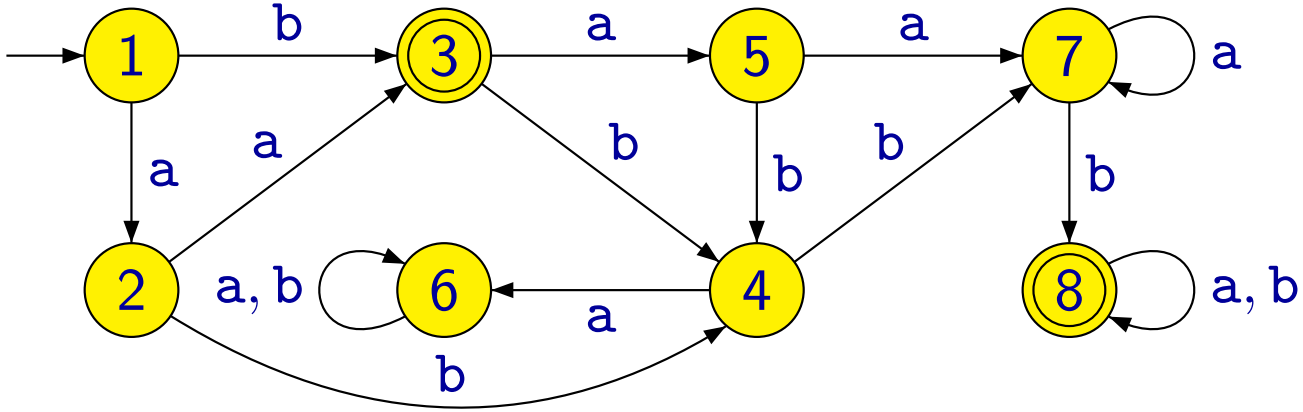
Předpokládejme, že máme určité uspořádání prvků abecedy Σ, kterým je určeno uspořádání $<_L$ na Σ^*.

Označme pro každý stav $q \in Q$ symbolem u_q nejmenší slovo podle uspořádání $<_L$ takové, že $q_0 \xrightarrow{u_q} q$.

DKA A je v **normovaném tvaru**, pokud:

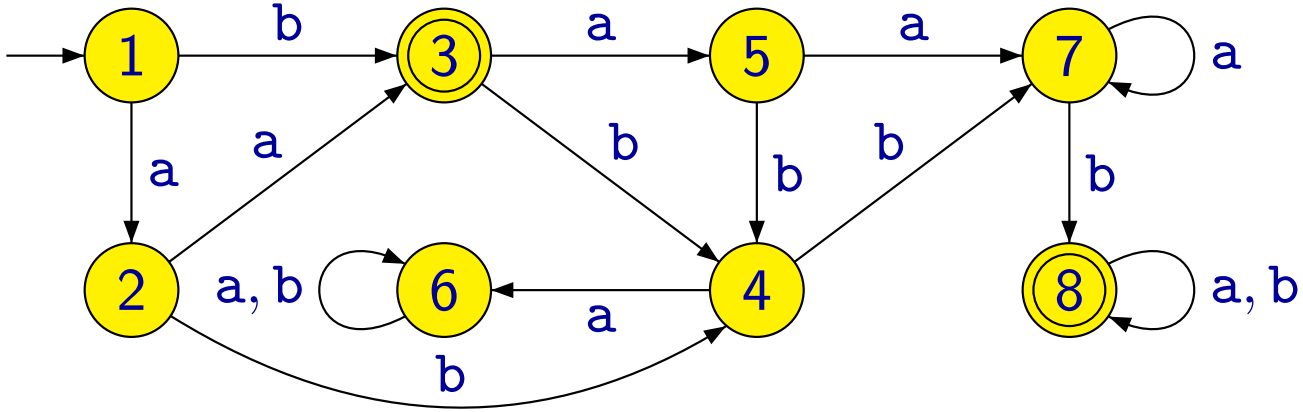
- $Q = \{1, 2, 3, \ldots, n\}$ pro nějaké $n \geq 1$.
- 1 je počáteční stav.
- Pro každou dvojici stavů $i, j \in \{1, 2, \ldots, n\}$ takovou, že $i < j$, platí $u_i <_L u_j$.

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:

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

Automaty a operace na jazycích

Pro jednoduché jazyky můžeme být schopni navrhnout automat přímo. Pro složitější jazyky už to může být obtížnější.

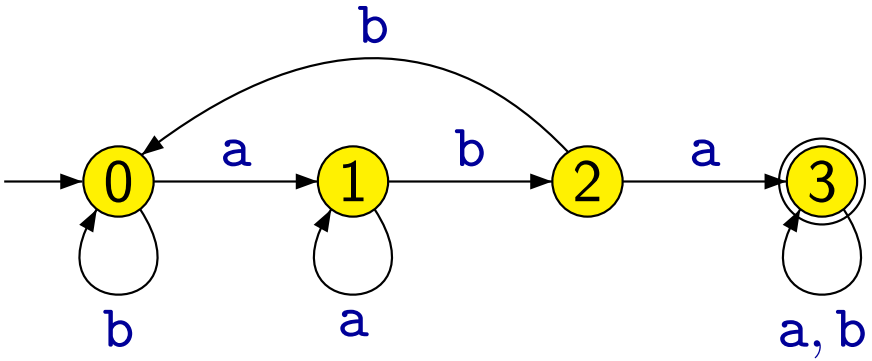
Často můžeme být schopni složitější jazyk popsat jako výsledek nějaké operace (sjednocení, průnik, doplněk, zřetězení, iterace, zrcadlový obraz apod.) aplikované na nějaké jednodušší jazyky.

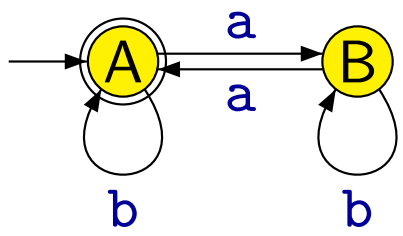
V takém případě může být schůdnější následující „modulární“ postup:

- Nejprve sestrojit automaty pro ony jednodušší jazyky.
- Z vytvořených automatů určitou algoritmickou konstrukcí vytvořit automat rozpoznávající jazyk, který je výsledkem příslušné operace aplikované na jazyky rozpoznávané původními automaty.

Automat pro průnik jazyků

Máme následující dva automaty:

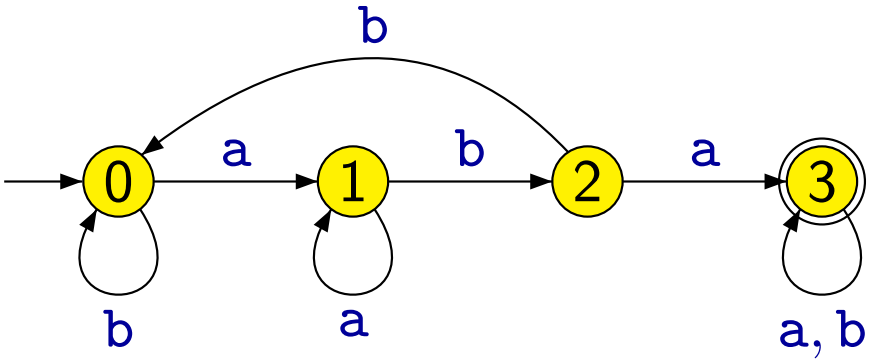


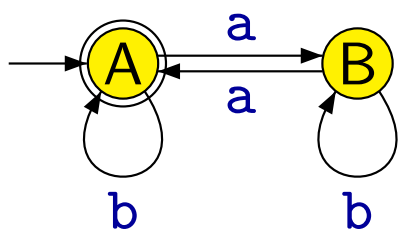


Příjmou oba slovo ababb?

Automat pro průnik jazyků

Máme následující dva automaty:



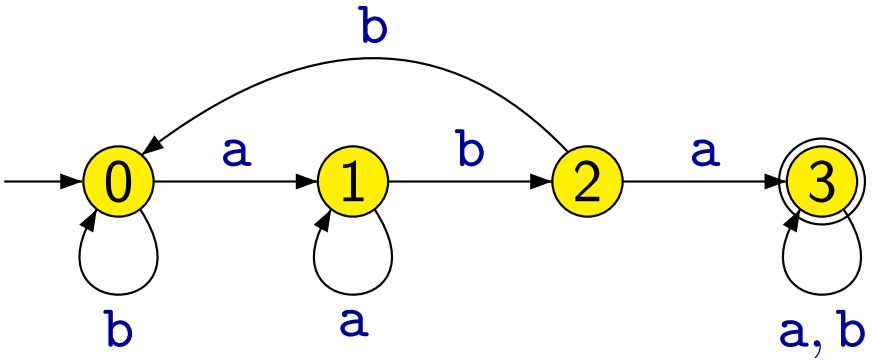


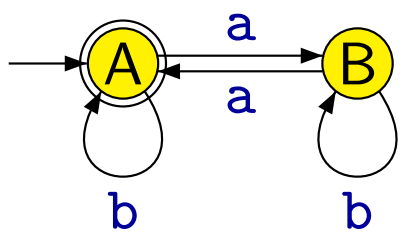
Příjmou oba slovo ababb?

Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

Automat pro průnik jazyků

Máme následující dva automaty:





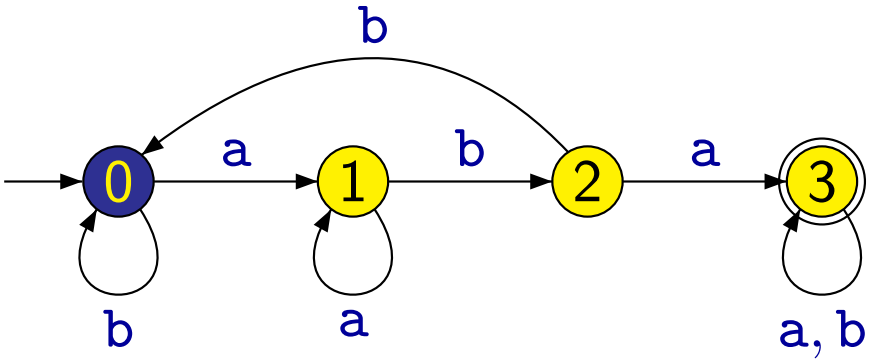
Příjmou oba slovo ababb?

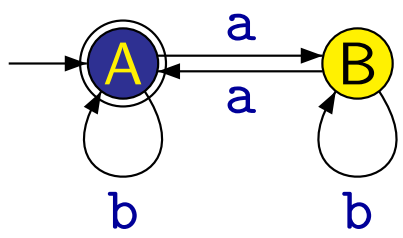
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

Místo toho můžeme nechat oba automaty číst dané slovo současně.

Automat pro průnik jazyků

Máme následující dva automaty:





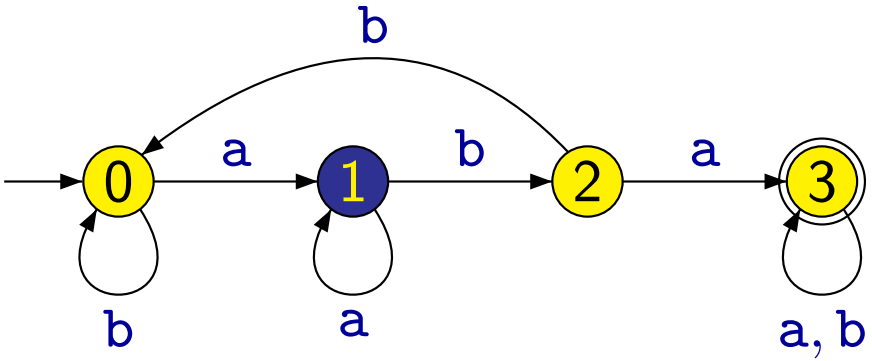
Příjmou oba slovo ababb?

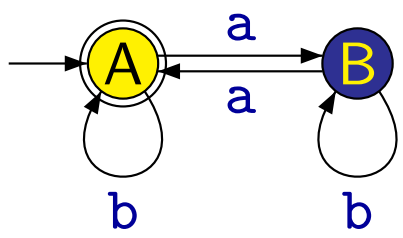
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

Místo toho můžeme nechat oba automaty číst dané slovo současně.

Automat pro průnik jazyků

Máme následující dva automaty:





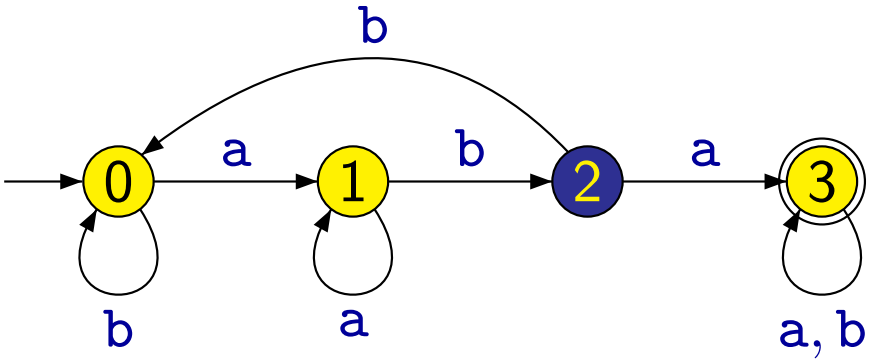
Příjmou oba slovo ababb?

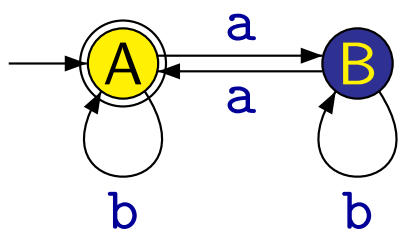
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

Místo toho můžeme nechat oba automaty číst dané slovo současně.

Automat pro průnik jazyků

Máme následující dva automaty:





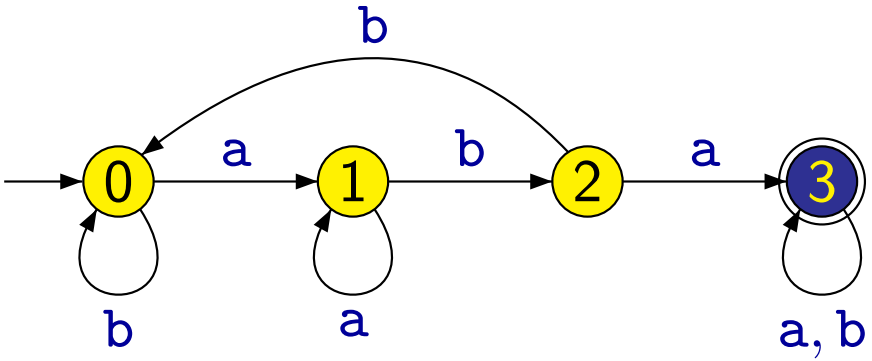
Příjmou oba slovo ababb?

