Vyhledávání v textu

Problém

Vstup: t – dlouhý text, s – hledané řetězec

Výstup: ANO – pokud se řetězec s nachází v textu t,
NE – pokud se tam nenachází

Například chceme zjistit, zda se v textu `aaababaabab` nachází řetězec `abaa`.

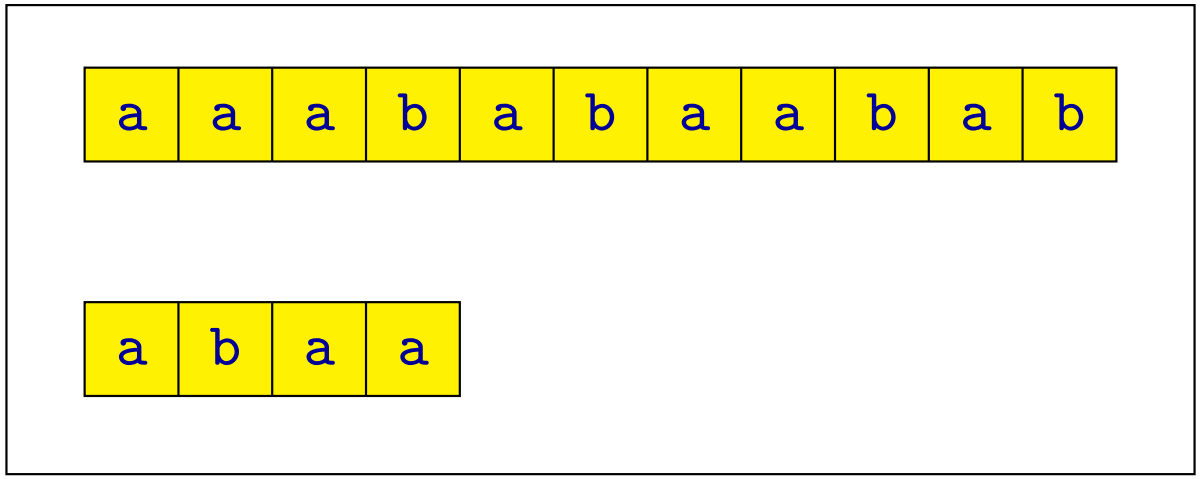
Vyhledávání v textu

Jednoduchý algoritmus, který nás asi napadne v první chvíli:

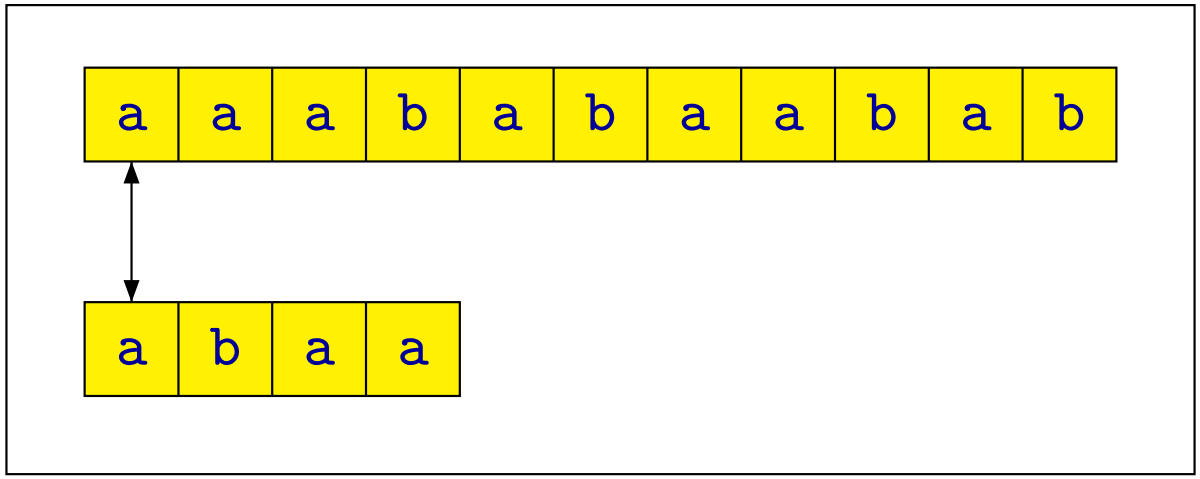
NAIVNÍ-VYHLEDÁVÁNÍ(t, s)

```
 1  n ← length(t)
 2  m ← length(s)
 3  for i ← 0 to n − m
 4        do nalezen ← TRUE
 5           for j ← 0 to m − 1
 6                 do if s[j] ≠ t[i + j]
 7                       then nalezen ← FALSE; break
 8           if nalezen
 9              then return TRUE
10  return FALSE
```