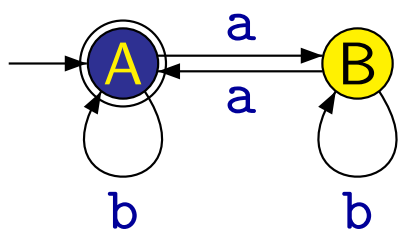
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

Místo toho můžeme nechat oba automaty číst dané slovo současně.

Automat pro průnik jazyků

Máme následující dva automaty:





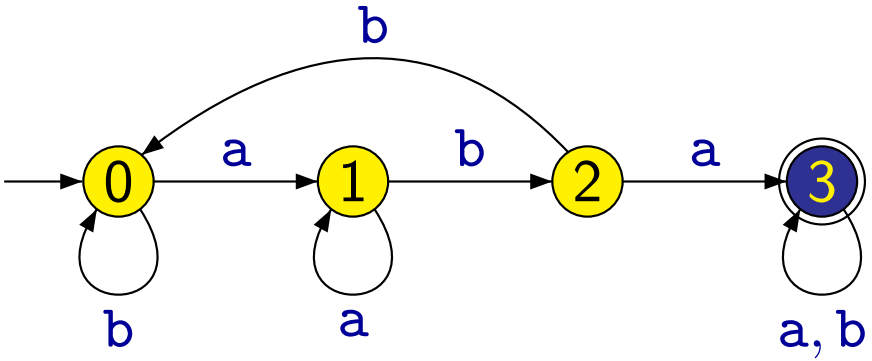
Příjmou oba slovo ababb?

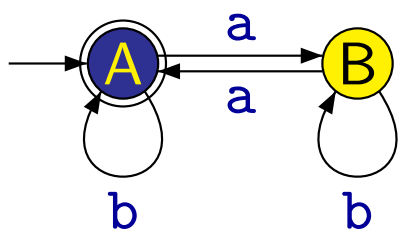
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

Místo toho můžeme nechat oba automaty číst dané slovo současně.

Automat pro průnik jazyků

Máme následující dva automaty:





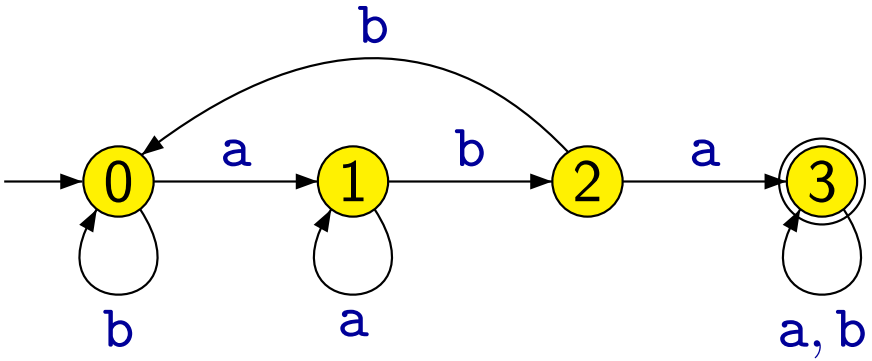
Příjmou oba slovo ababb?

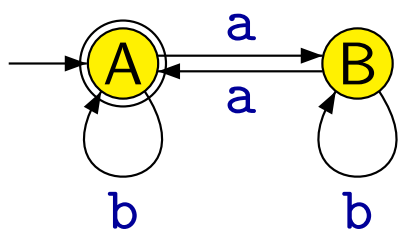
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

Místo toho můžeme nechat oba automaty číst dané slovo současně.

Automat pro průnik jazyků

Máme následující dva automaty:





Příjmou oba slovo ababb?

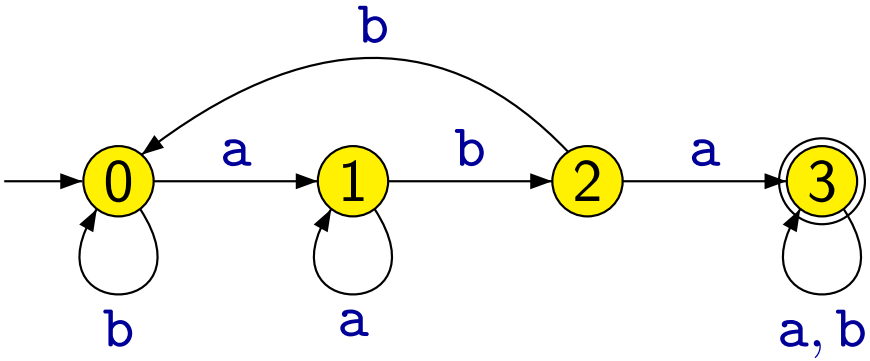
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ANO, pokud oba odpoví ANO.

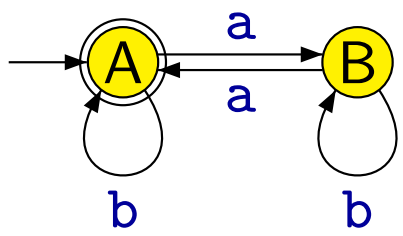
Místo toho můžeme nechat oba automaty číst dané slovo současně.

Při tomto čtení si stačí v každém kroku pamatovat dvojici stavů, ve kterých se oba automaty nacházejí.

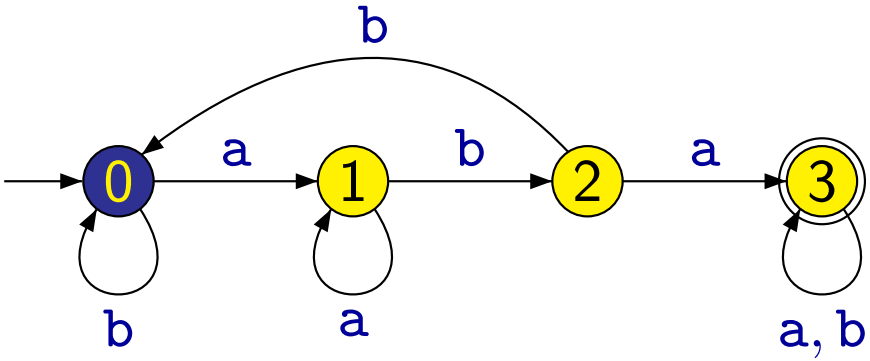
Toto může být realizováno konečným automatem, který simuluje činnost obou automatů současně! (Dvojic stavů je konečně mnoho.)