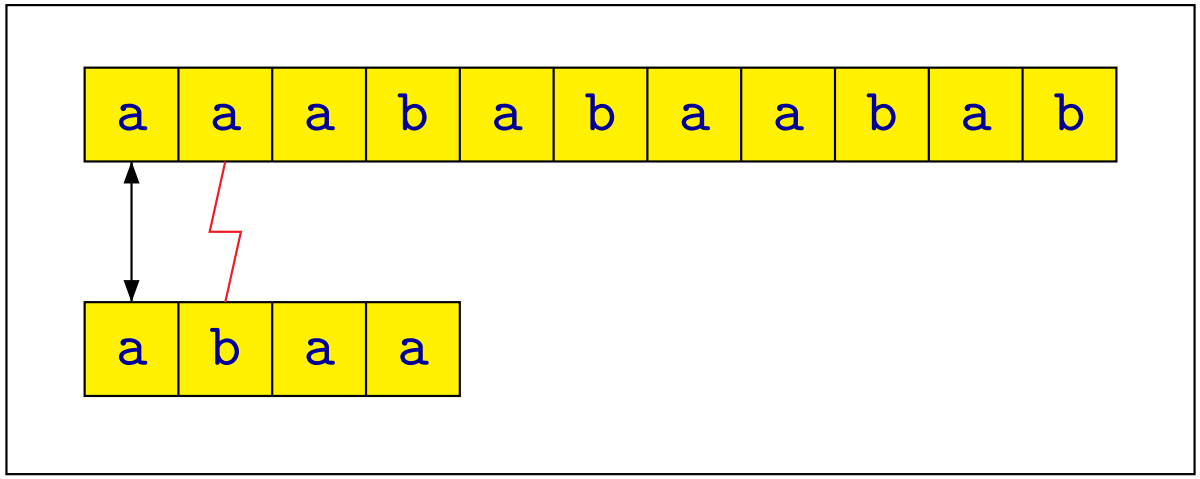
Vyhledávání v textu



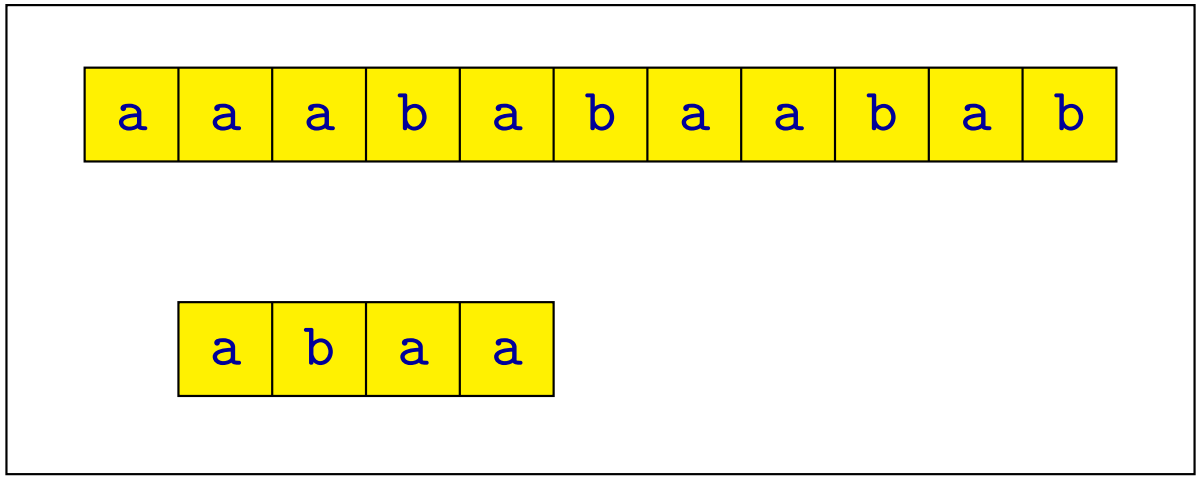
Vyhledávání v textu



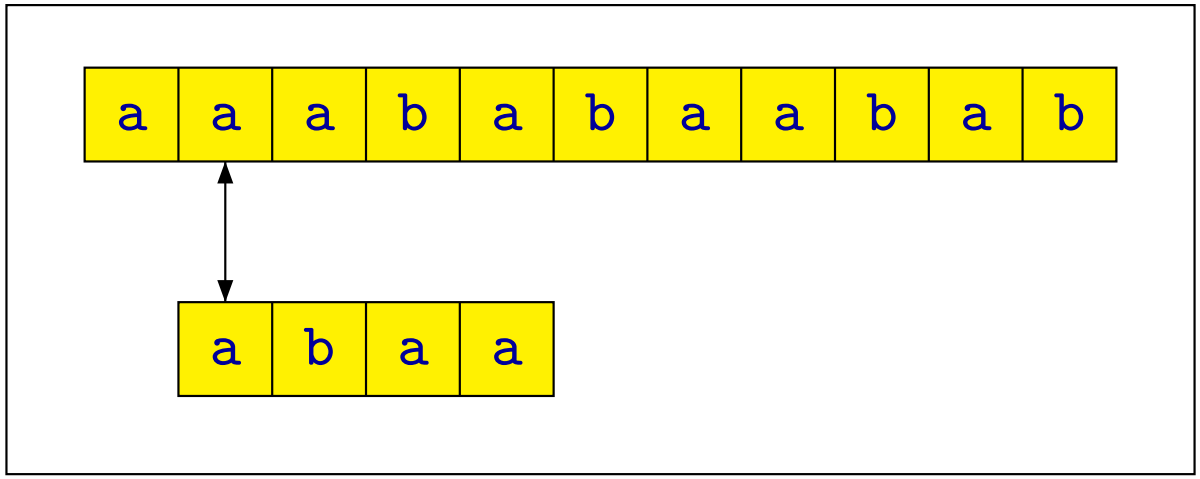
Vyhledávání v textu



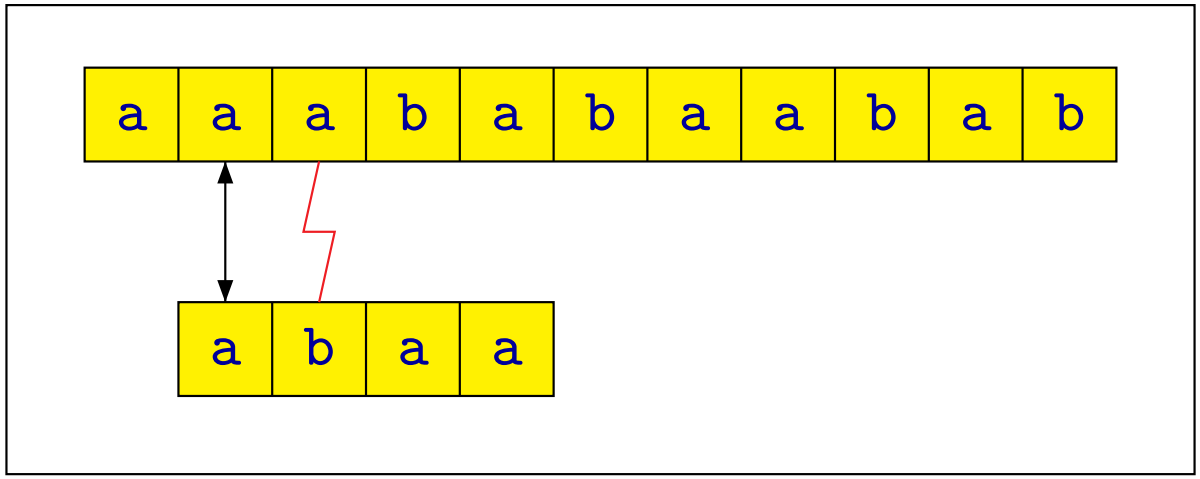
Vyhledávání v textu



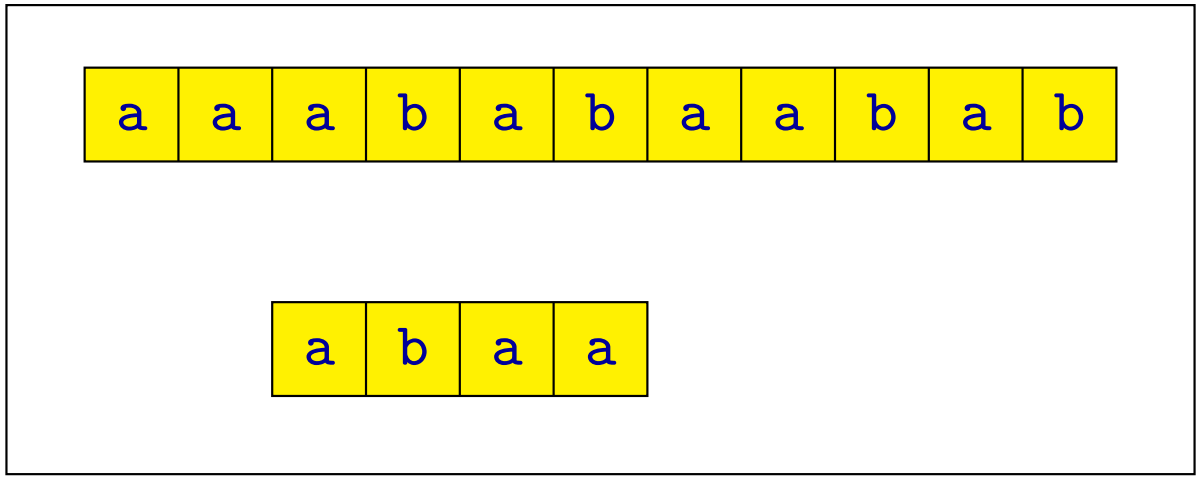
Vyhledávání v textu



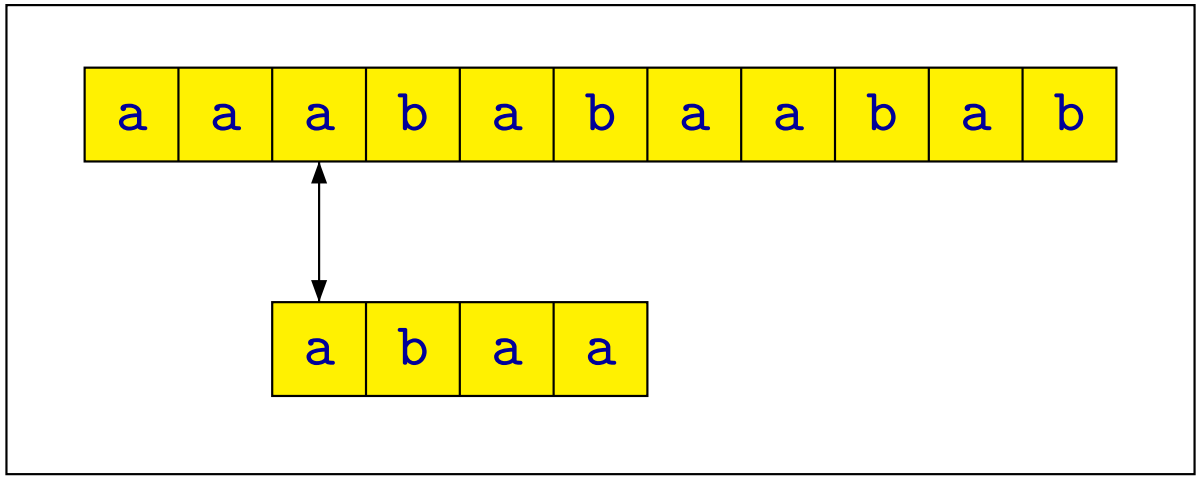
Vyhledávání v textu



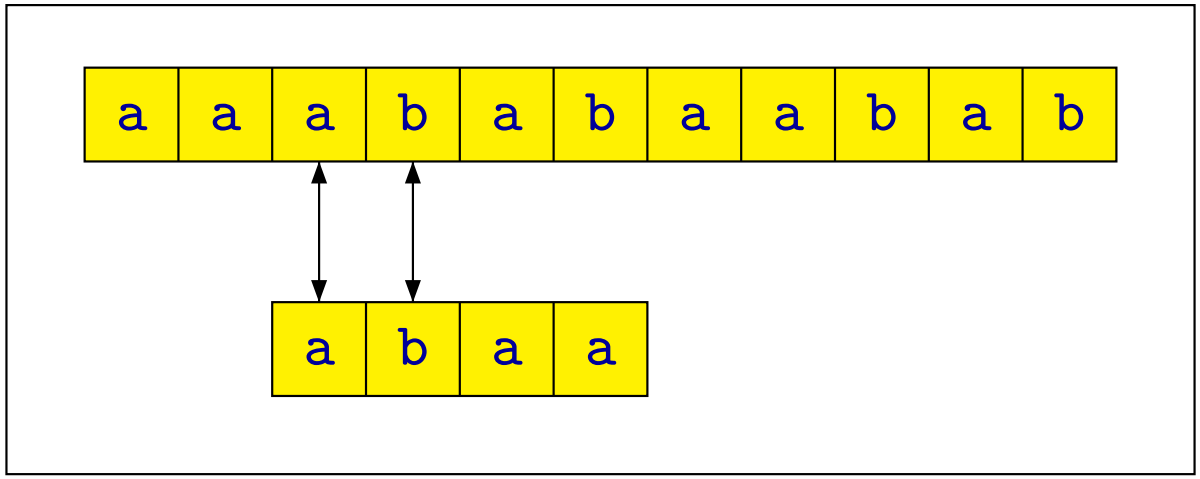
Vyhledávání v textu



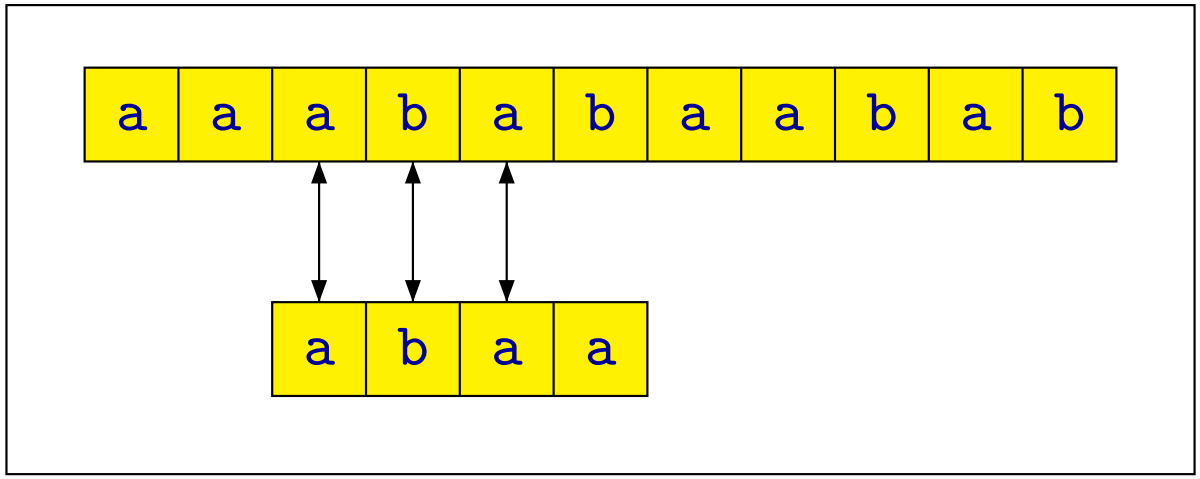
Vyhledávání v textu



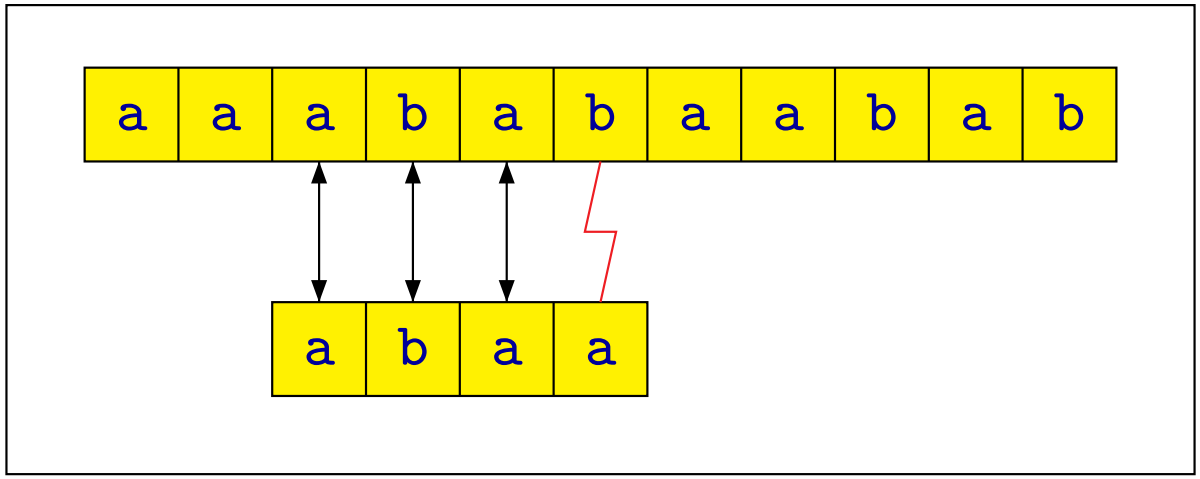
Vyhledávání v textu



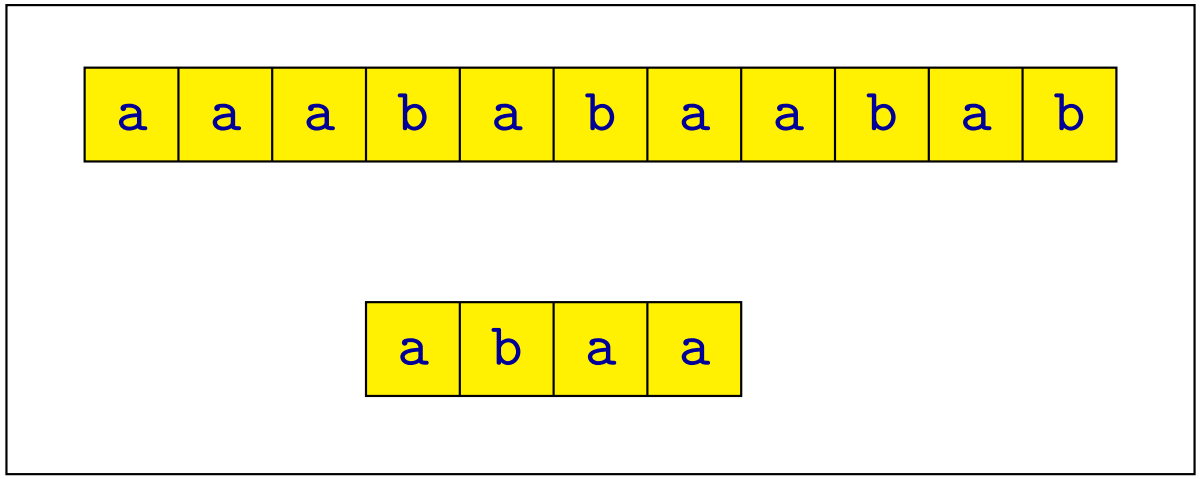
Vyhledávání v textu



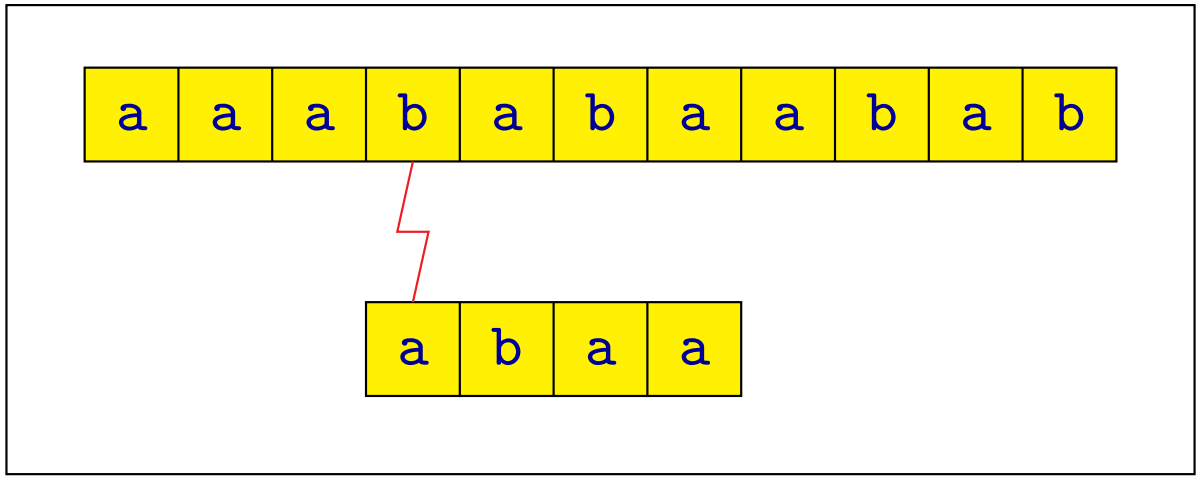
Vyhledávání v textu



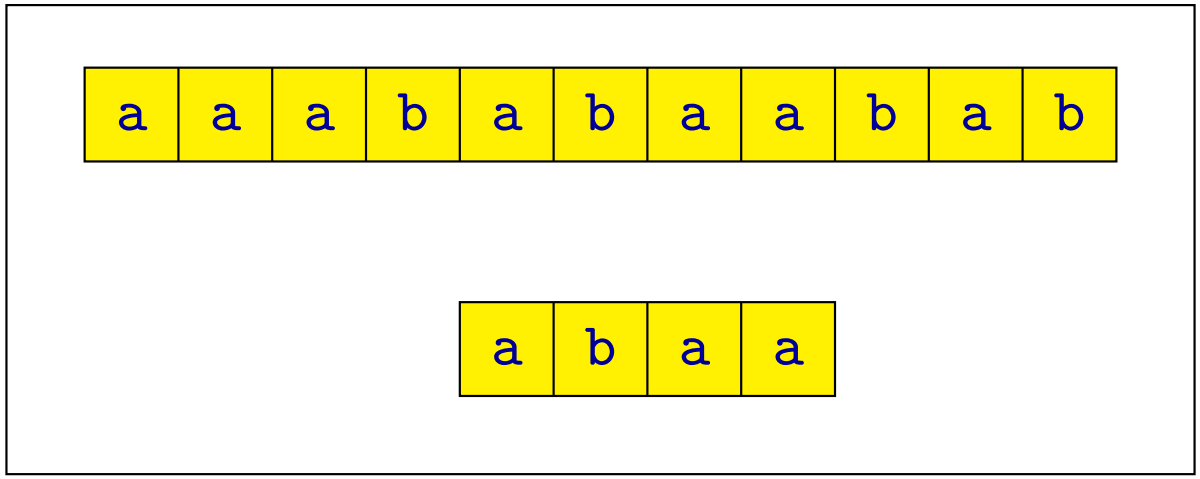
Vyhledávání v textu



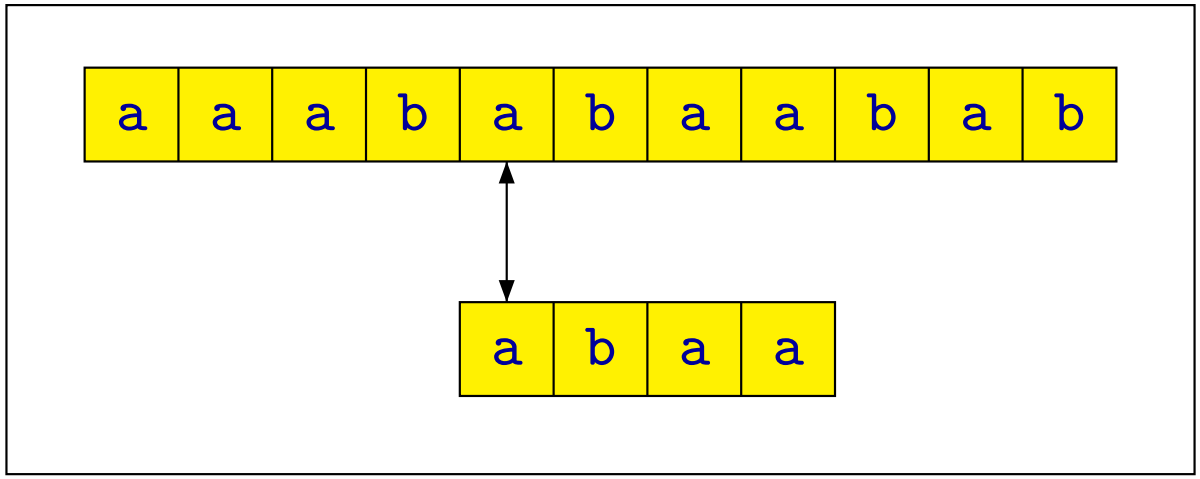
Vyhledávání v textu



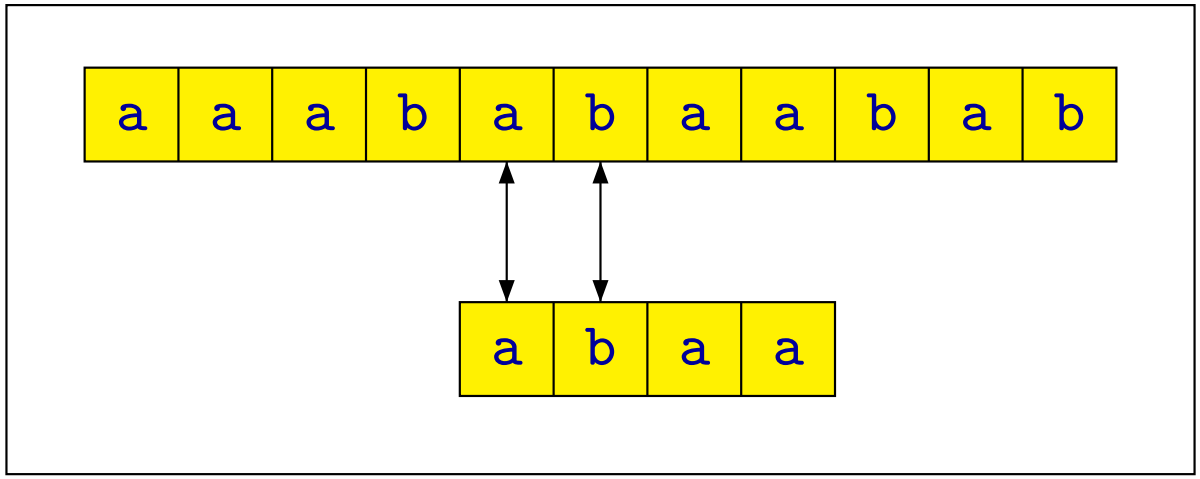
Vyhledávání v textu



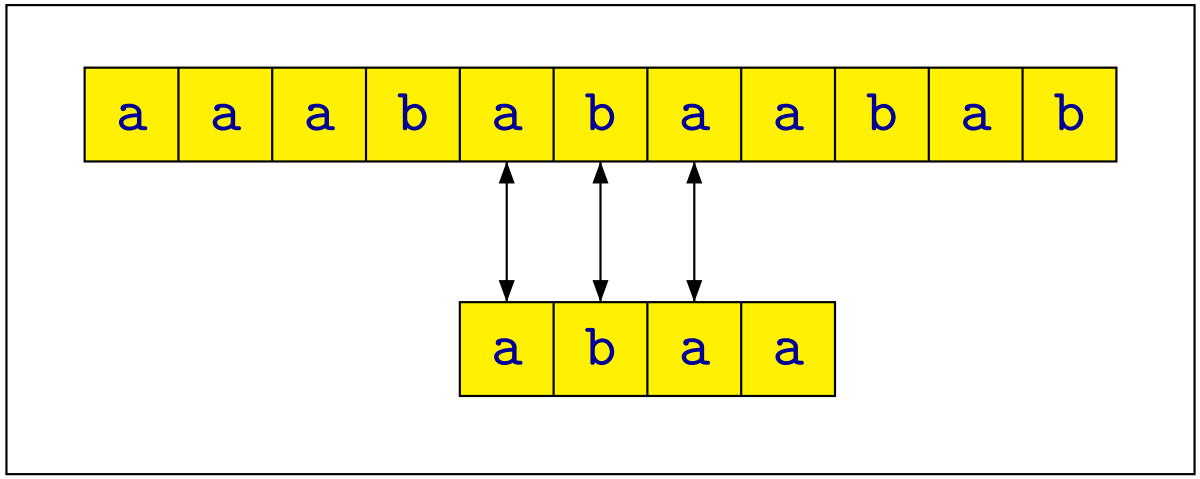
Vyhledávání v textu



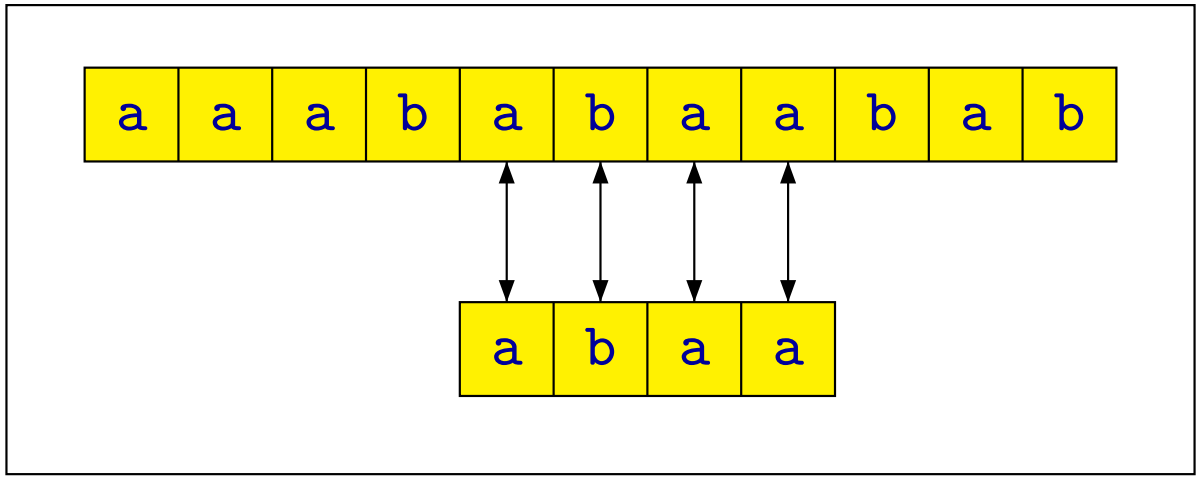
Vyhledávání v textu



Vyhledávání v textu



Vyhledávání v textu



Vyhledávání v textu

Pozorování: Nijak nevyužíváme informaci o části slova s, která souhlasí, a v dalším kroku začínáme zase od začátku slova s.