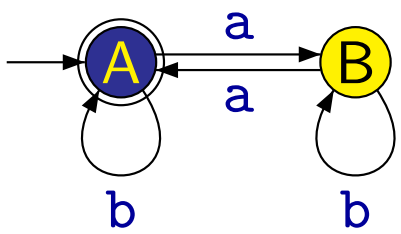
Automat pro průnik jazyků

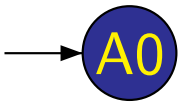




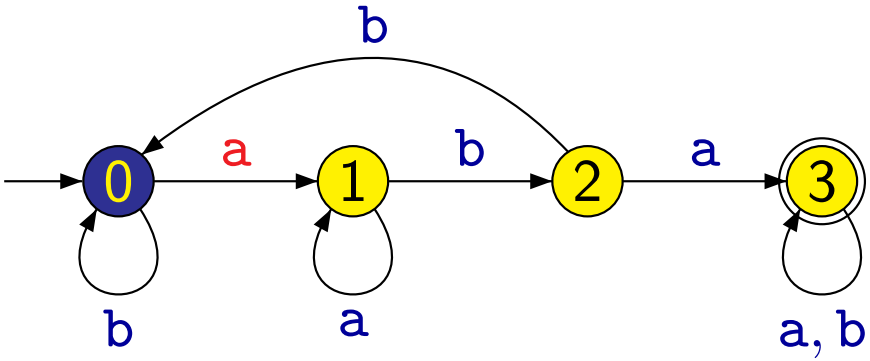
Automat pro průnik jazyků

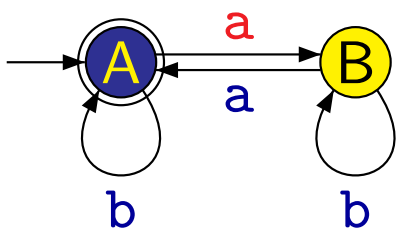




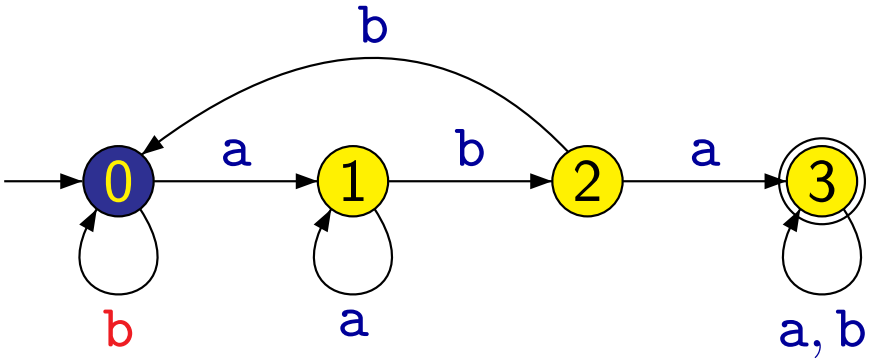


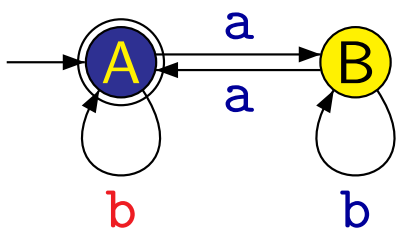
Automat pro průnik jazyků

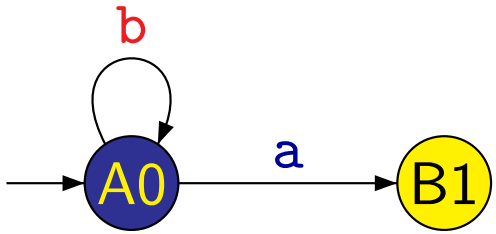




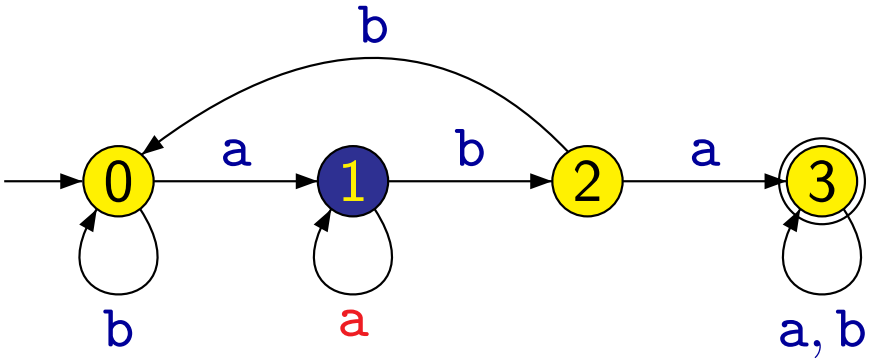
Automat pro průnik jazyků

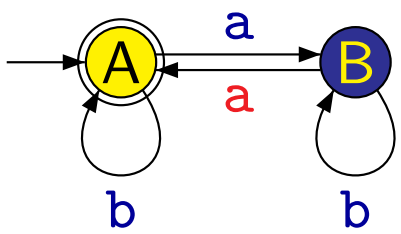


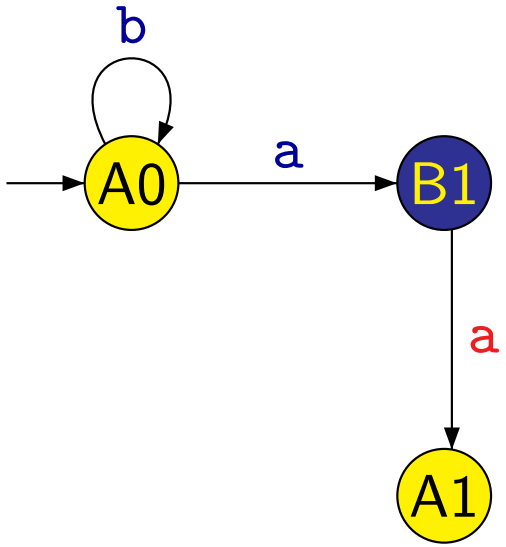




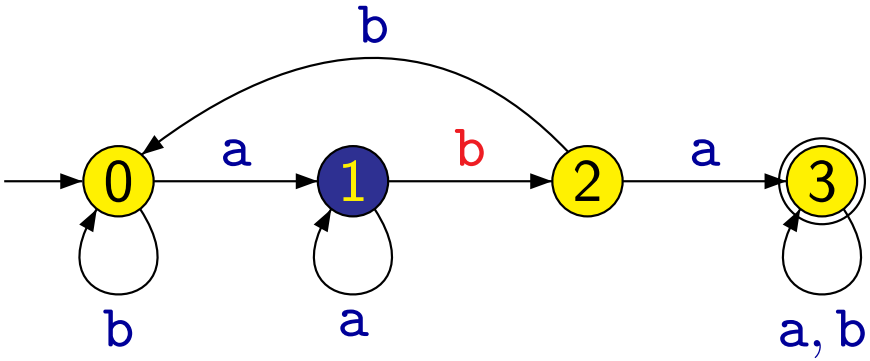
Automat pro průnik jazyků

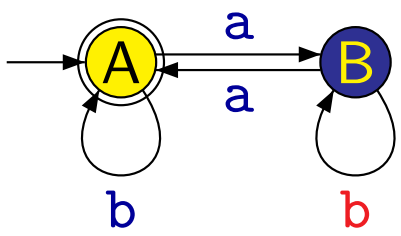


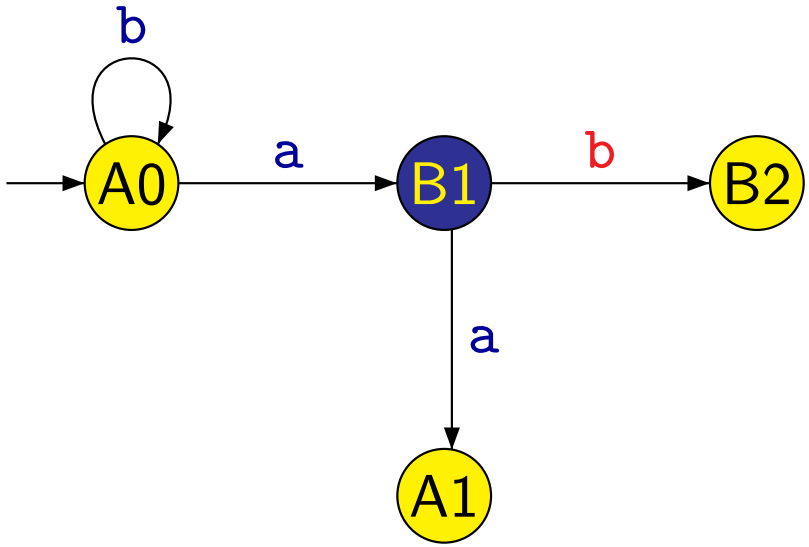




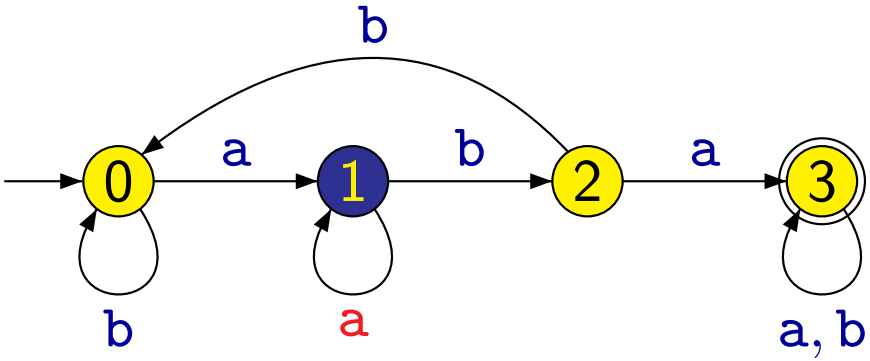
Automat pro průnik jazyků

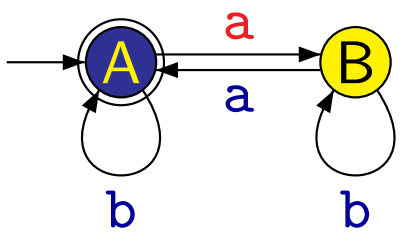


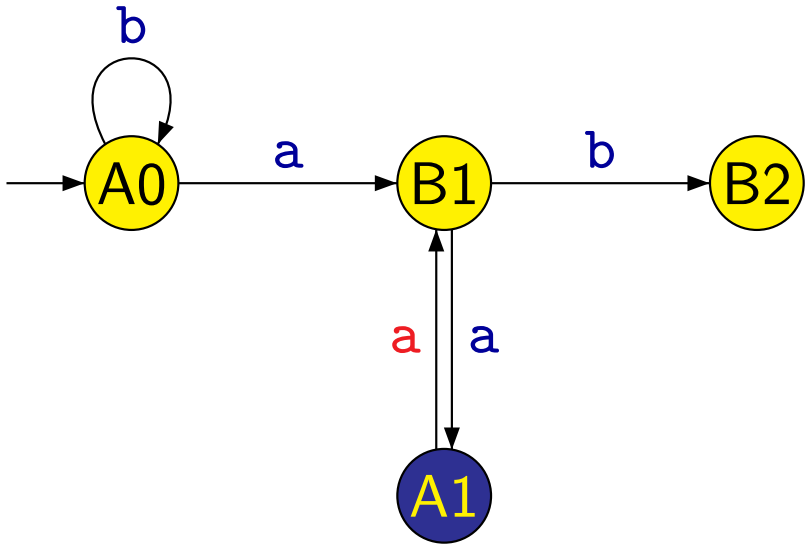




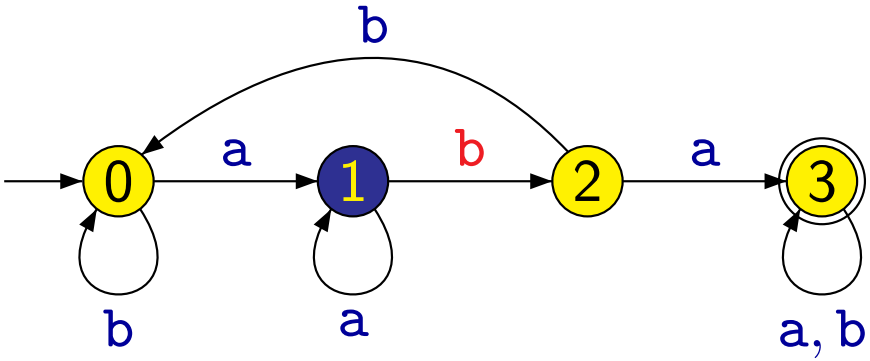
Automat pro průnik jazyků

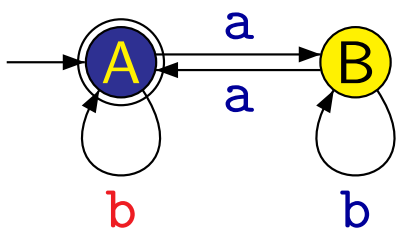