Nápad: Slovo t číst znak po znaku a pamatovat si jaká část slova s s koncem dosud načteného textu. Upravovat tento údaj vždy jen na základě dalšího jednoho načteného znaku:

LEPŠÍ-VYHLEDÁVÁNÍ(t, s)

```
1  n ← length(t)
2  m ← length(s)
3  q ← 0
4  for i ← 0 to n − 1
5        do q ← δ(q, t[i])
6           if q = m
7              then return TRUE
8  return FALSE
```

Prefixy, sufixy a podslova

V této souvislosti se nám hodí tři následující pojmy:

Definice

Slovo x je **prefixem** slova y, jestliže existuje slovo v takové, že $y = xv$.

Slovo x je **sufixem** slova y, jestliže existuje slovo u takové, že $y = ux$.

Slovo x je **podslovem** slova y, jestliže existují slova u a v taková, že $y = uxv$.

Příklad:

- Prefixy slova `abaab` jsou ε, `a`, `ab`, `aba`, `abaa`, `abaab`.
- Sufixy slova `abaab` jsou ε, `b`, `ab`, `aab`, `baab`, `abaab`.
- Podslova slova `abaab` jsou ε, `a`, `b`, `ab`, `ba`, `ab`, `aba`, `baa`, `abb`, `abaa`, `baab`, `abaab`.

Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

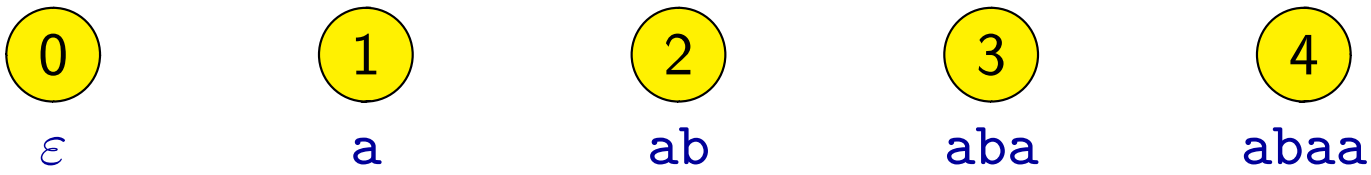
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:



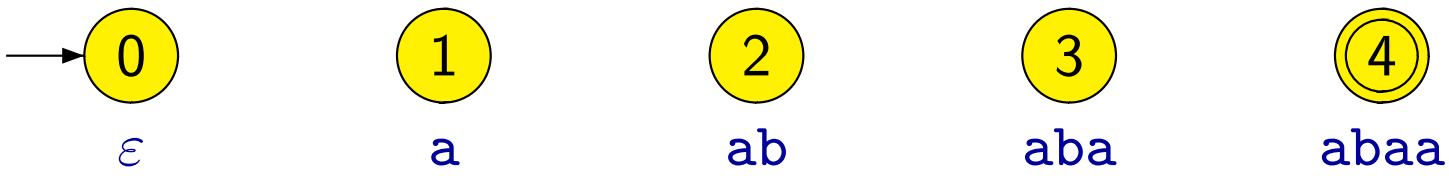
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:



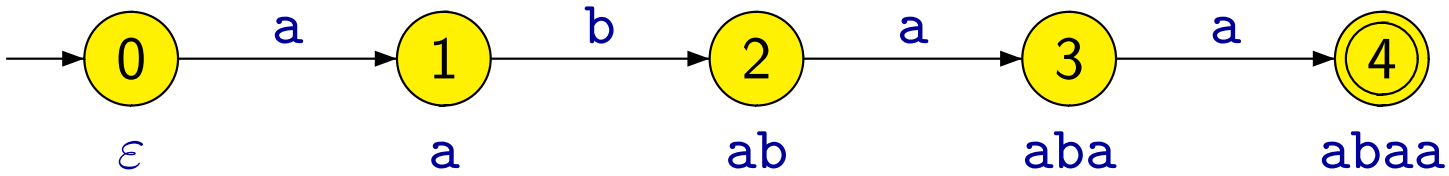
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:



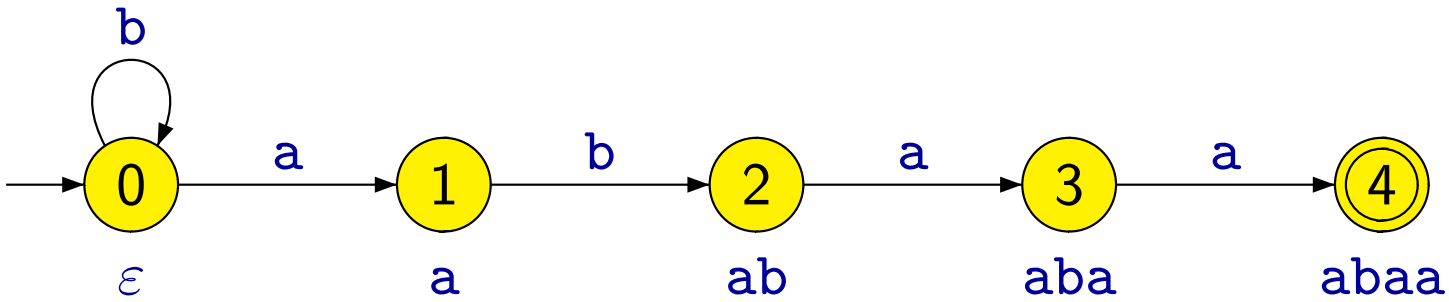
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:



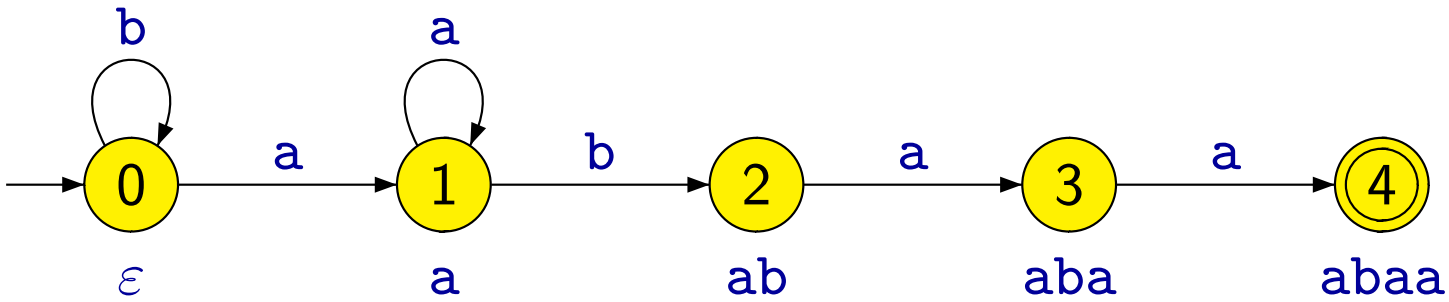
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:



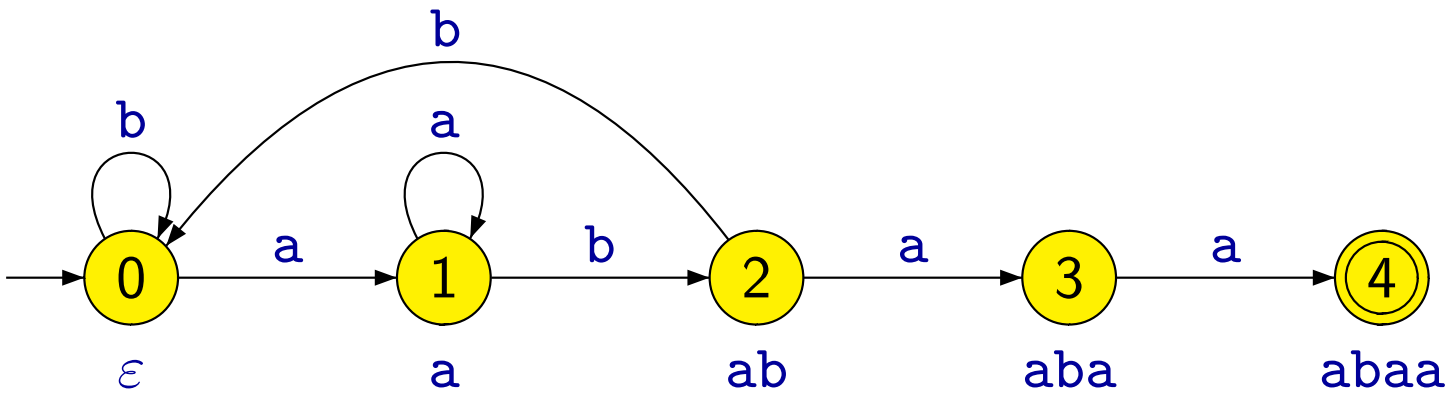
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:



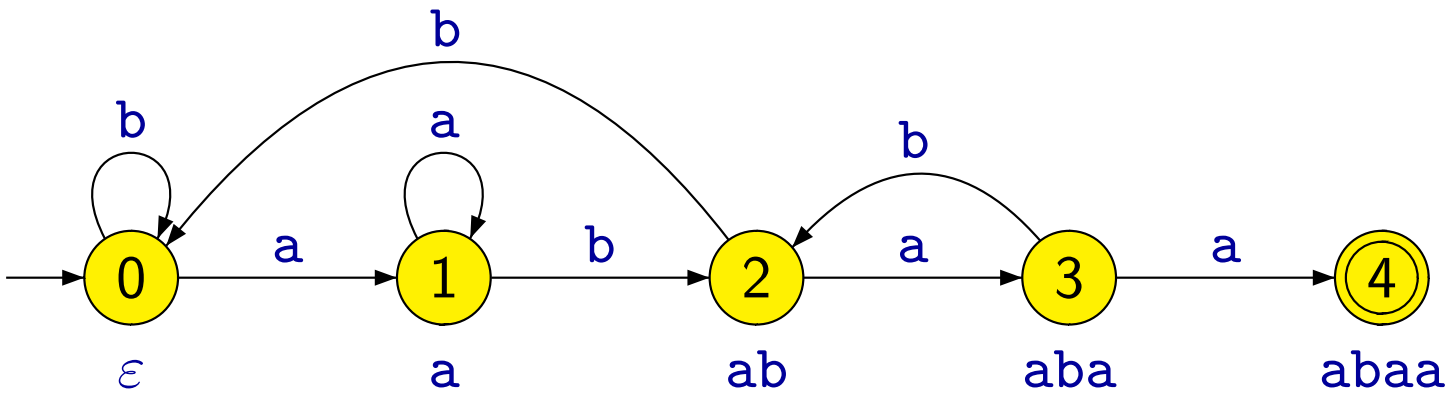
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:



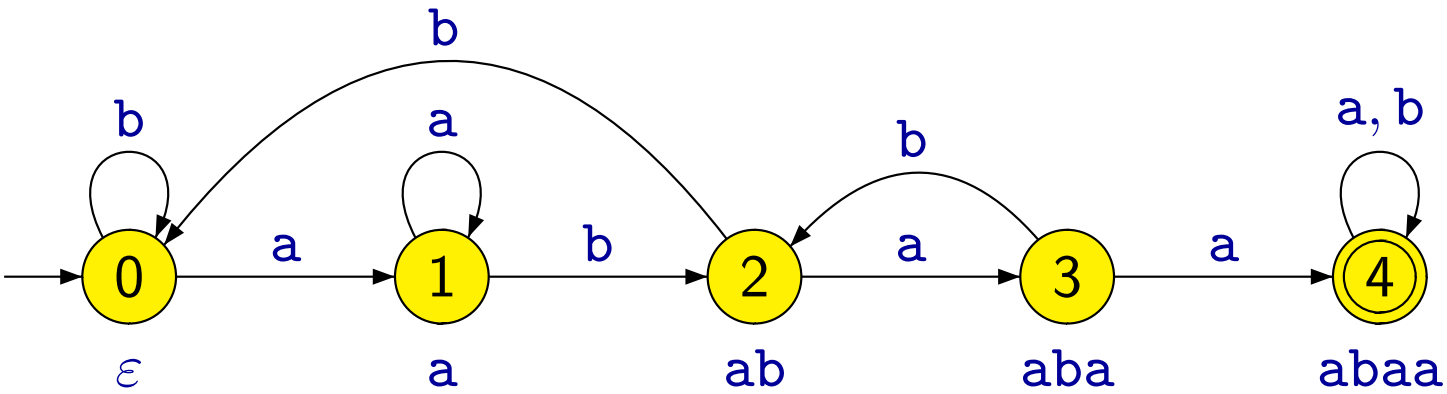
Vyhledávání v textu

Hodnota q, kterou si během výpočtu pamatujeme je tedy délka prefixu slova s, který současně sufixem dosud přečtené části slova t.

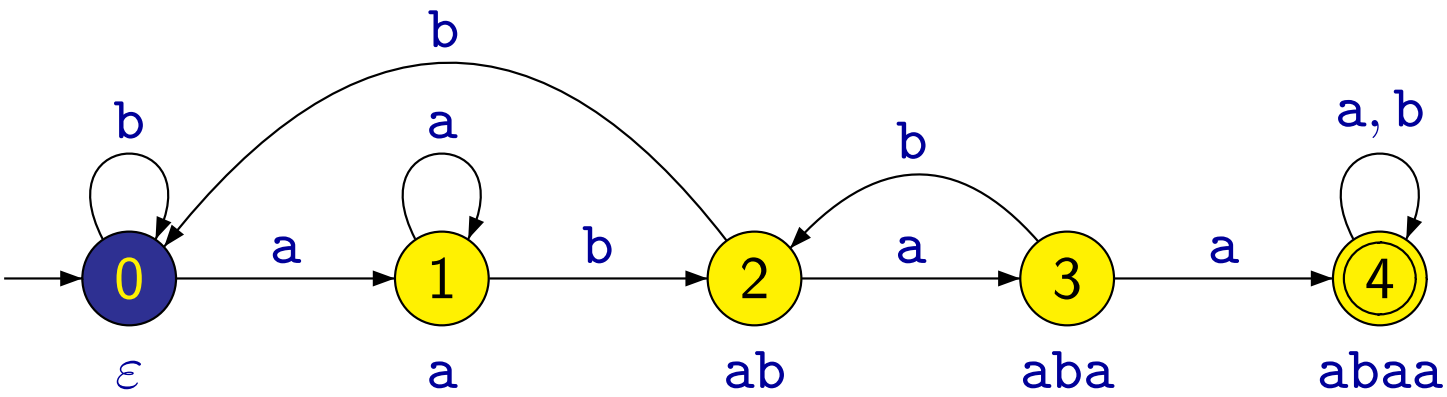
Poznámka: Pokud je takových prefixů víc, pamatujeme si délku nejdelšího z nich.

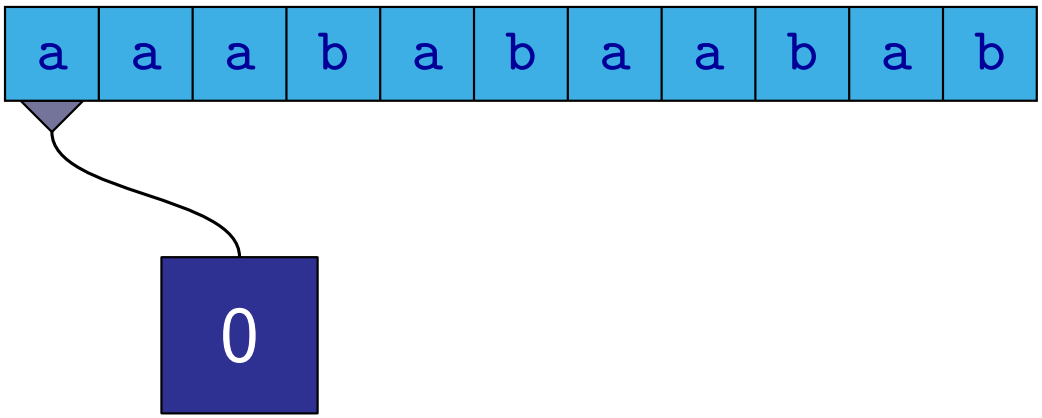
Proměnná zjevně může nabývat jen hodnot $\{0, 1, \ldots, m\}$. Hodnoty m nabývá jen v případě, že bylo nalezeno slovo s.