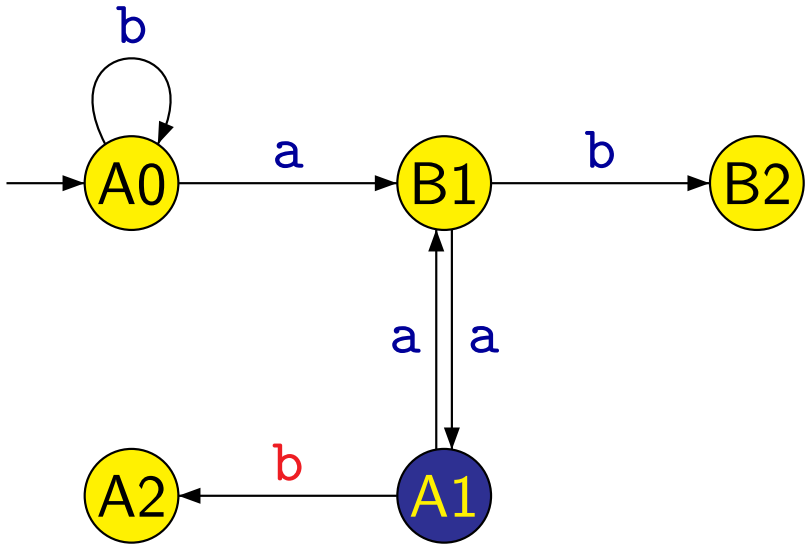




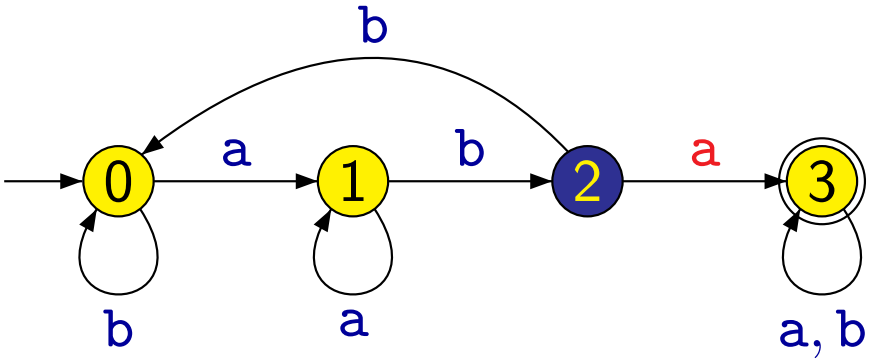
Automat pro průnik jazyků

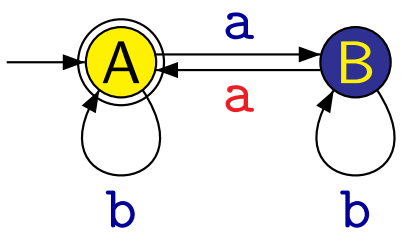


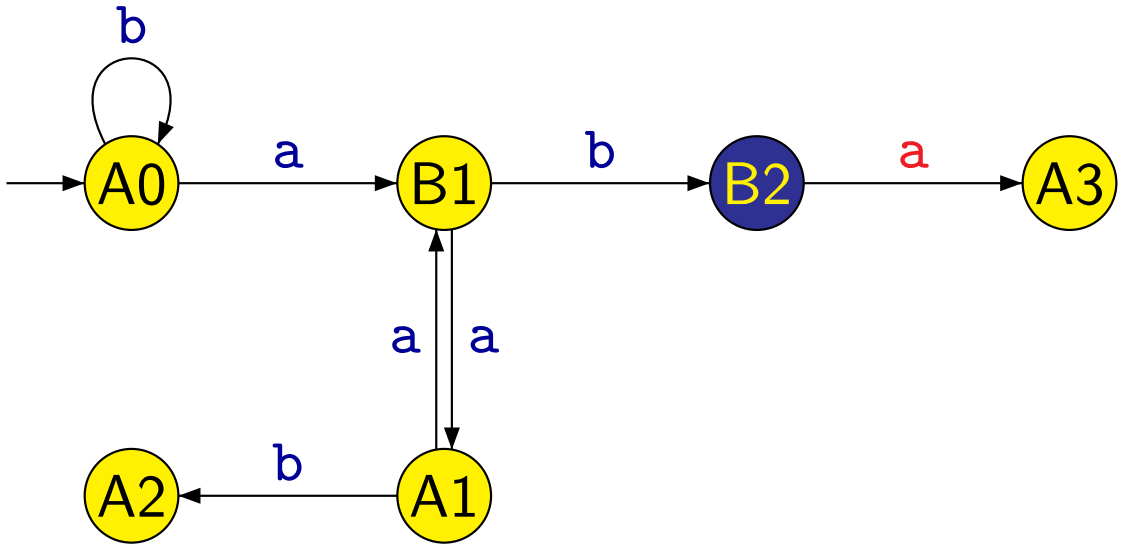




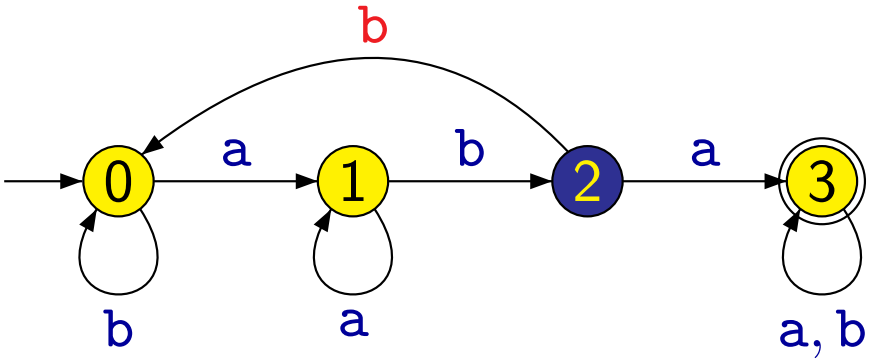
Automat pro průnik jazyků

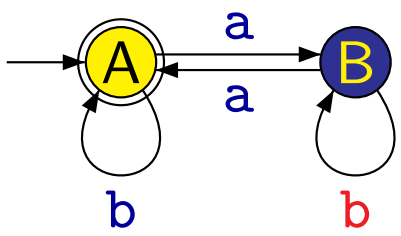


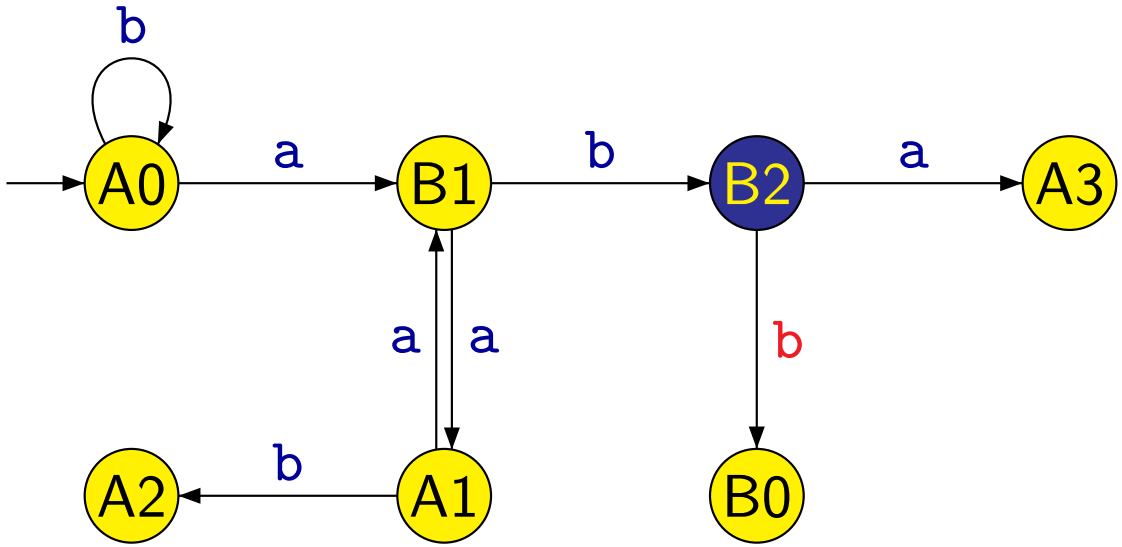




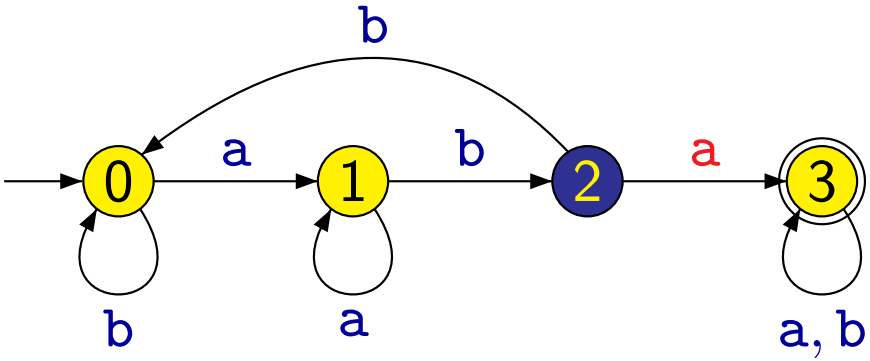
Automat pro průnik jazyků

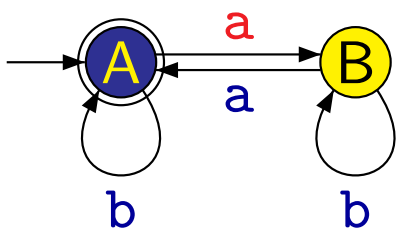


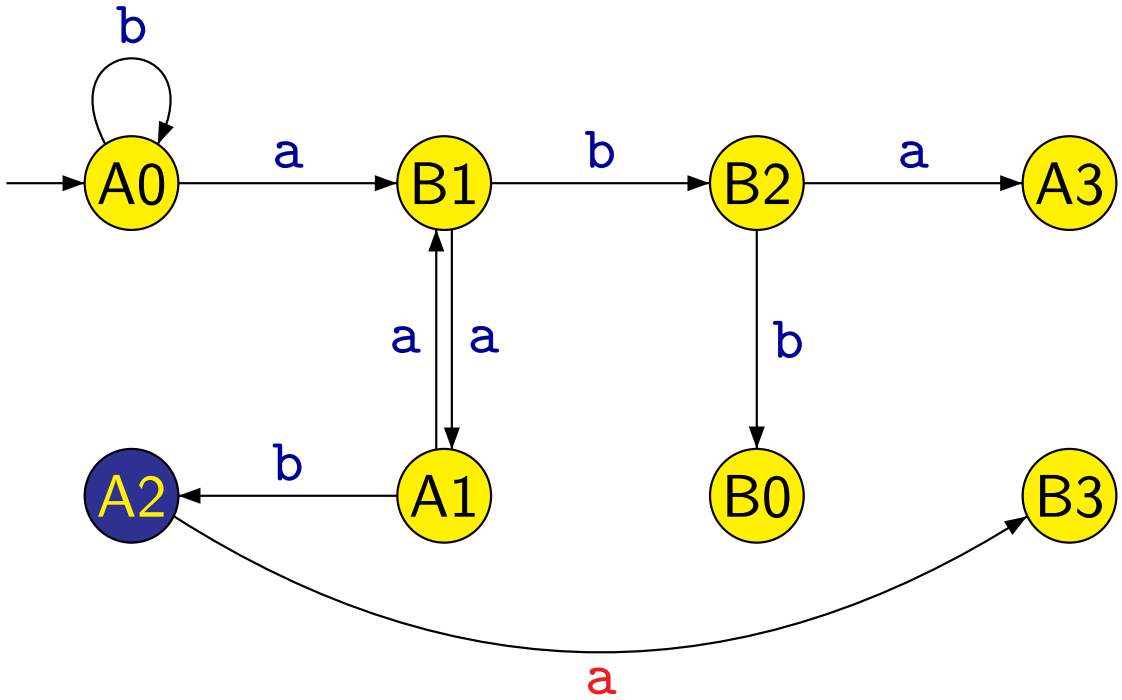




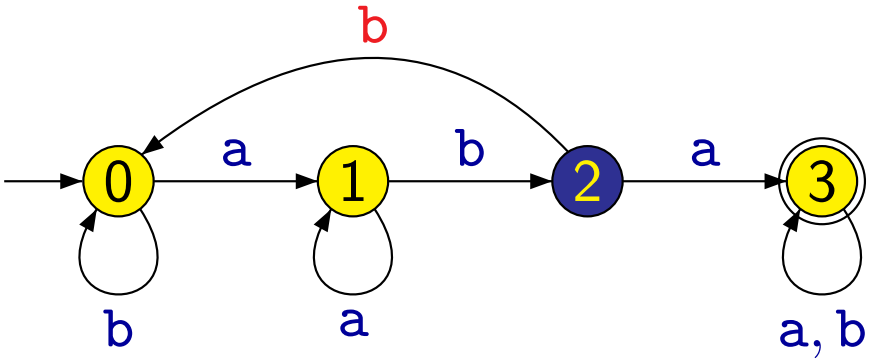
Automat pro průnik jazyků

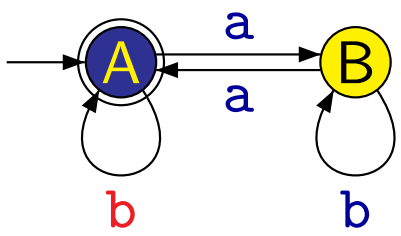


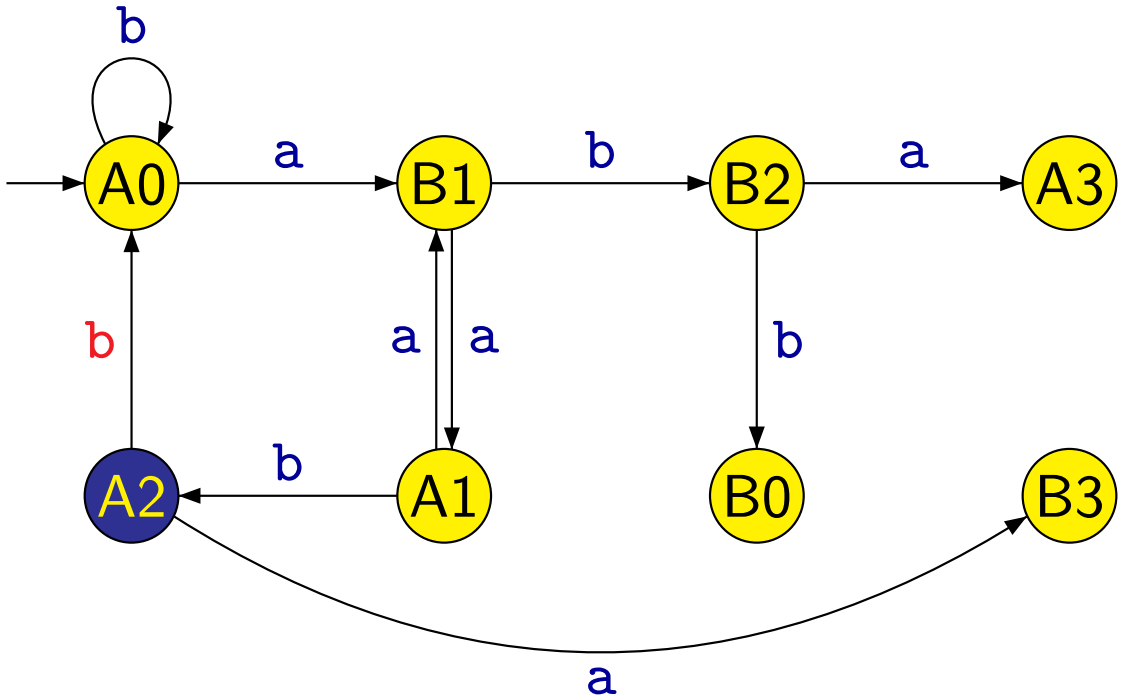




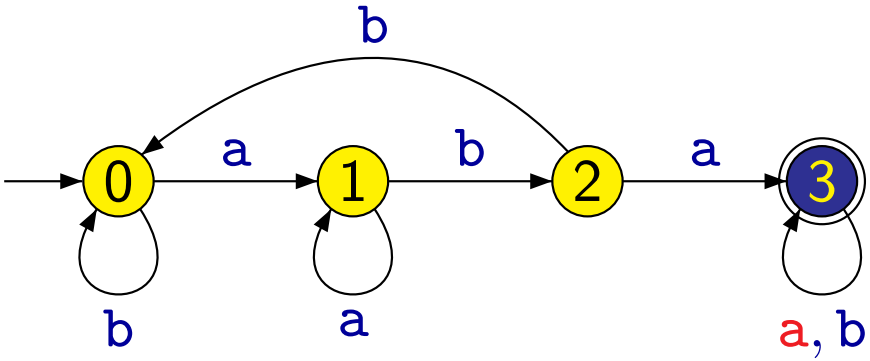
Automat pro průnik jazyků

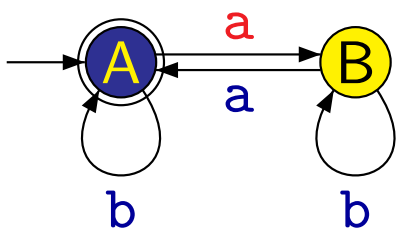


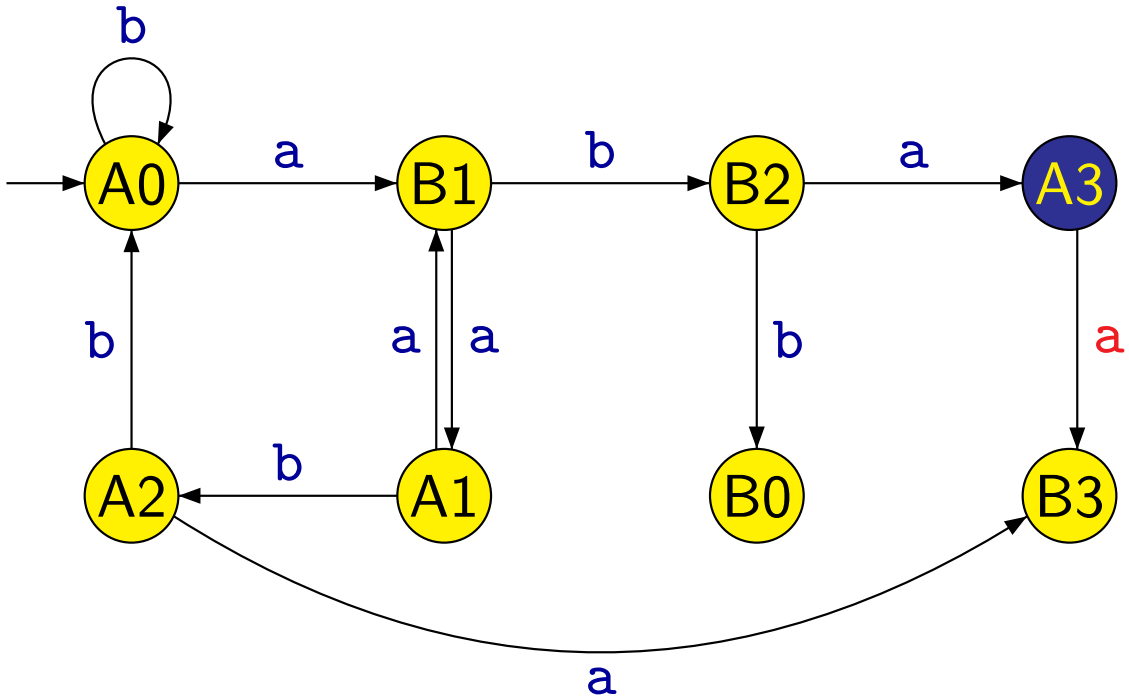




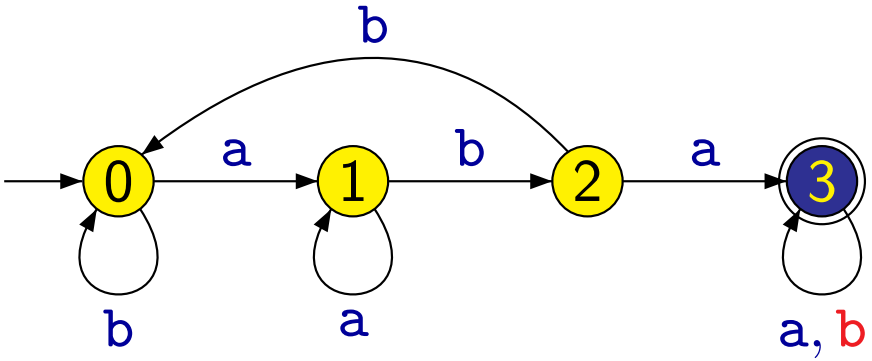
Automat pro průnik jazyků

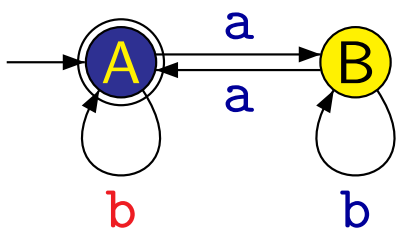


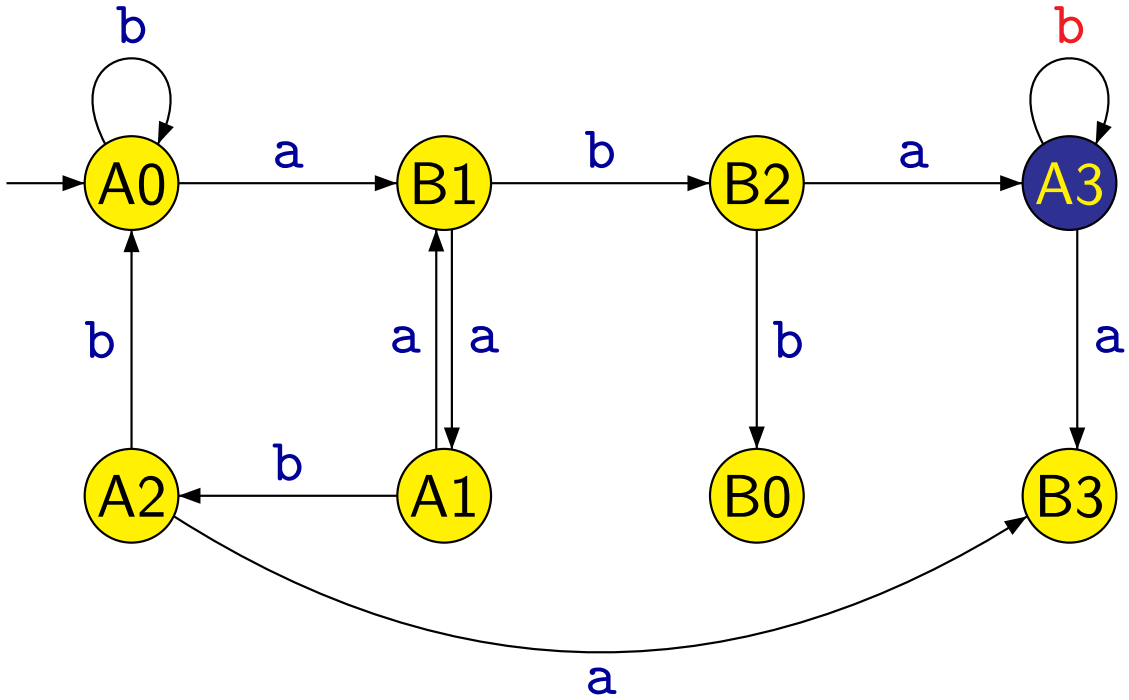




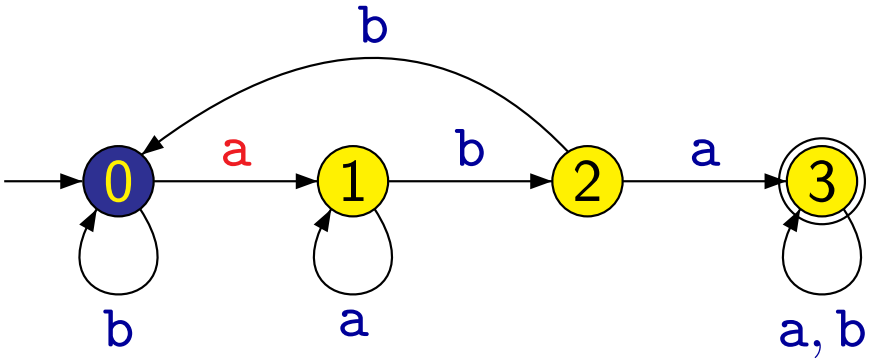
Automat pro průnik jazyků

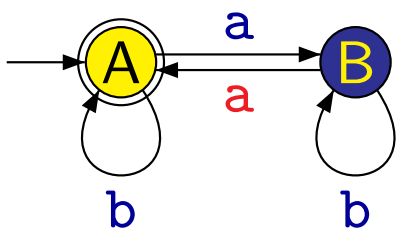


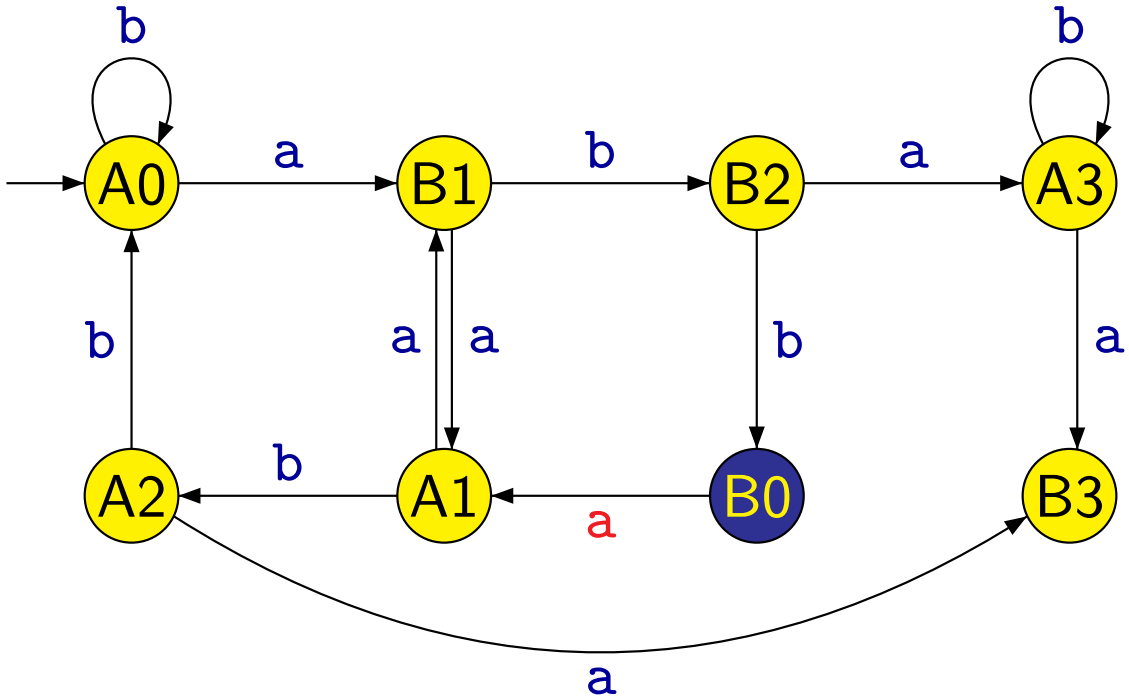




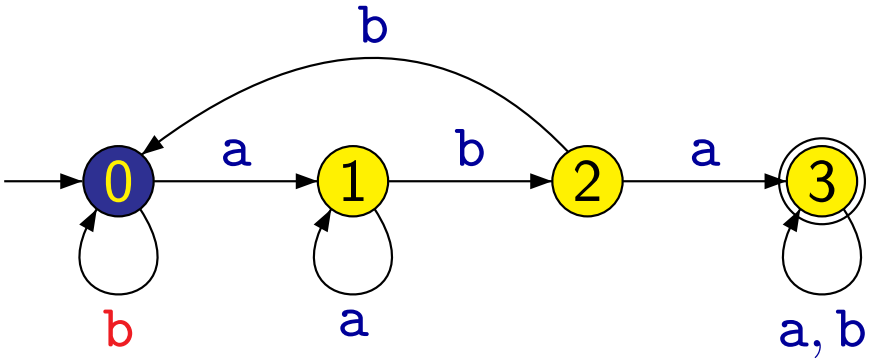
Automat pro průnik jazyků

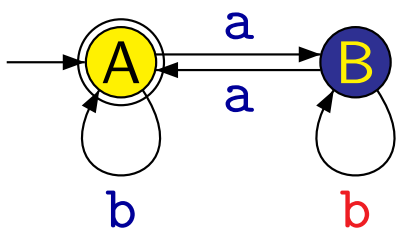


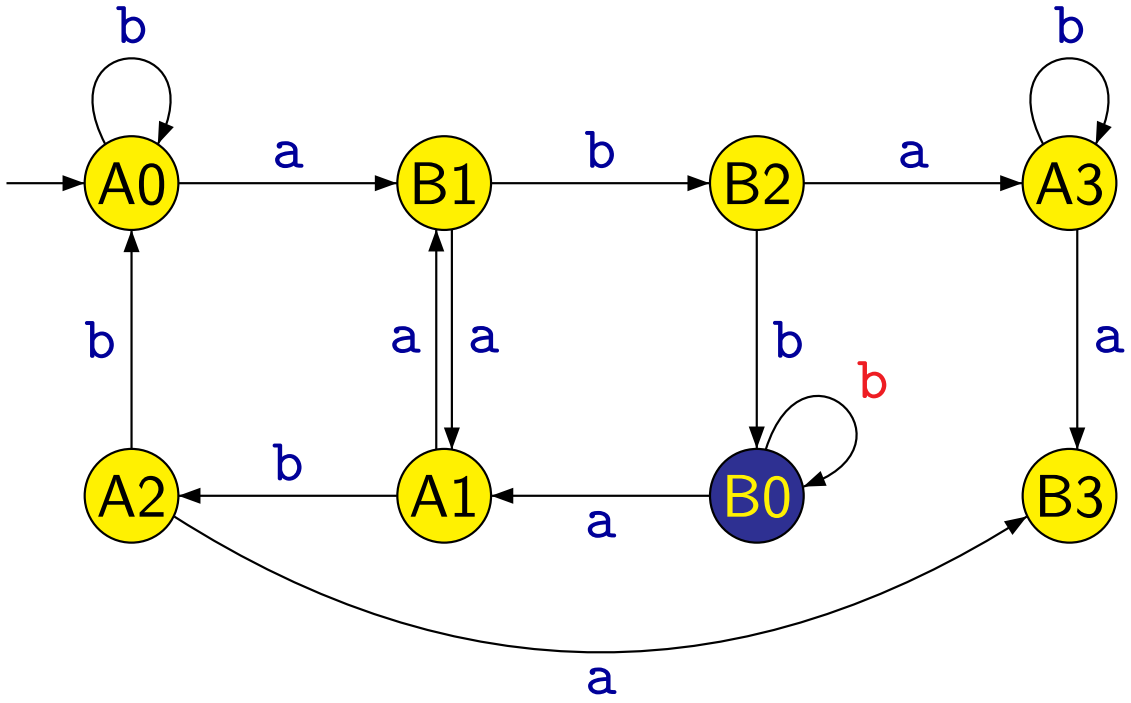




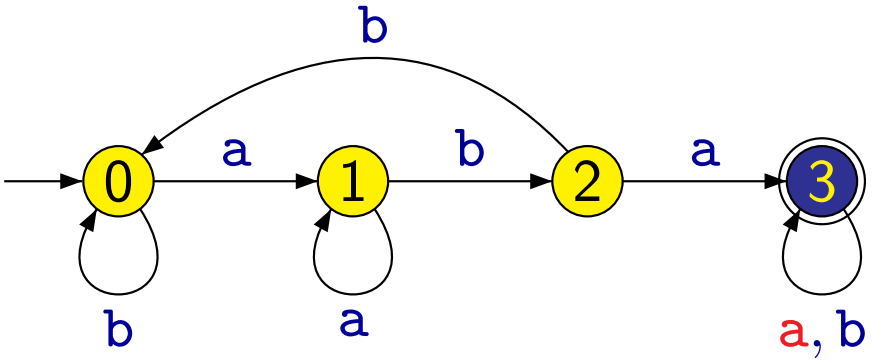
Automat pro průnik jazyků

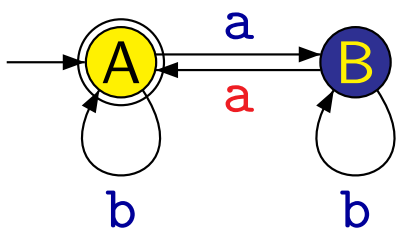


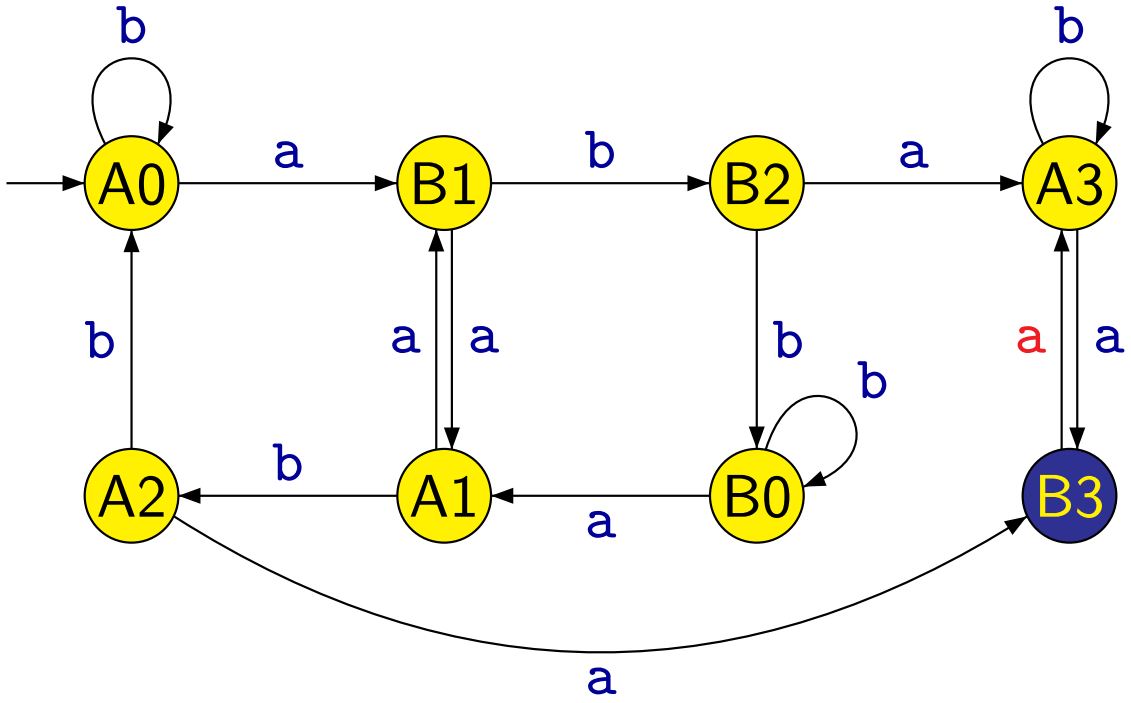




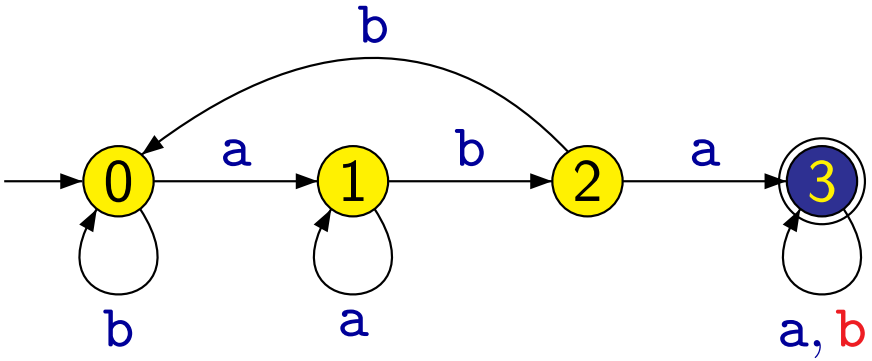
Automat pro průnik jazyků

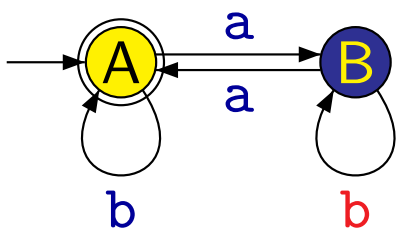


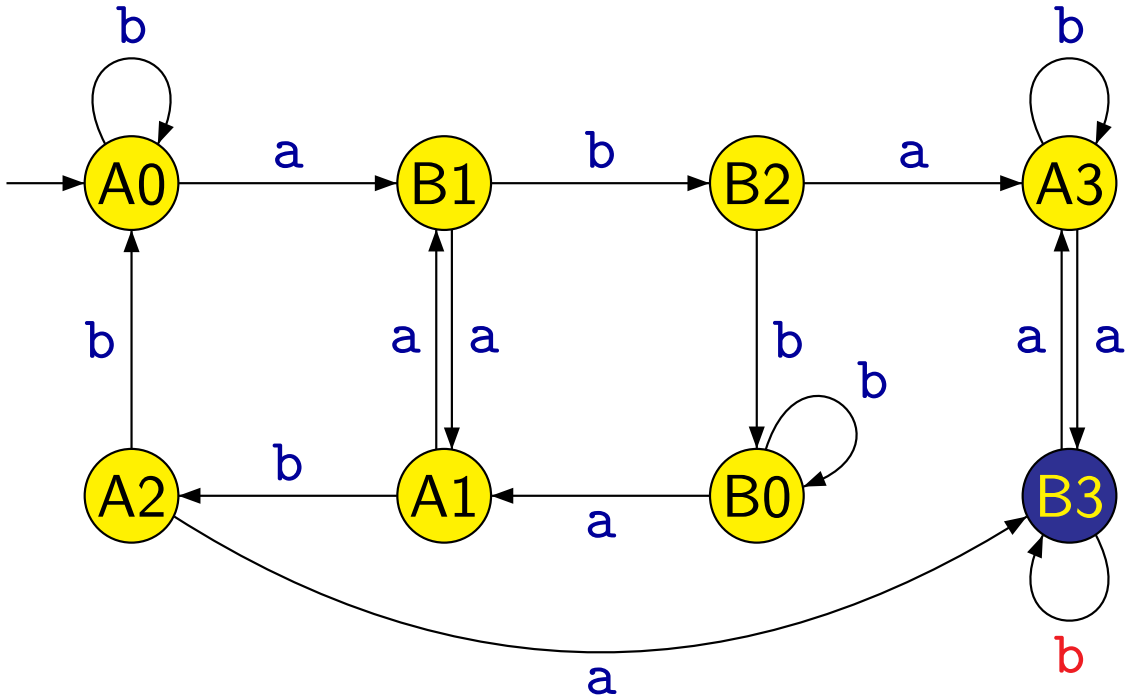




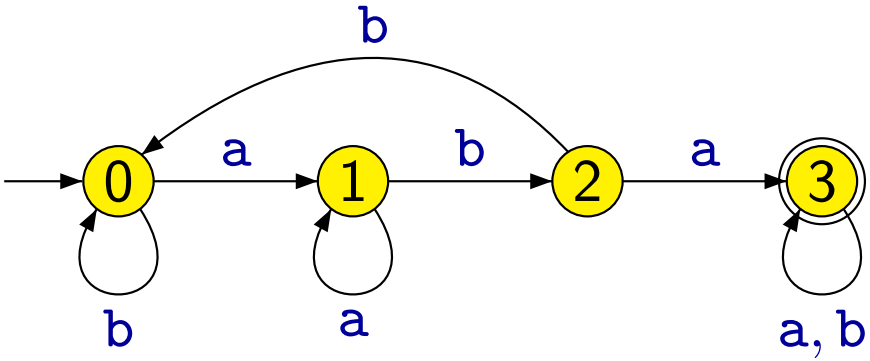
Automat pro průnik jazyků

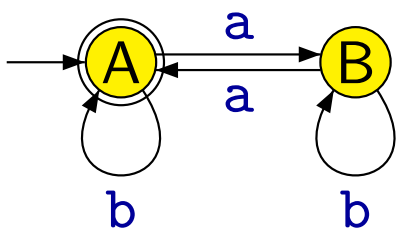


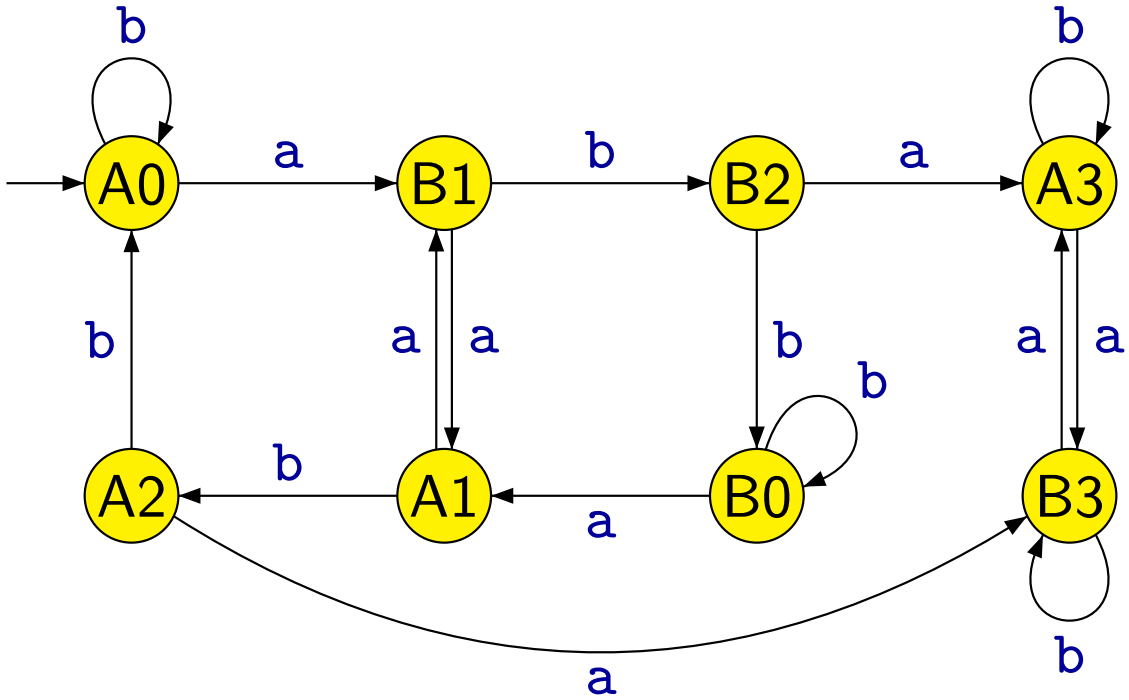




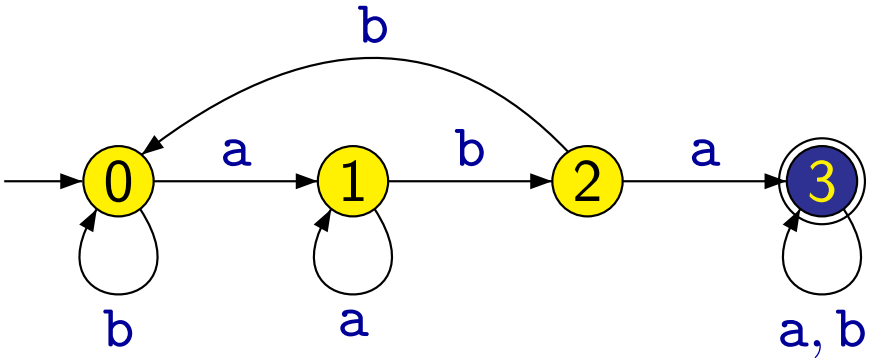
Automat pro průnik jazyků

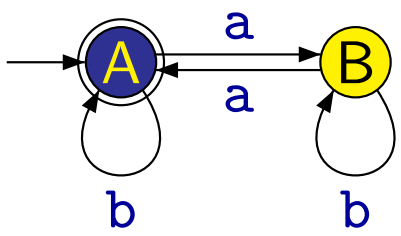


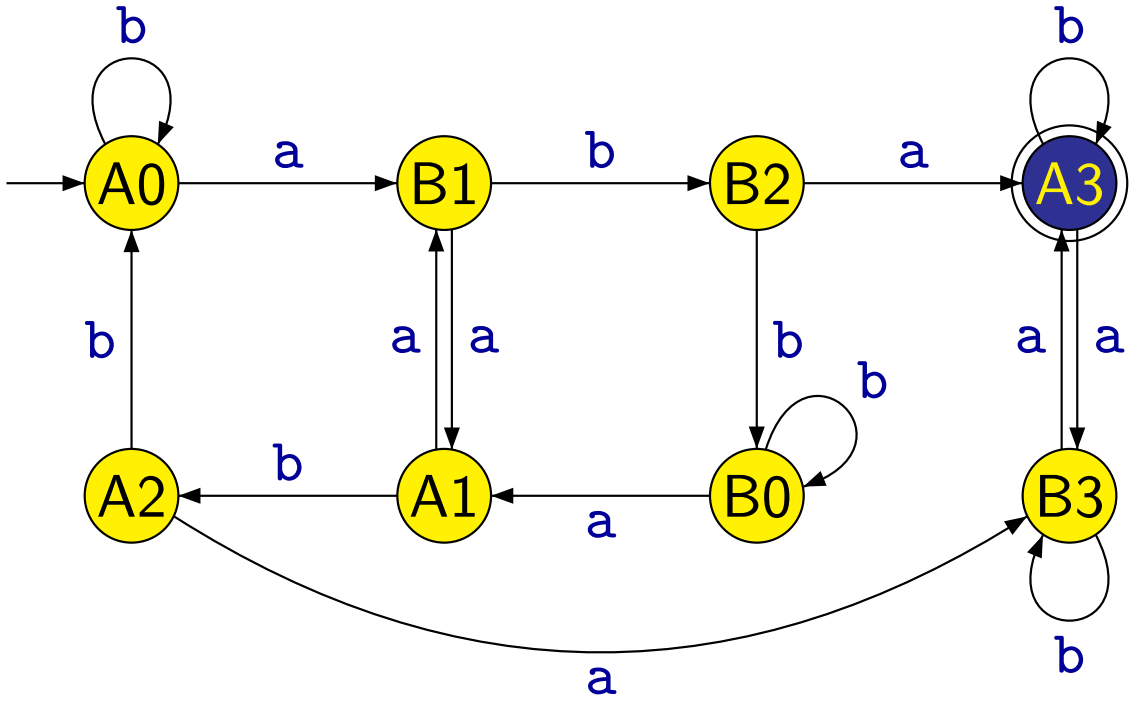




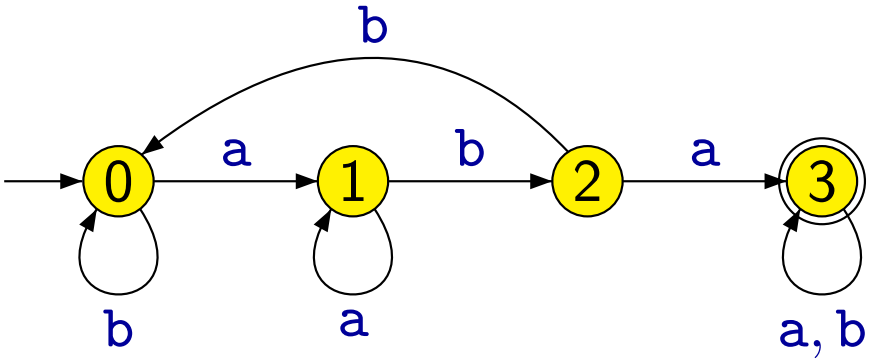
Automat pro průnik jazyků