Chování tohoto systému si opět můžeme znázornit jako graf:

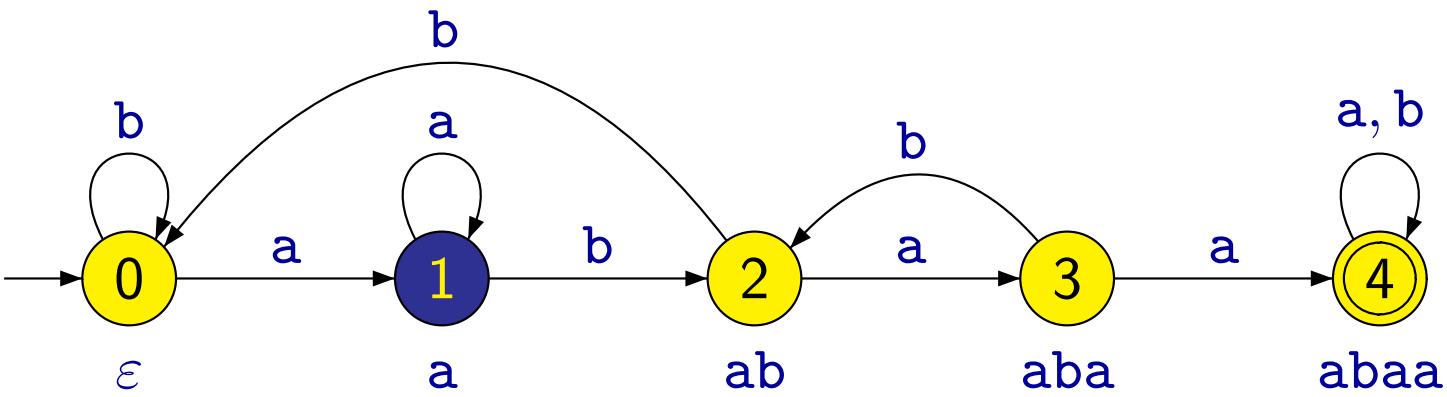


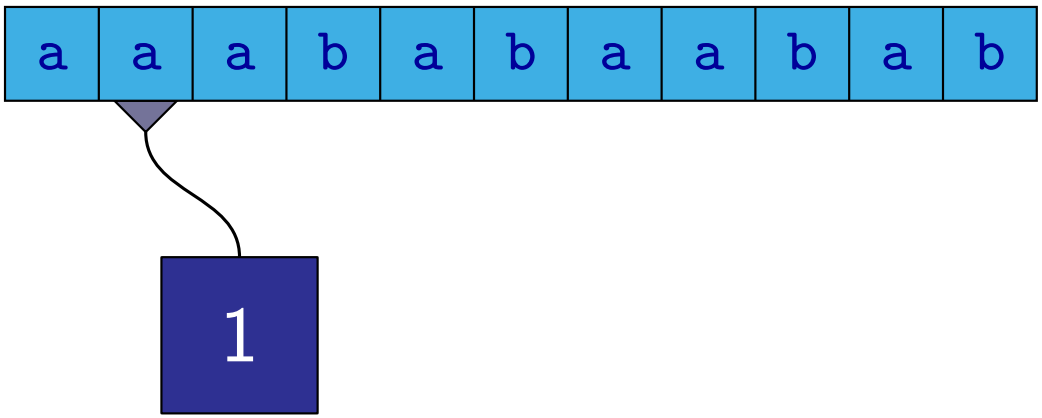
Vyhledávání v textu



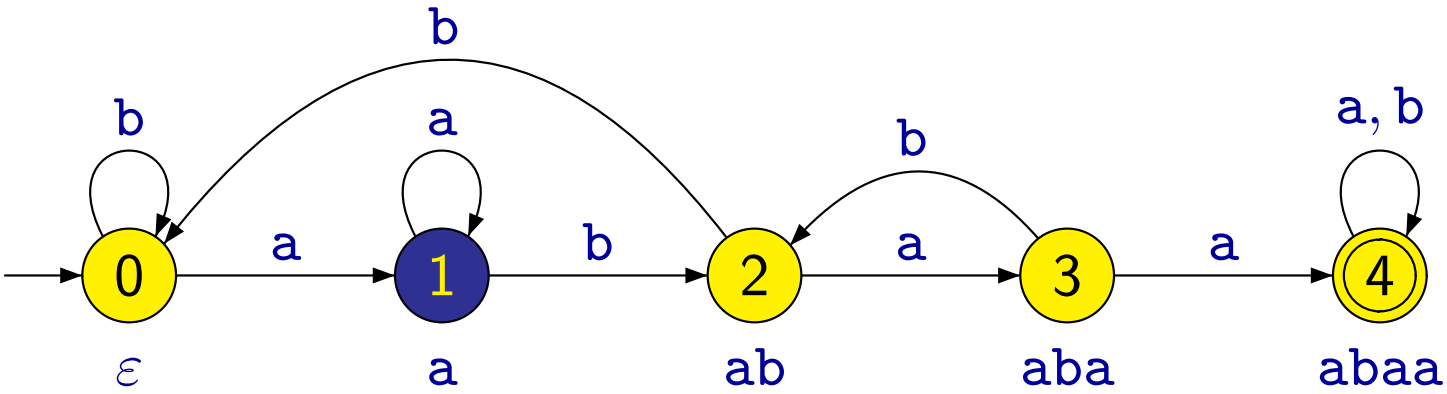


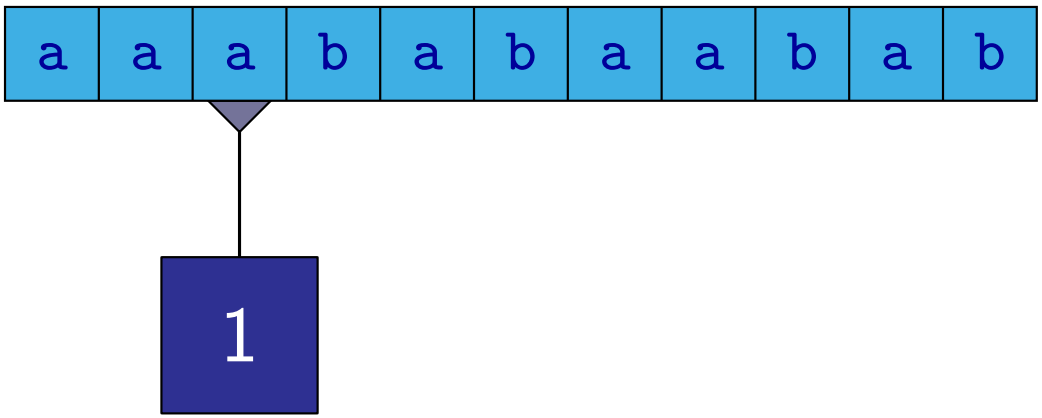
Vyhledávání v textu



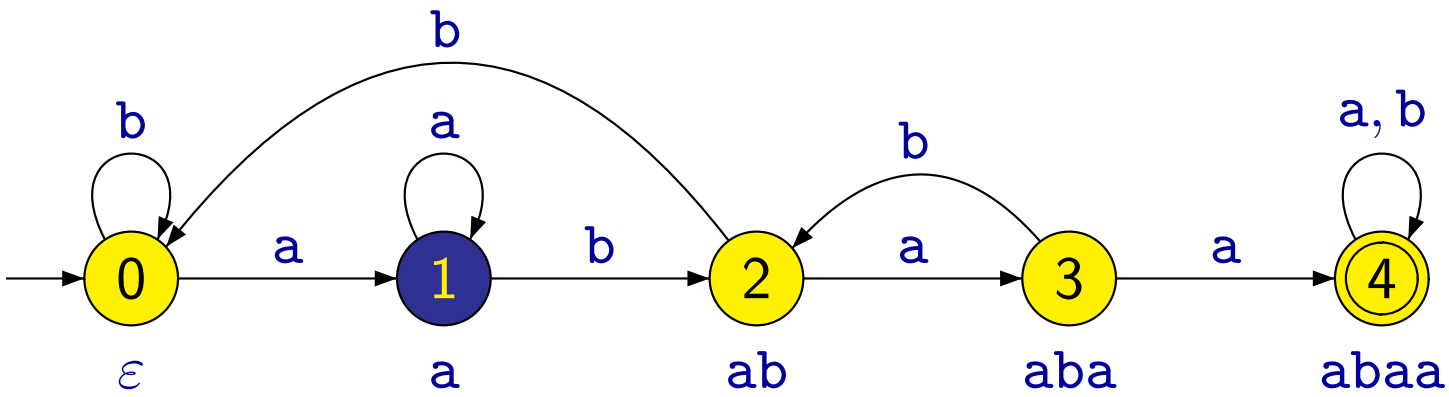


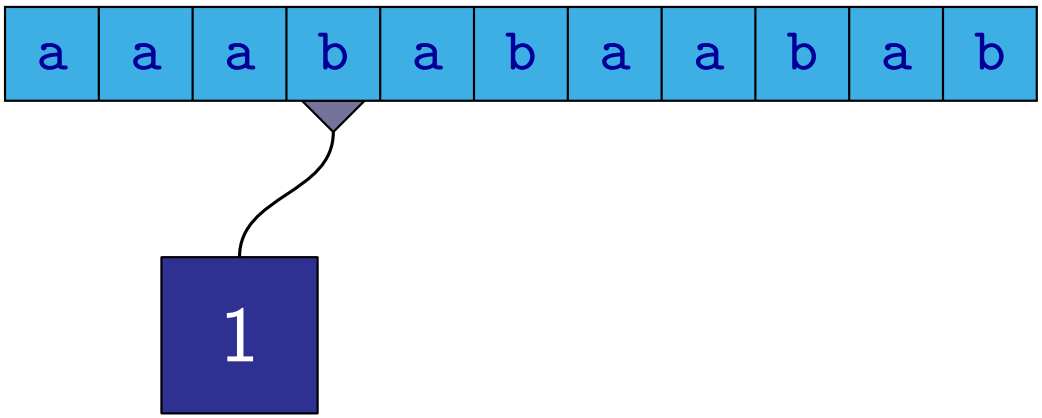
Vyhledávání v textu



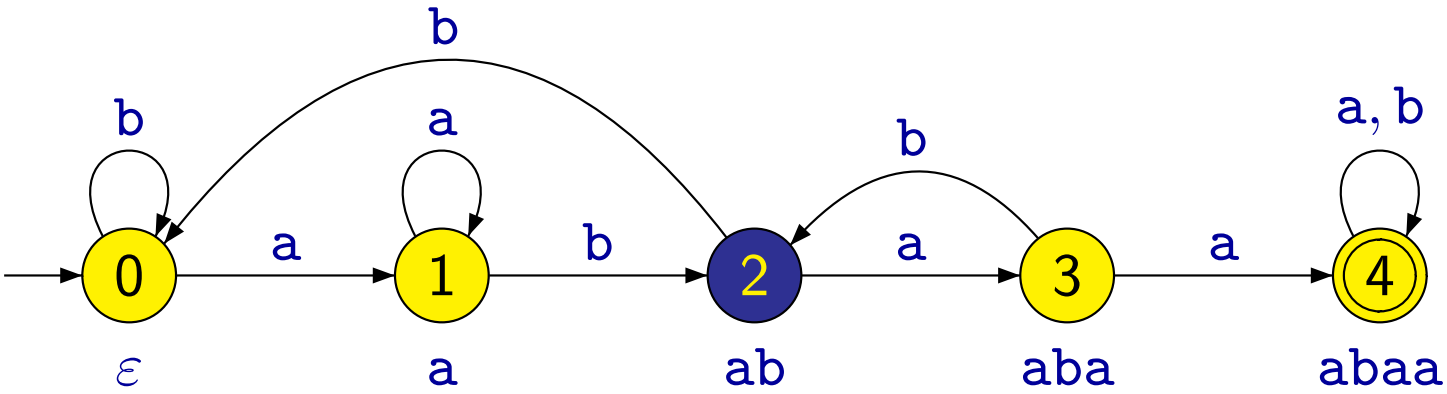


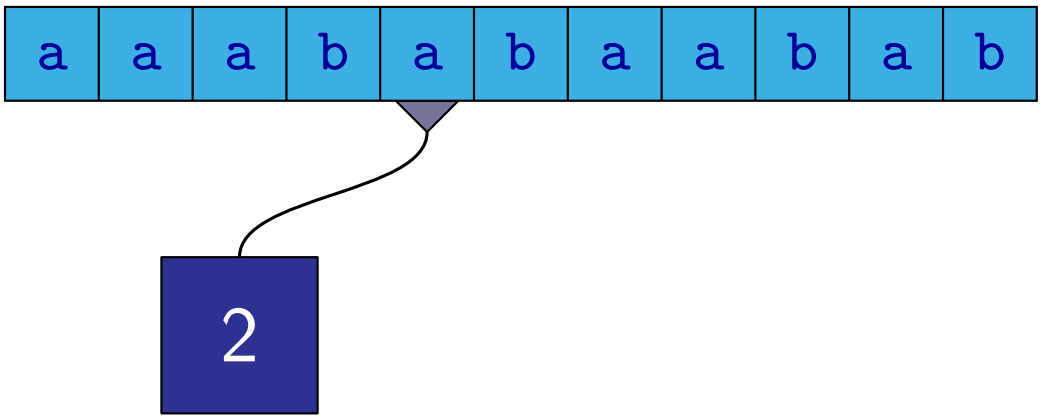
Vyhledávání v textu



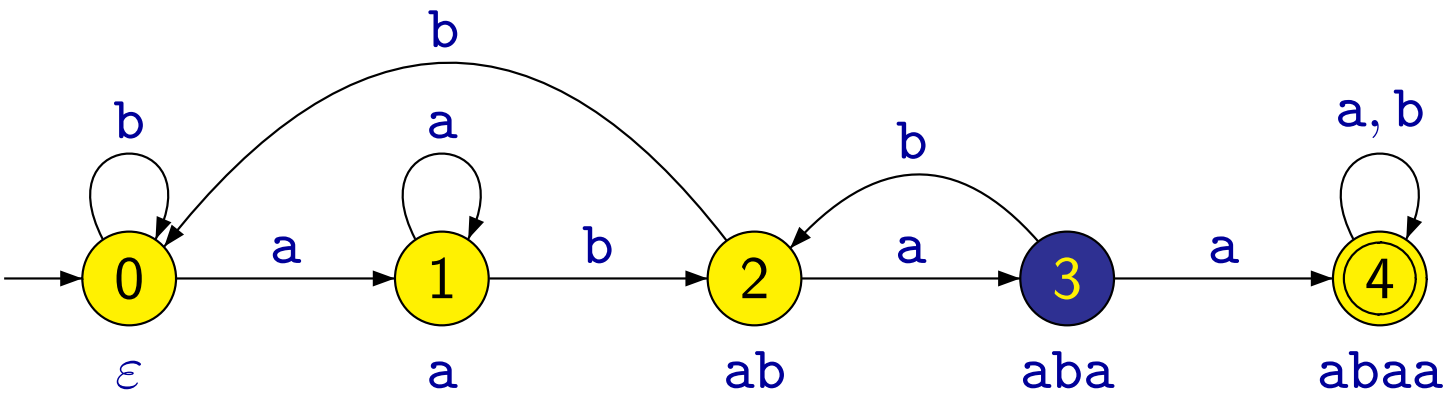


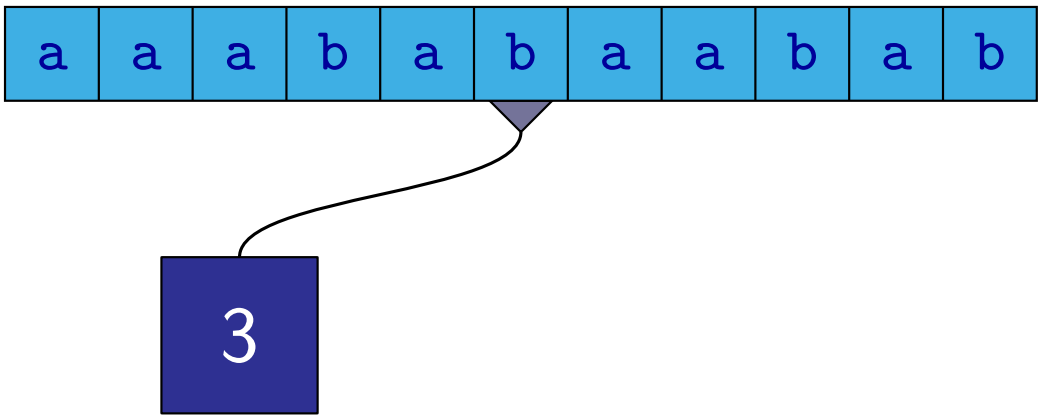
Vyhledávání v textu



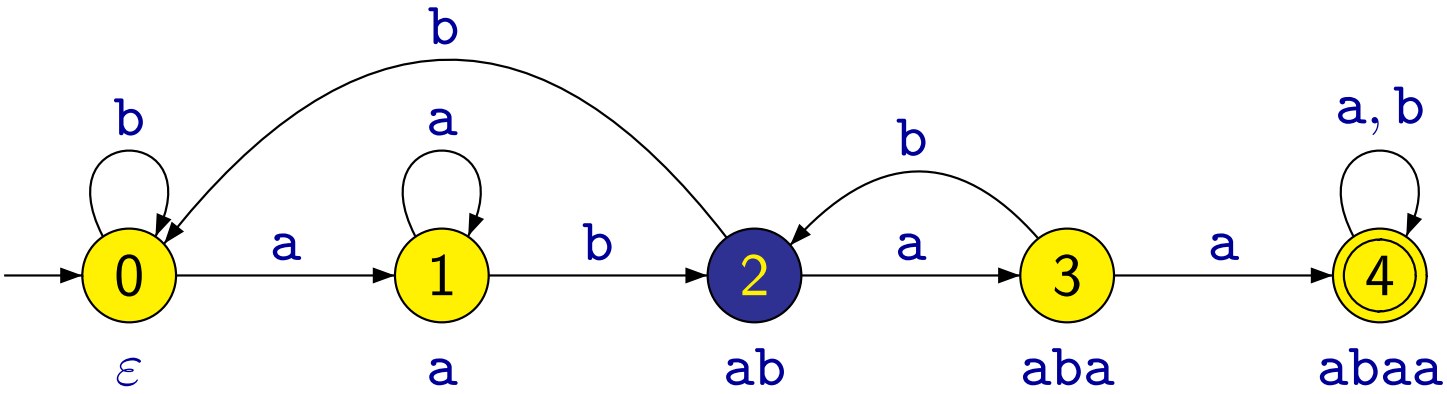