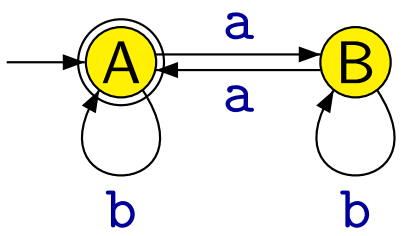


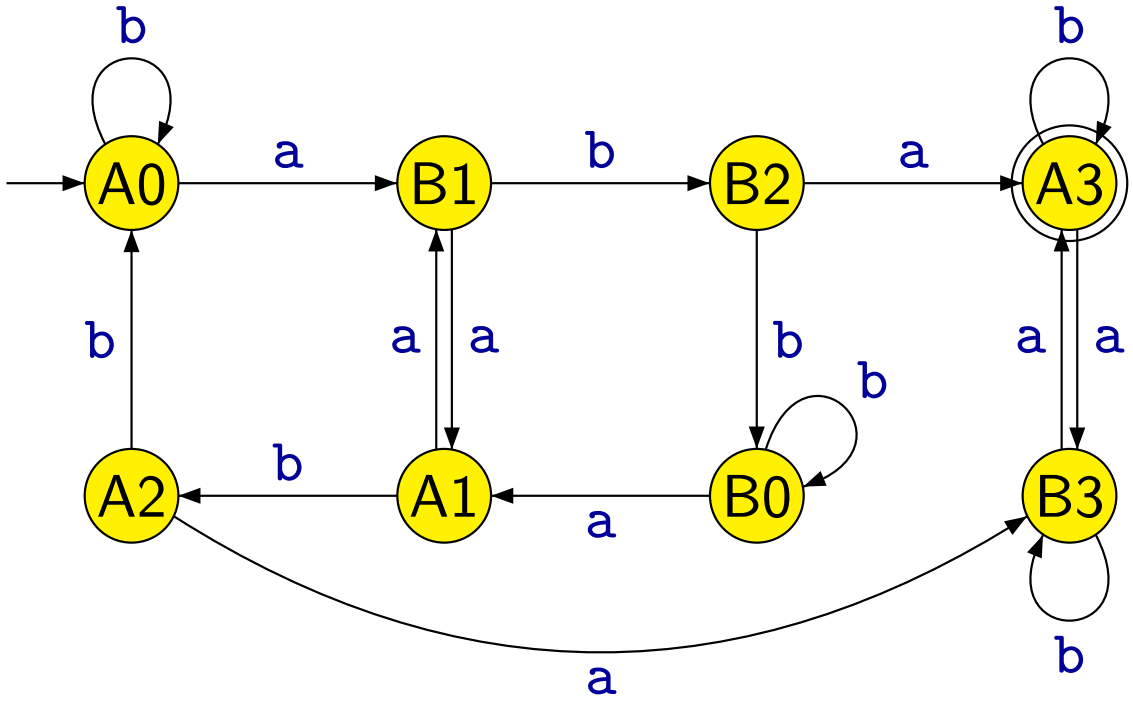




Automat pro průnik jazyků







Automat pro průnik jazyků

Formálně můžeme popsat tuto konstrukci následovně:

Předpokládáme, že máme dva deterministické konečné automaty $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ a $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$.

K nim setrojíme DKA $A = (Q, \Sigma, \delta, q_0, F)$ kde:

- $Q = Q_1 \times Q_2$
- $\delta(\,(q_1, q_2), a\,) = (\,\delta_1(q_1, a), \delta_2(q_2, a)\,)$ pro všechna $q_1 \in Q_1$, $q_2 \in Q_2$, $a \in \Sigma$
- $q_0 = (q_{01}, q_{02})$
- $F = F_1 \times F_2$

Není težké ověřit, že pro libovolné slovo $w \in \Sigma^*$ platí, že $w \in L(A)$ právě tehdy, když $w \in L(A_1)$ a $w \in L(A_2)$, tj.

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

Průnik regulárních jazyků

Věta

Jestliže jazyky $L_1, L_2 \subseteq \Sigma^*$ jsou regulární, pak také jazyk $L_1 \cap L_2$ je regulární.