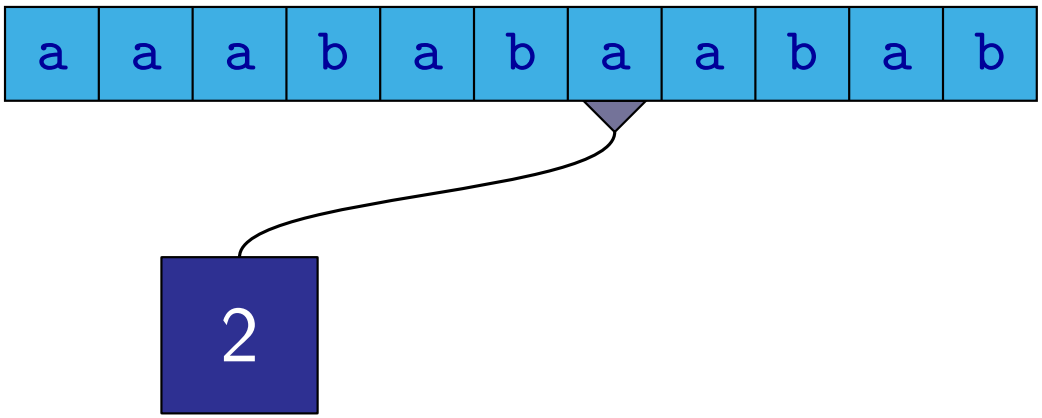
Vyhledávání v textu



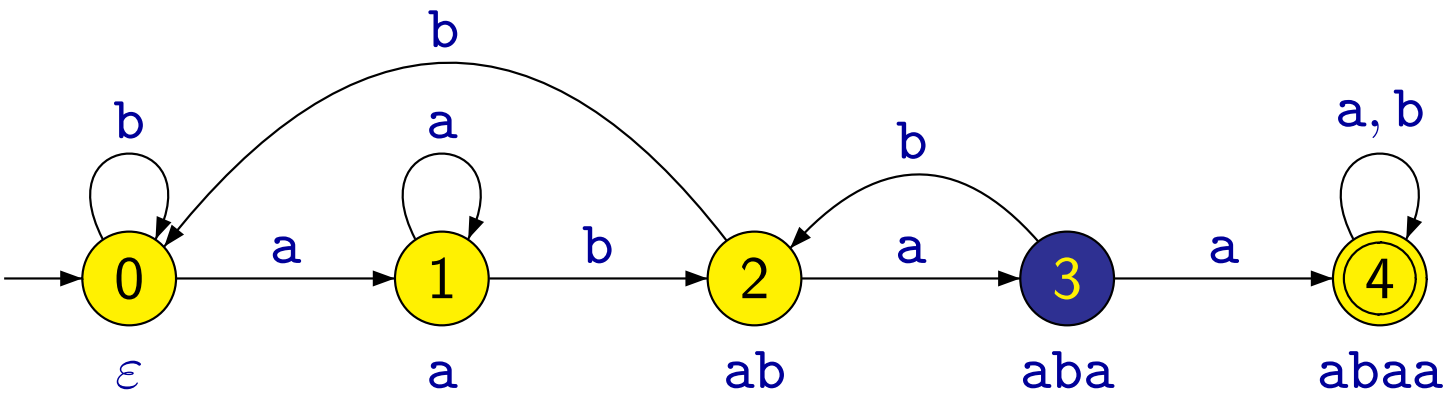


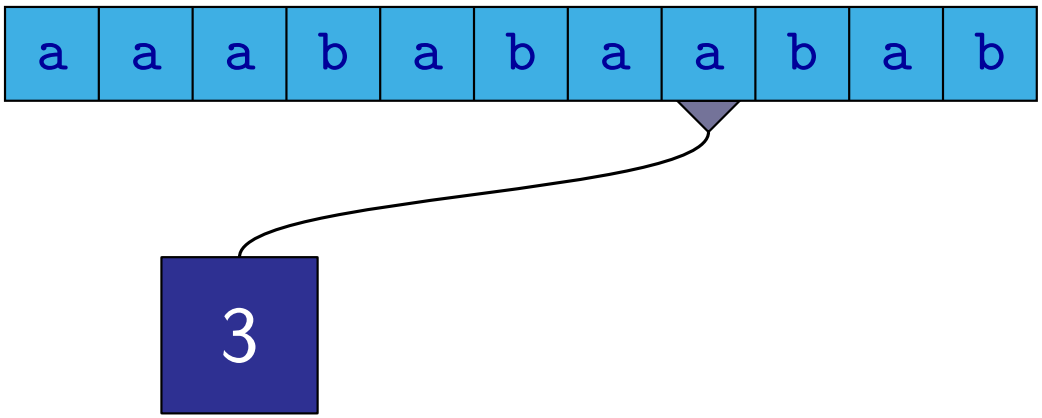
Vyhledávání v textu



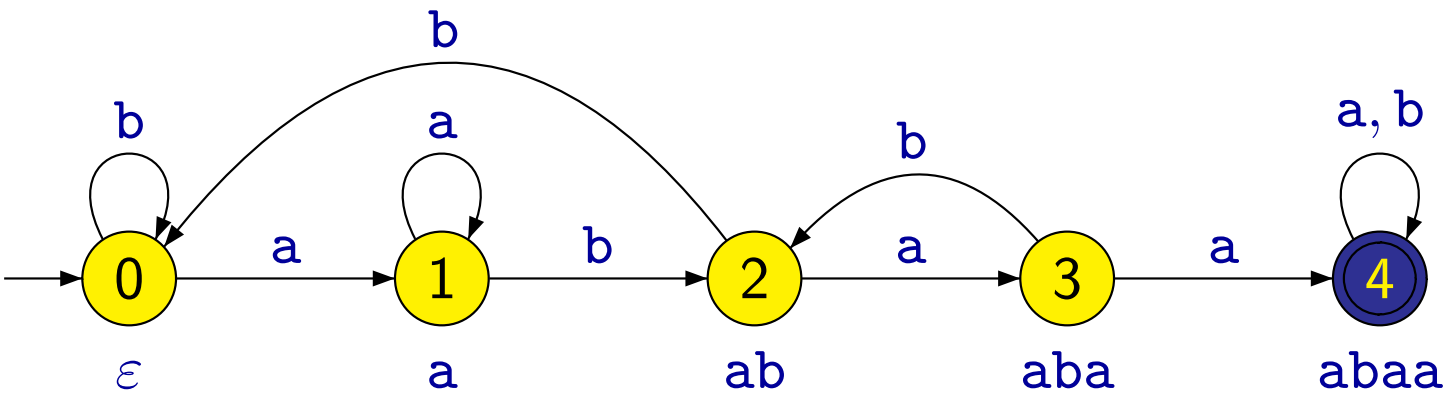


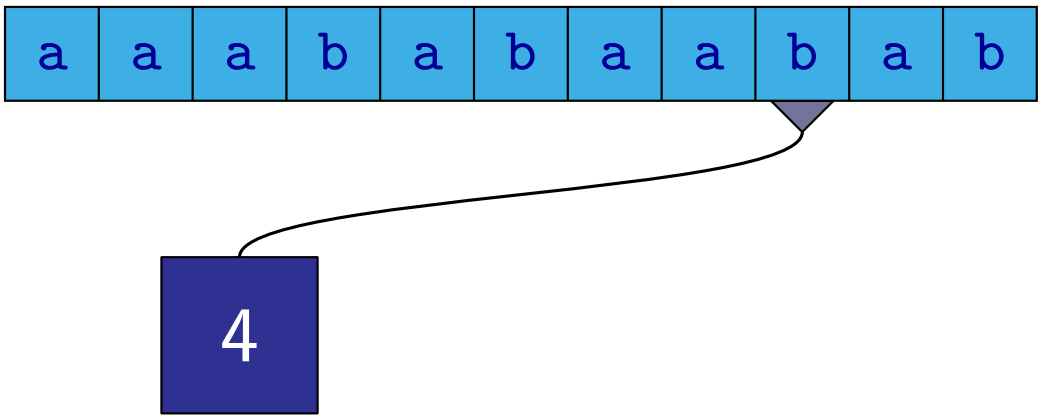
Vyhledávání v textu



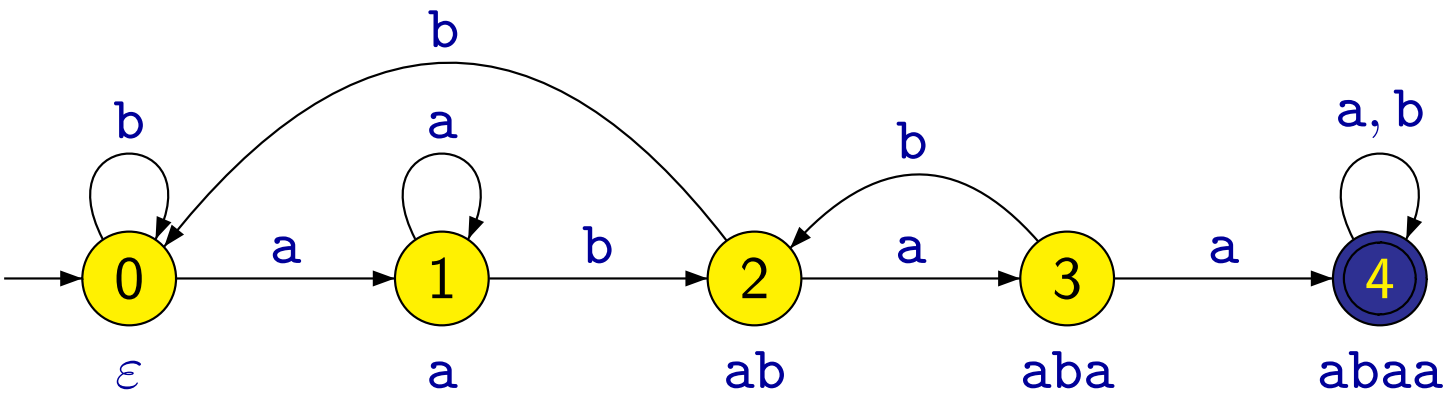


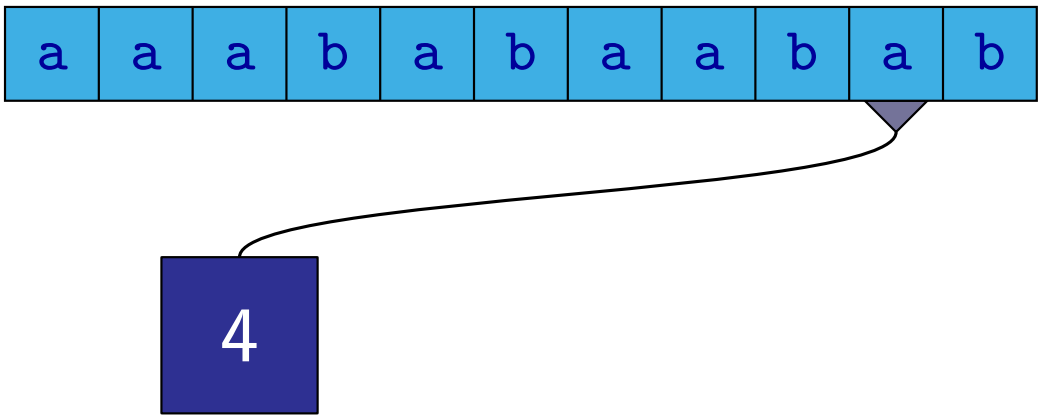
Vyhledávání v textu



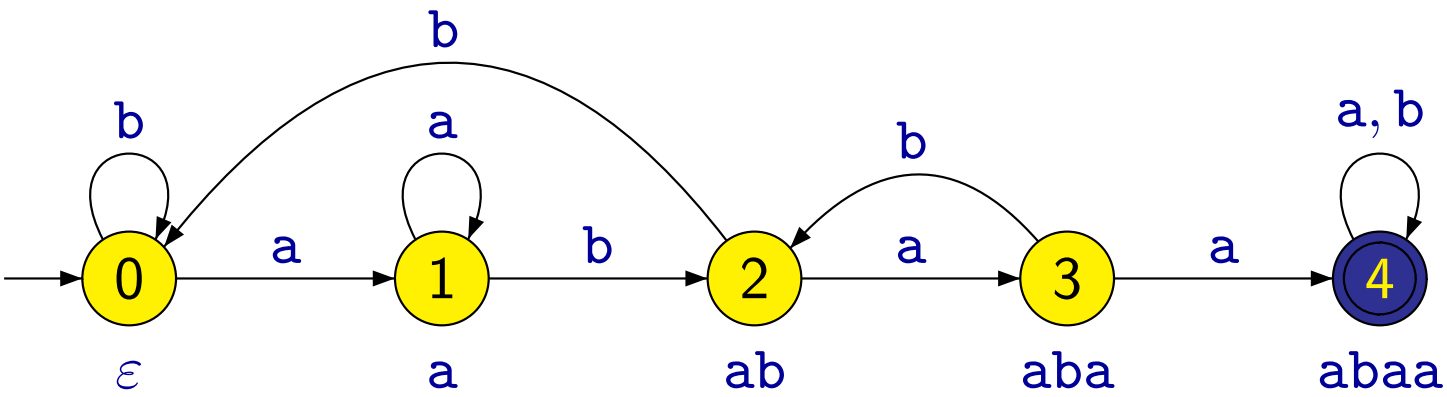


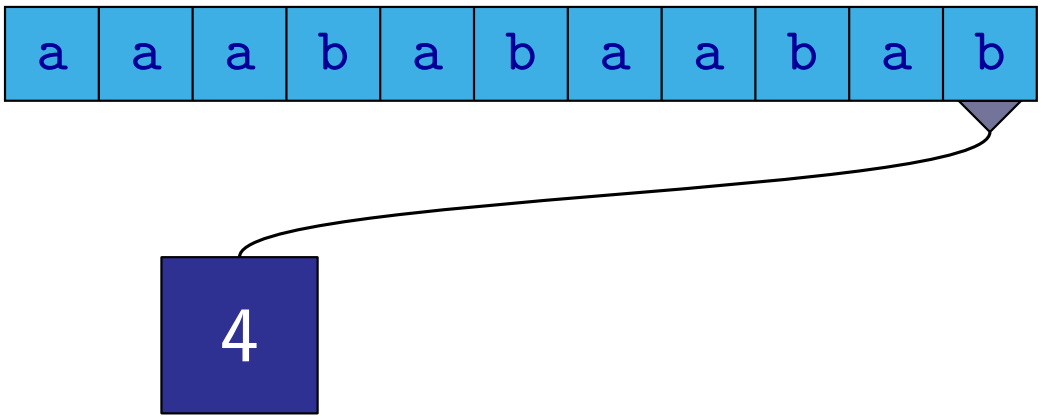
Vyhledávání v textu



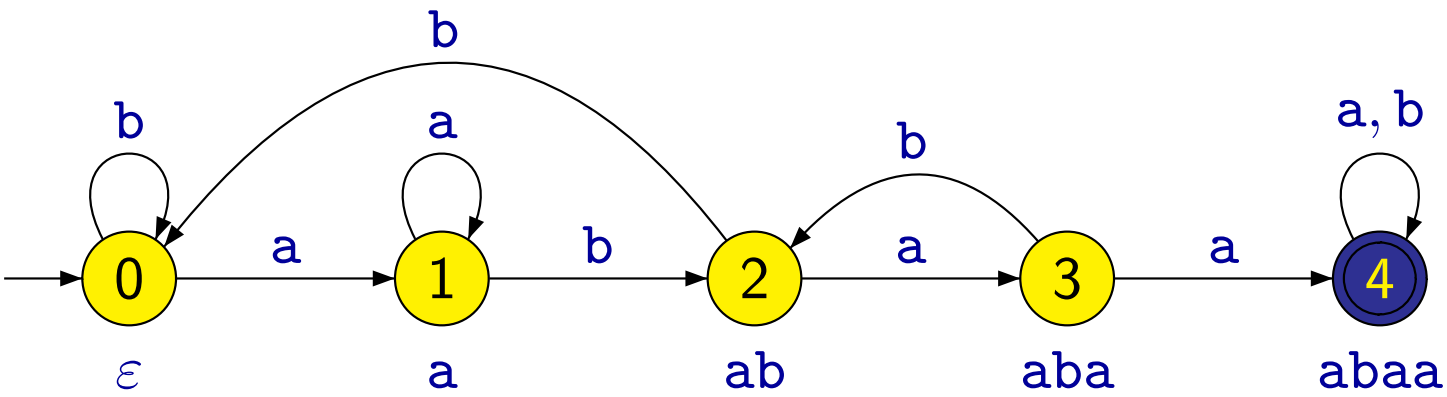


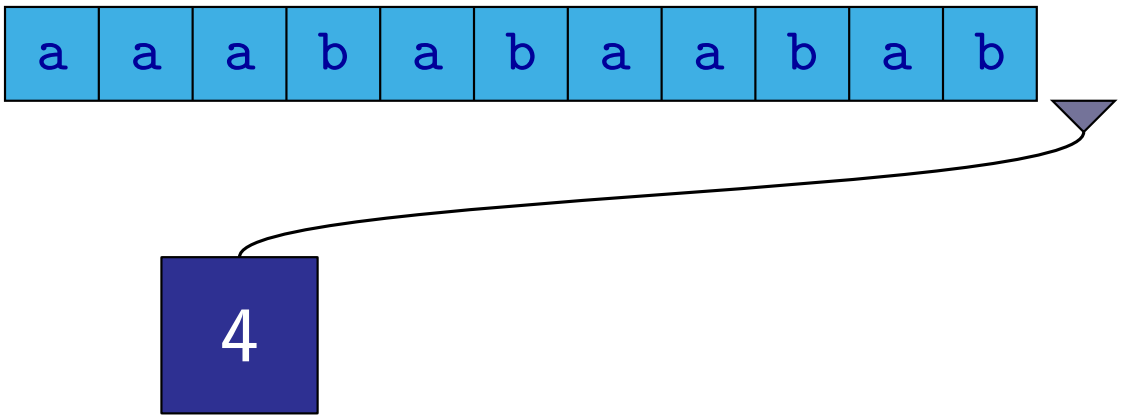
Vyhledávání v textu



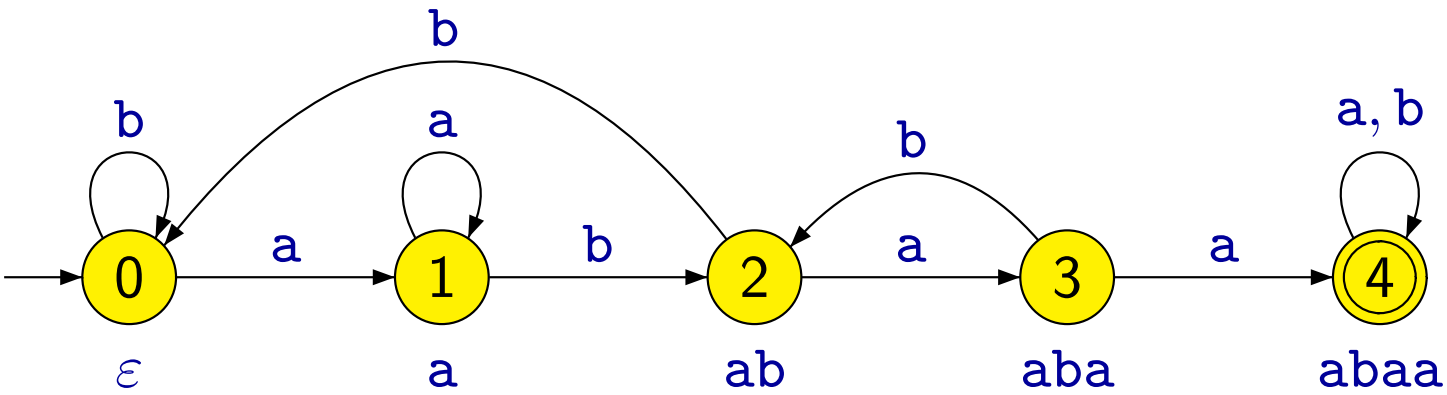


Vyhledávání v textu





Vyhledávání v textu



Místo grafu můžeme stejnou informaci reprezentovat tabulkou:

	a	b
→ 0	1	0
1	1	2
2	3	0
3	4	2
← 4	4	4

Vyhledávání v textu

Jak zvolit pro danou dvojici q a t_i novou hodnotu q, tj. hodnotu $\delta(q, t_i)$?

Vyhledávání v textu

Jak zvolit pro danou dvojici q a t_i novou hodnotu q, tj. hodnotu $\delta(q, t_i)$?

Zvolíme $\delta(q, t_i) = q'$ takové, že slovo $s_0 s_1 \cdots s_{q'-1}$ je nejdelším prefixem slova s takovým, že je současně suffixem slova $s_0 s_1 \cdots s_{q-1} t_i$.

Poznámka: Předpokládáme, že $s = s_0 s_1 \cdots s_{m-1}$.