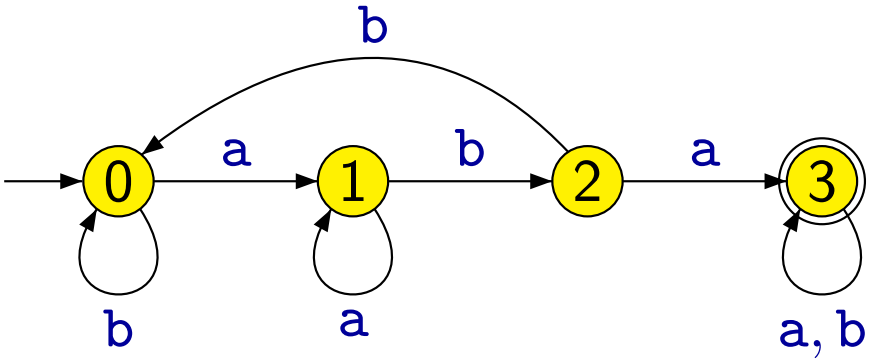
Důkaz: Předpokládejme, že A_1 a A_2 jsou deterministické konečné automaty takové, že

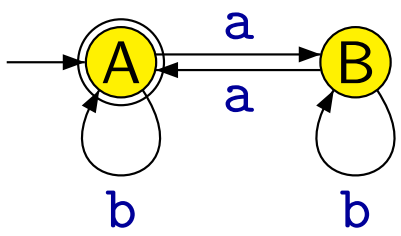
$$L_1 = L(A_1) \qquad L_2 = L(A_2)$$

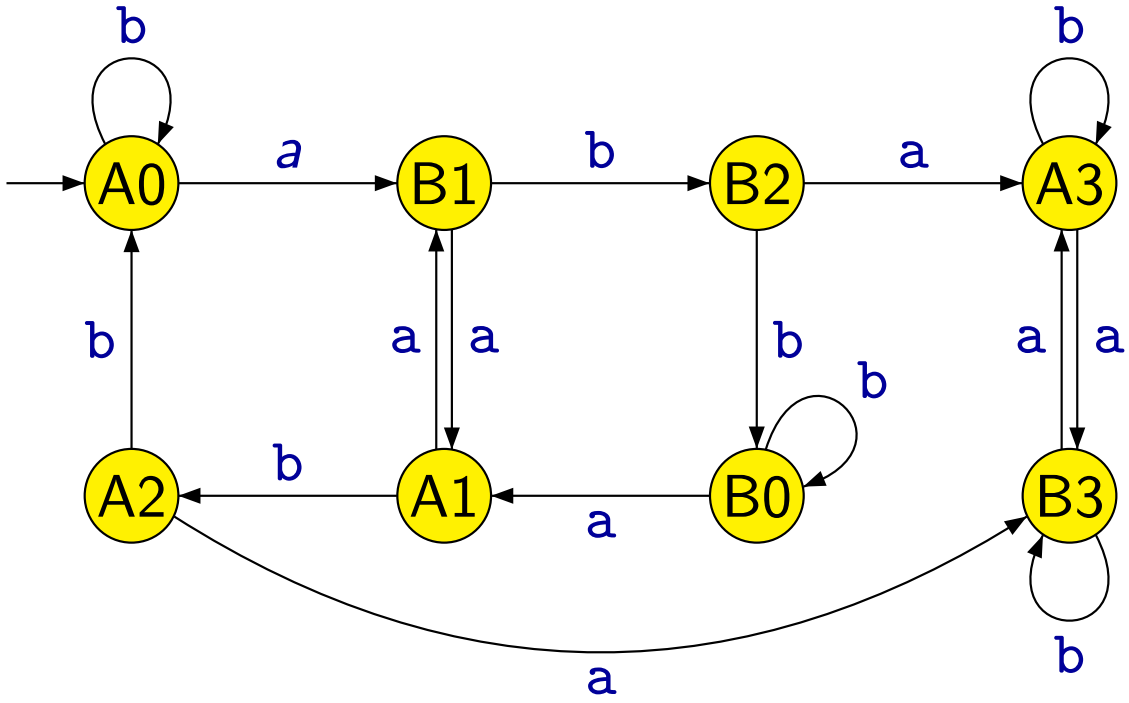
Popsanou konstrukcí k nim můžeme sestrojit deterministický konečný automat A takový, že

$$L(A) = L(A_1) \cap L(A_2) = L_1 \cap L_2$$

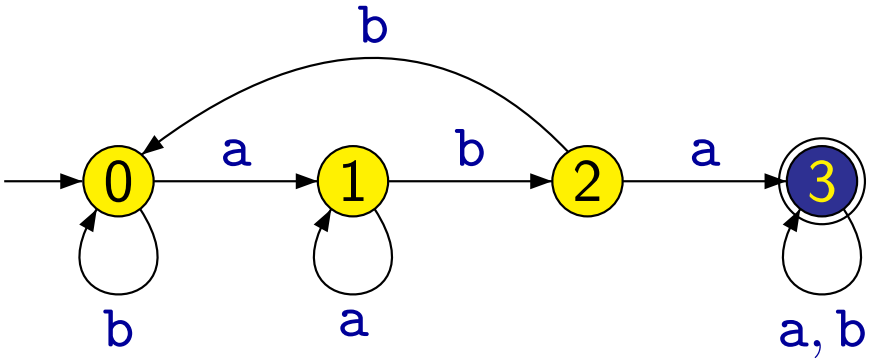
Automat pro sjednocení jazyků

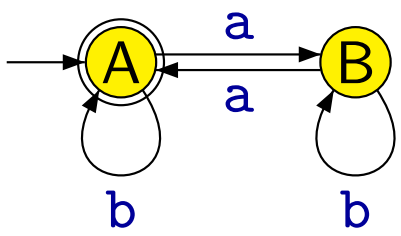


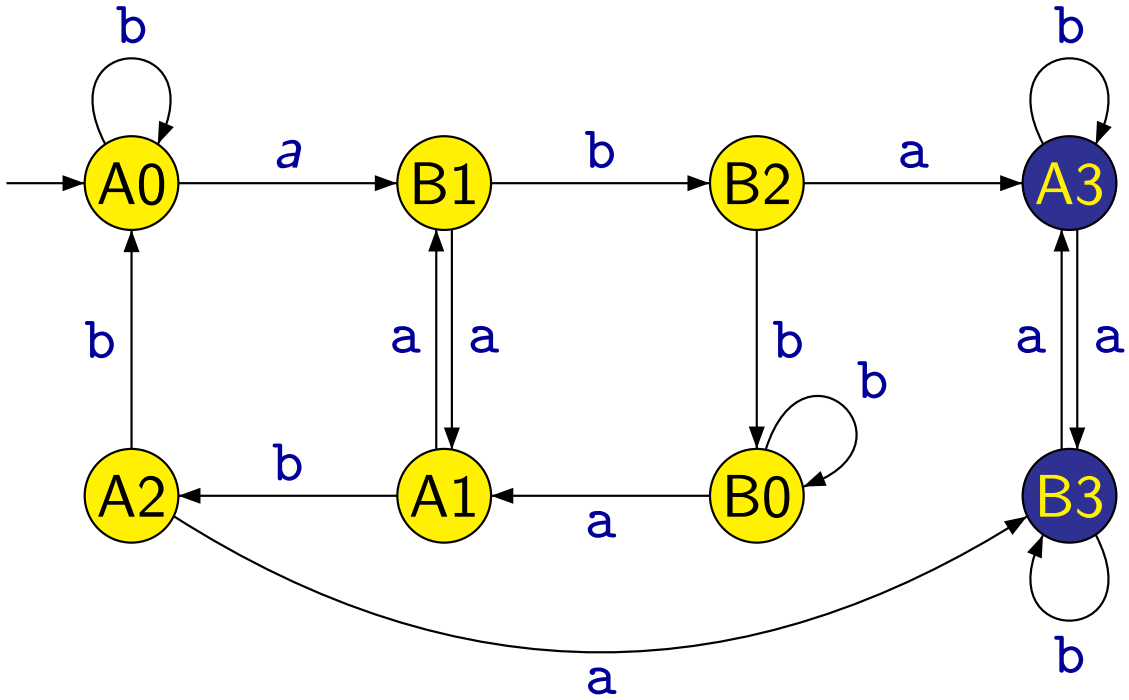




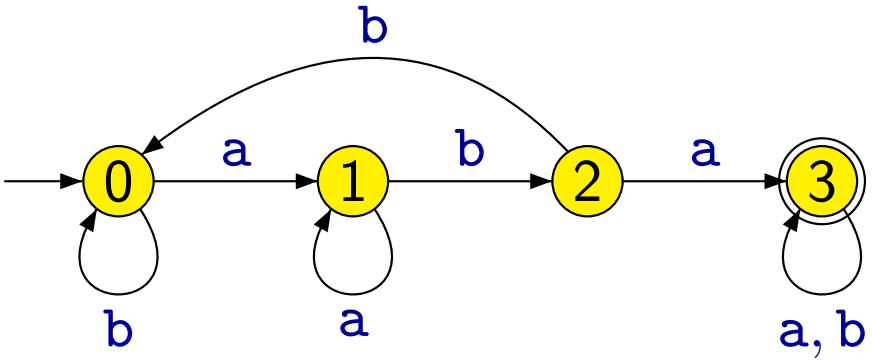
Automat pro sjednocení jazyků

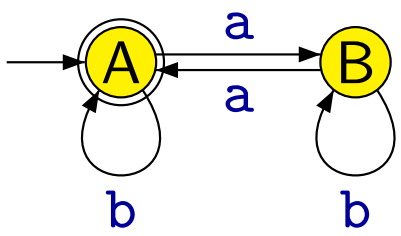


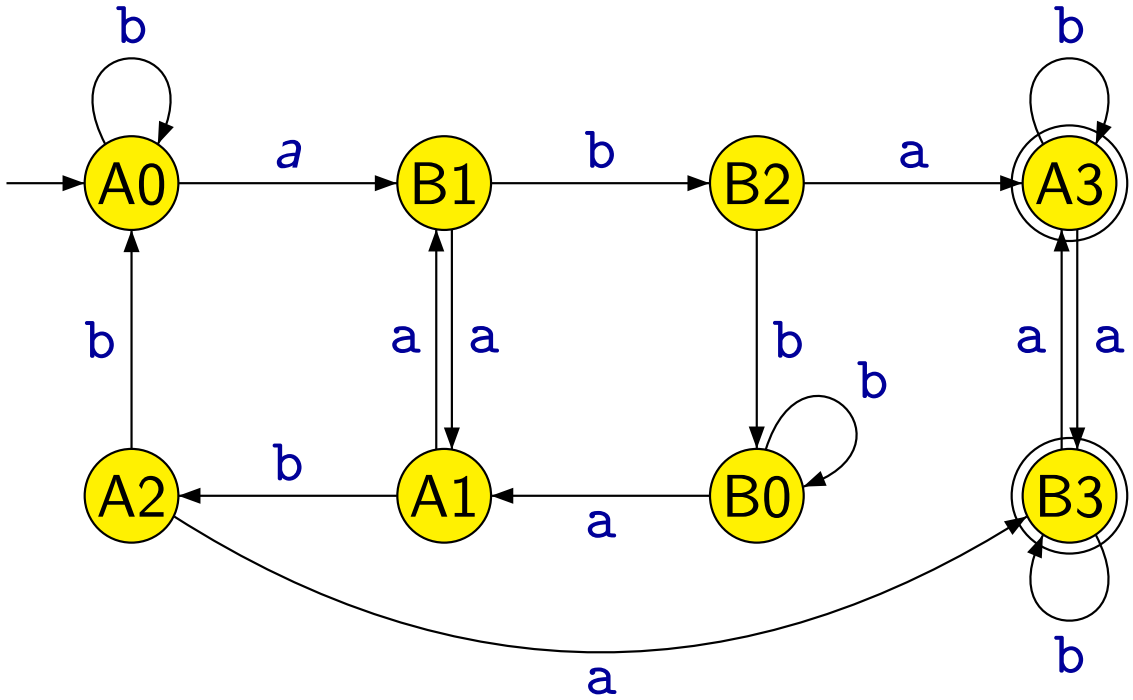




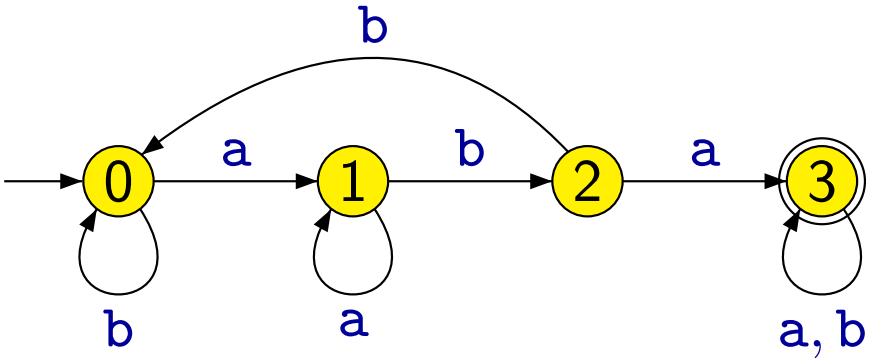
Automat pro sjednocení jazyků

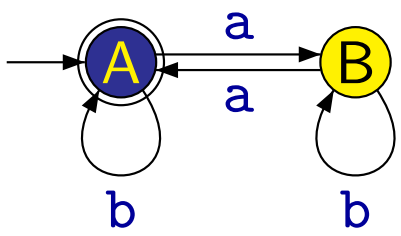


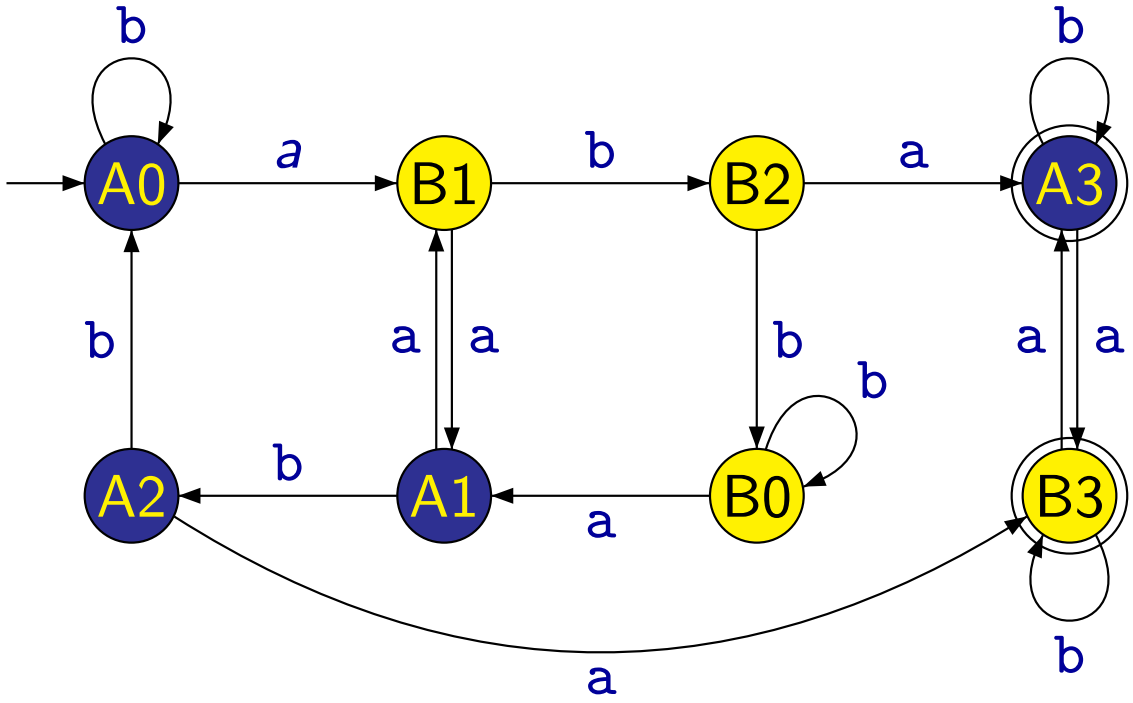




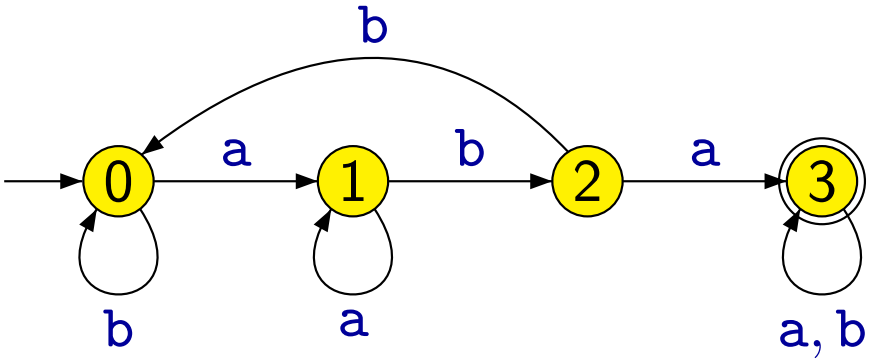
Automat pro sjednocení jazyků

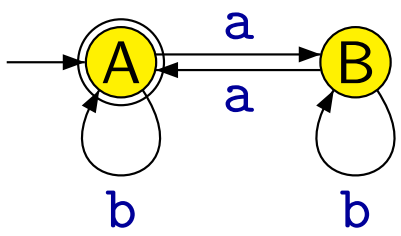


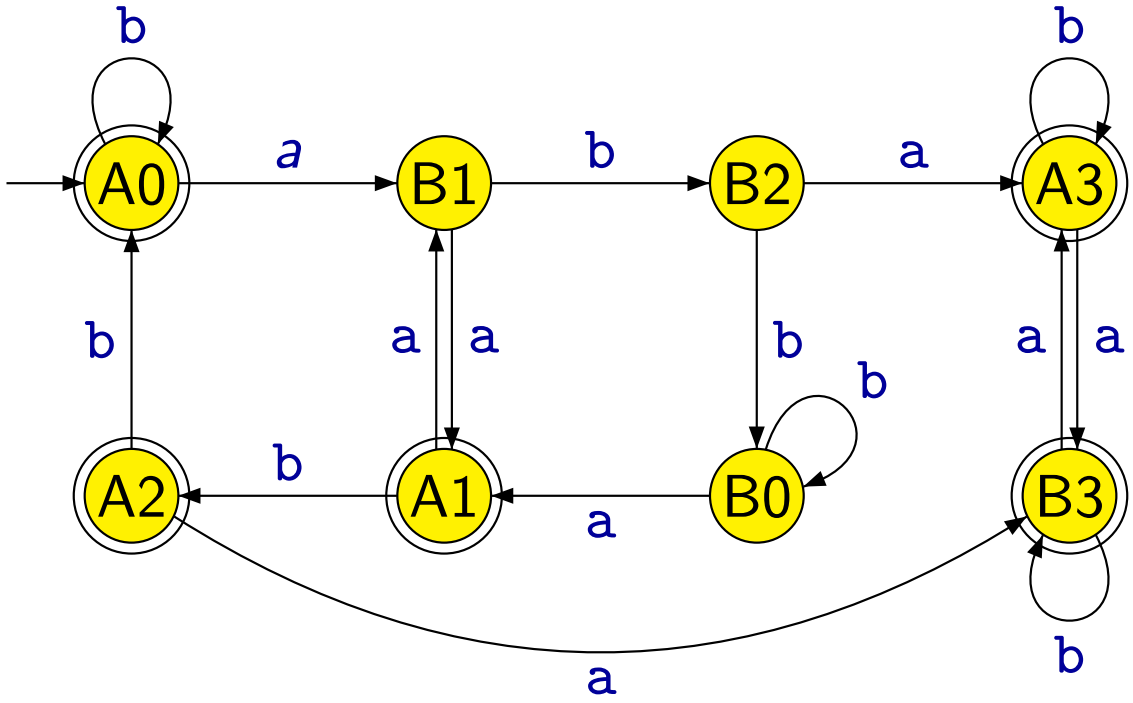




Automat pro sjednocení jazyků







Sjednocení regulárních jazyků

Konstukce automatu A, který přijímá **sjednocení** jazyků přijímaných automaty A_1 a A_2, tj. jazyk

$$L(A_1) \cup L(A_1)$$

je téměř stejná jako v případě automatu přijímajícího $L(A_1) \cap L(A_2)$.

Jediný rozdíl je v definici množiny přijímajících stavů:

- $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

Sjednocení regulárních jazyků

Konstukce automatu A, který přijímá **sjednocení** jazyků přijímaných automaty A_1 a A_2, tj. jazyk

$$L(A_1) \cup L(A_1)$$

je téměř stejná jako v případě automatu přijímajícího $L(A_1) \cap L(A_2)$.

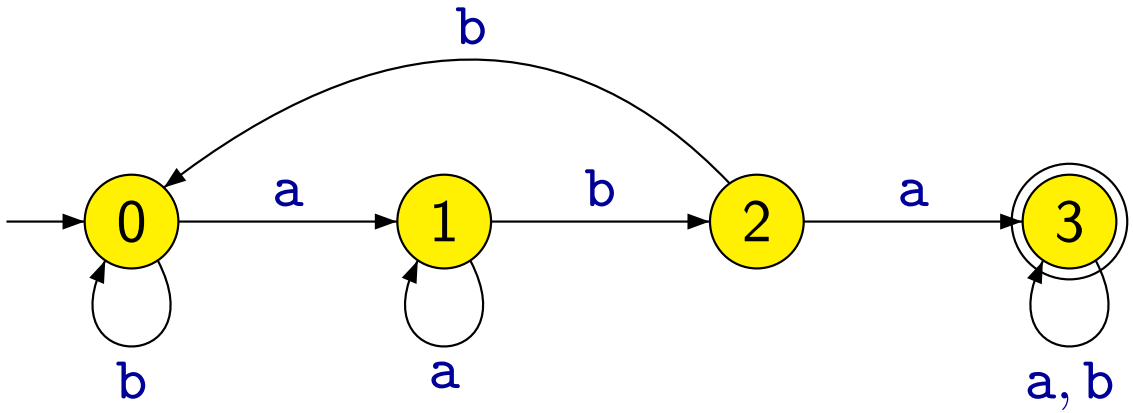
Jediný rozdíl je v definici množiny přijímajících stavů:

- $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

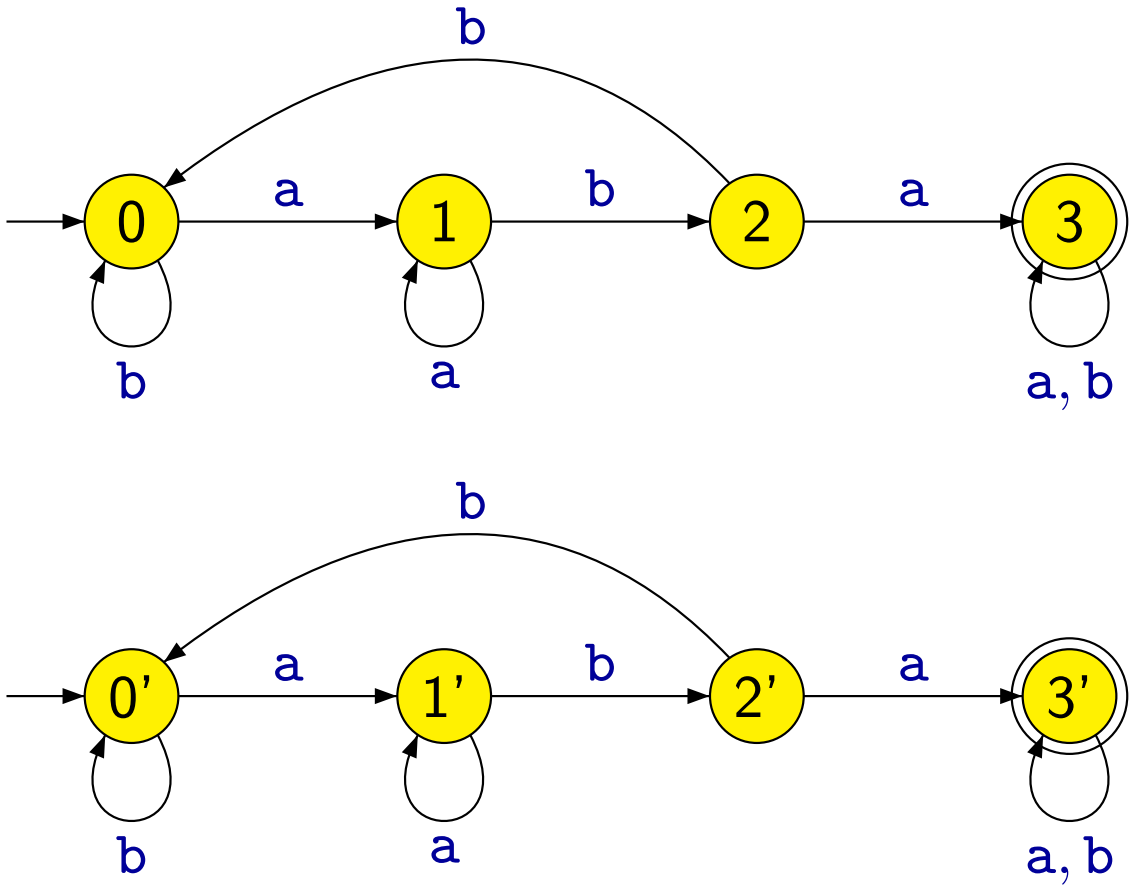
Věta

Jestliže jazyky $L_1, L_2 \subseteq \Sigma^*$ jsou regulární, pak také jazyk $L_1 \cup L_2$ je regulární.

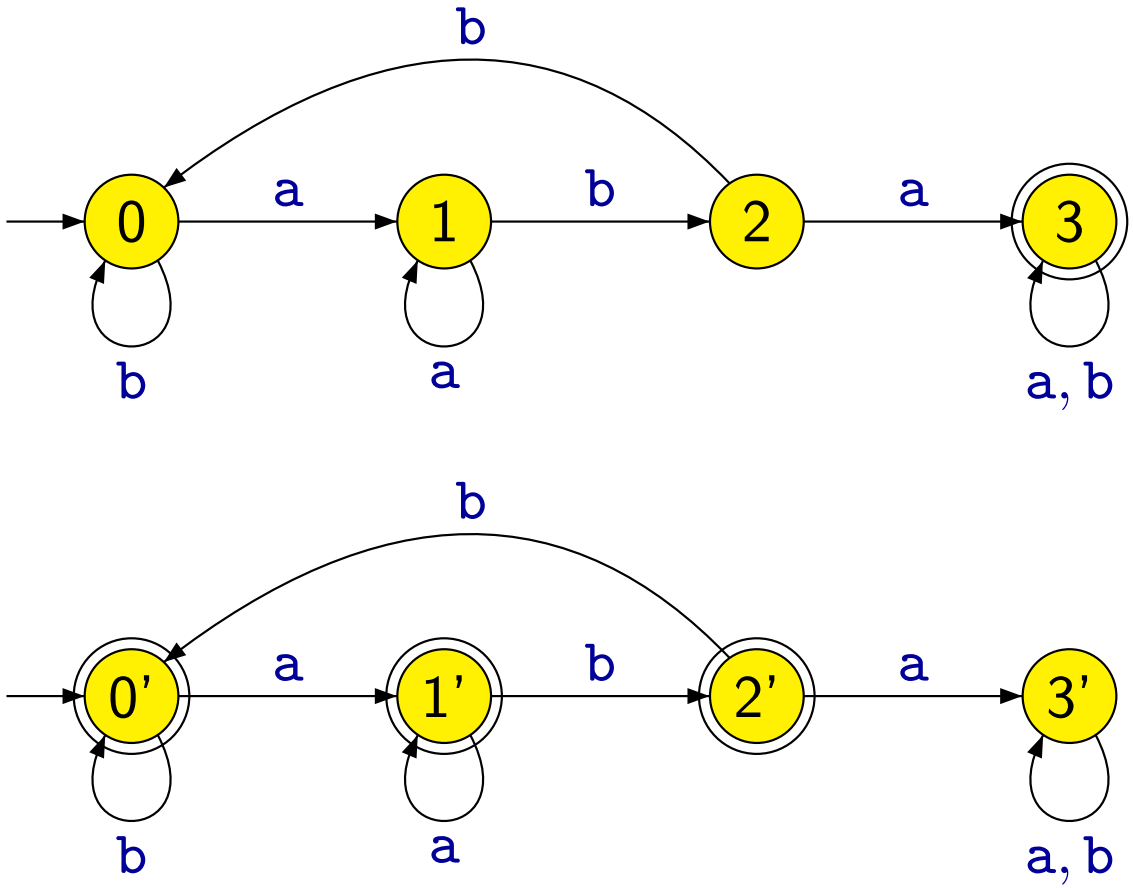
Automat pro doplněk jazyka



Automat pro doplněk jazyka



Automat pro doplněk jazyka



Doplněk regulárního jazyka

K DKA $A = (Q, \Sigma, \delta, q_0, F)$ sestrojíme DKA $A' = (Q, \Sigma, \delta, q_0, Q - F)$.

Je očividné, že pro každé slovo $w \in \Sigma^*$ platí, že $w \in L(A')$ právě tehdy, když $w \notin L(A)$, tj.

$$L(A') = \overline{L(A)}$$

Doplněk regulárního jazyka

K DKA $A = (Q, \Sigma, \delta, q_0, F)$ sestrojíme DKA $A' = (Q, \Sigma, \delta, q_0, Q - F)$.

Je očividné, že pro každé slovo $w \in \Sigma^*$ platí, že $w \in L(A')$ právě tehdy, když $w \notin L(A)$, tj.

$$L(A') = \overline{L(A)}$$

Věta

Jestliže jazyk L je regulární, pak také jeho doplněk $\overline{L}$ je regulární.