Vyhledávání v textu

Jak zvolit pro danou dvojici q a t_i novou hodnotu q, tj. hodnotu $\delta(q, t_i)$?

Zvolíme $\delta(q, t_i) = q'$ takové, že slovo $s_0 s_1 \cdots s_{q'-1}$ je nejdelším prefixem slova s takovým, že je současně suffixem slova $s_0 s_1 \cdots s_{q-1} t_i$.

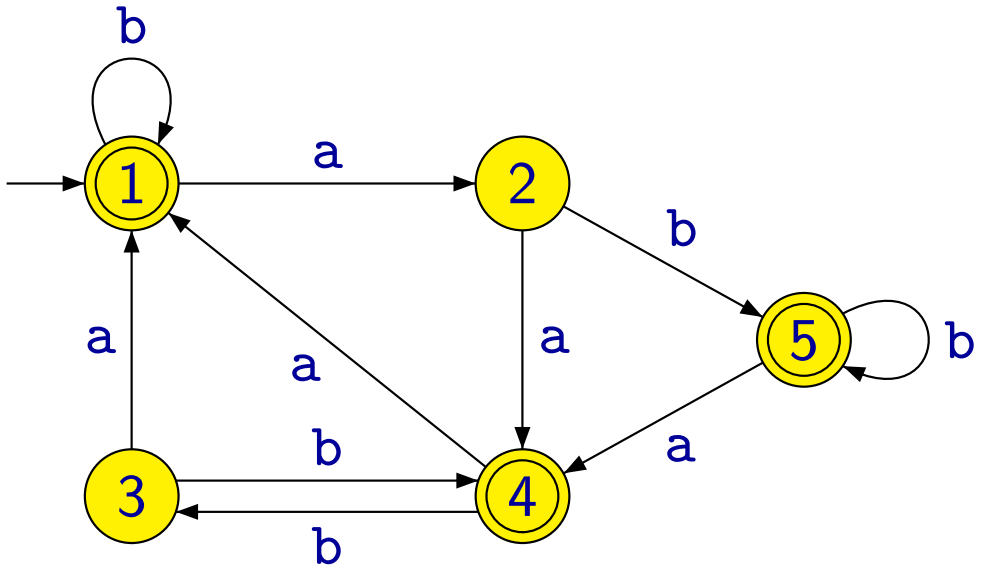
Poznámka: Předpokládáme, že $s = s_0 s_1 \cdots s_{m-1}$.

Poznámka: Výše uvedené úvahy vedou k algoritmu nazývanému podle jeho autorů Knuth-Morris-Pratt.

V tomto algoritmu se vyhneme tomu, že bychom výše uvedenou tabulku skutečně sestrojili. Místo ní se sestrojí určitá její stručnější reprezentace, která ale umožňuje hodnoty z tabulky rychle vypočítat.

Zde se ale nebudeme tímto algoritmem blíže zabývat.

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

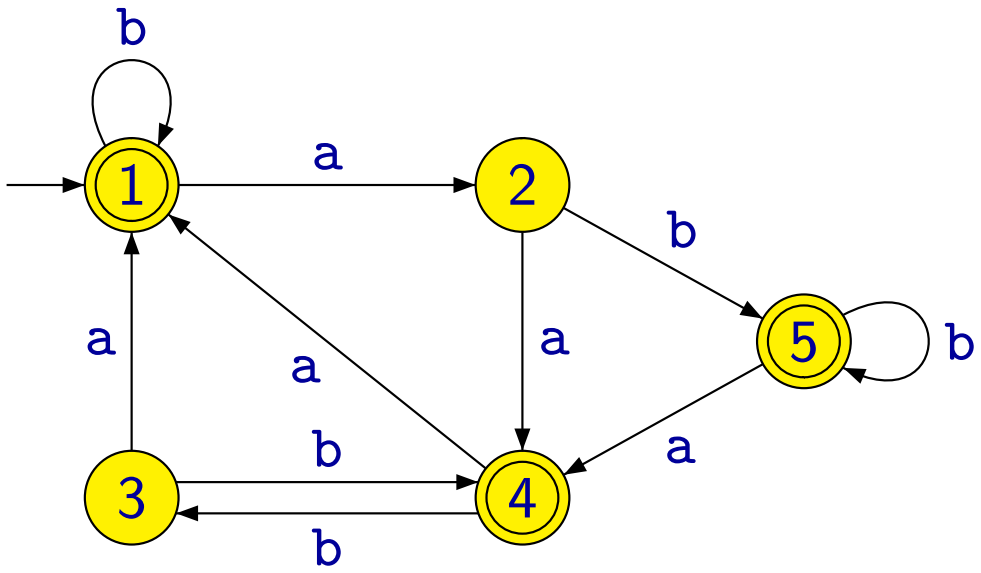
Formálně je **deterministický konečný automat** definován jako pětice

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

kde:

- Q je konečná množina **stavů**
- Σ je konečná **abeceda**
- $\delta : Q \times \Sigma \to Q$ 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

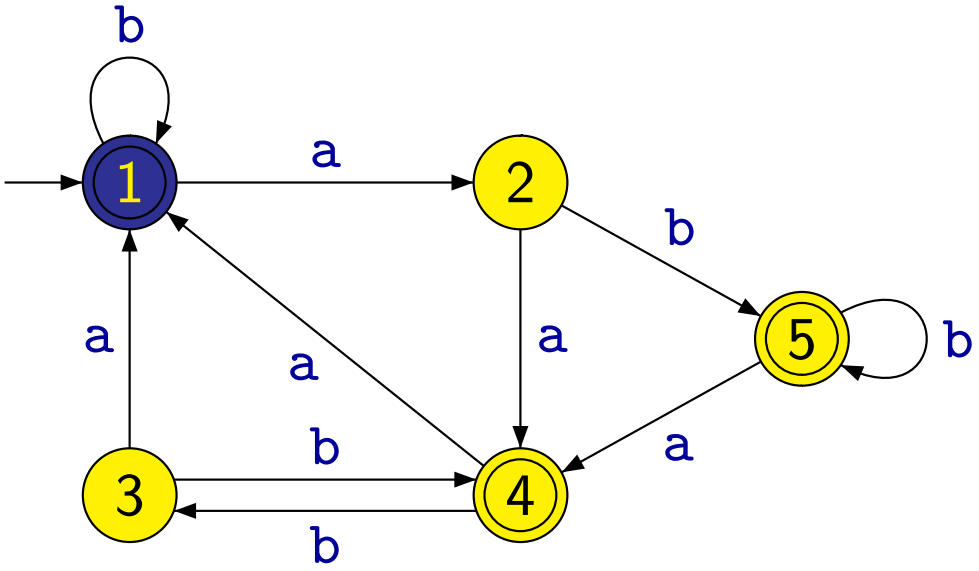
Místo zápisu

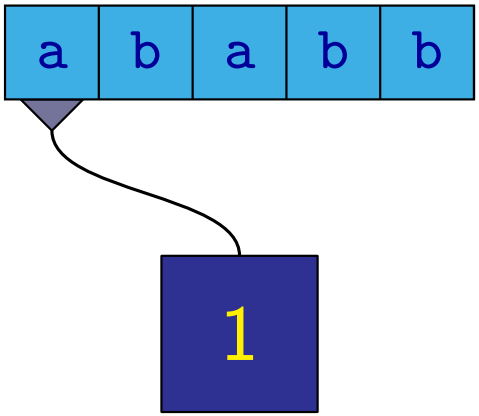
$$\begin{array}{ll} \delta(1,\mathtt{a}) = 2 & \delta(1,\mathtt{b}) = 1 \\ \delta(2,\mathtt{a}) = 4 & \delta(2,\mathtt{b}) = 5 \\ \delta(3,\mathtt{a}) = 1 & \delta(3,\mathtt{b}) = 4 \\ \delta(4,\mathtt{a}) = 1 & \delta(4,\mathtt{b}) = 3 \\ \delta(5,\mathtt{a}) = 4 & \delta(5,\mathtt{b}) = 5 \end{array}$$

budeme raději používat stručnější tabulku nebo grafické znázornění:

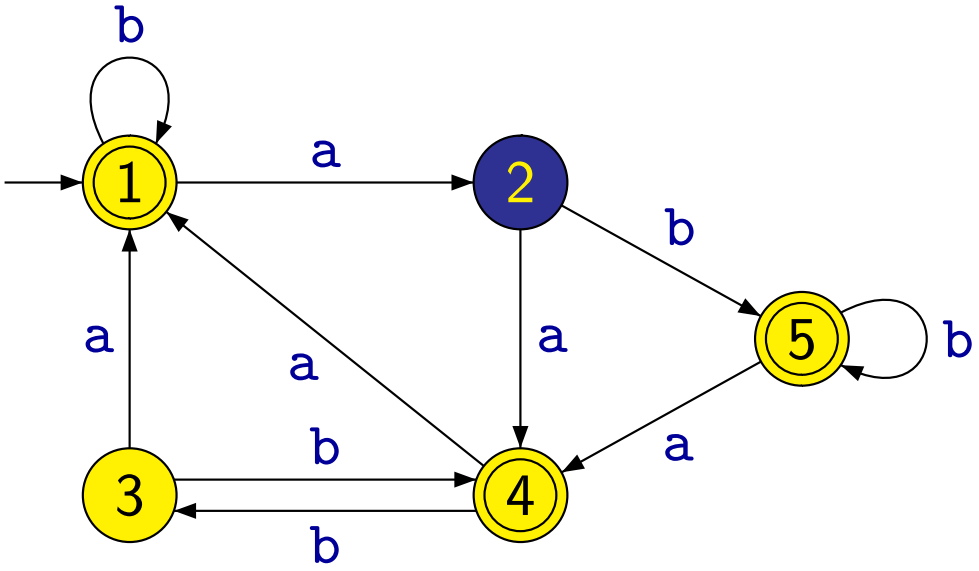
δ	a	b
$\leftrightarrow 1$	2	1
2	4	5
3	1	4
$\leftarrow 4$	1	3
$\leftarrow 5$	4	5

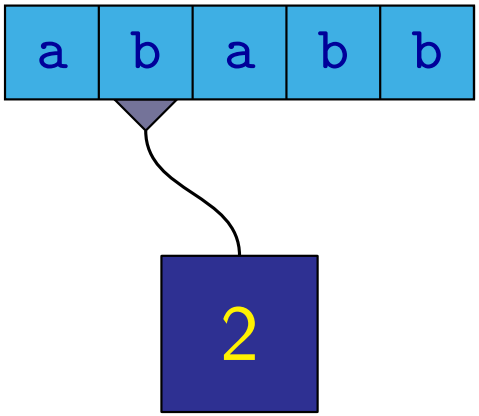
Deterministický konečný automat



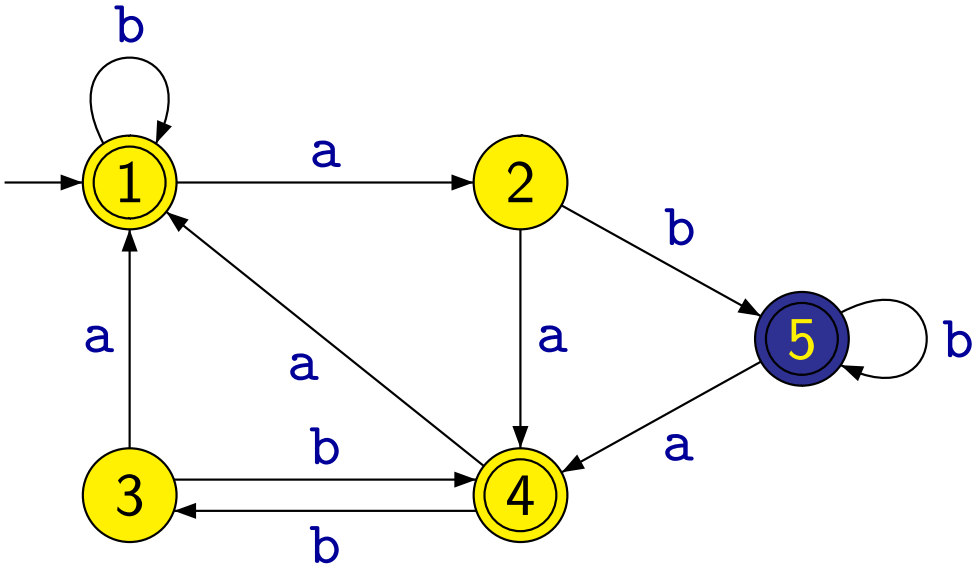


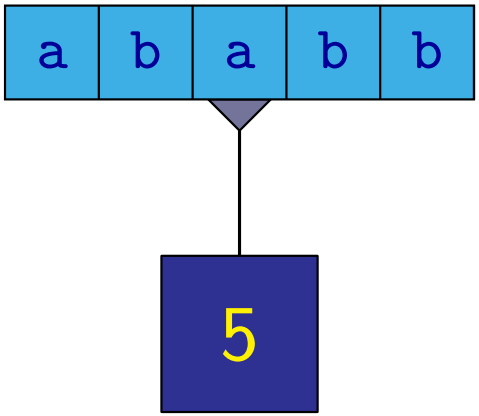
Deterministický konečný automat



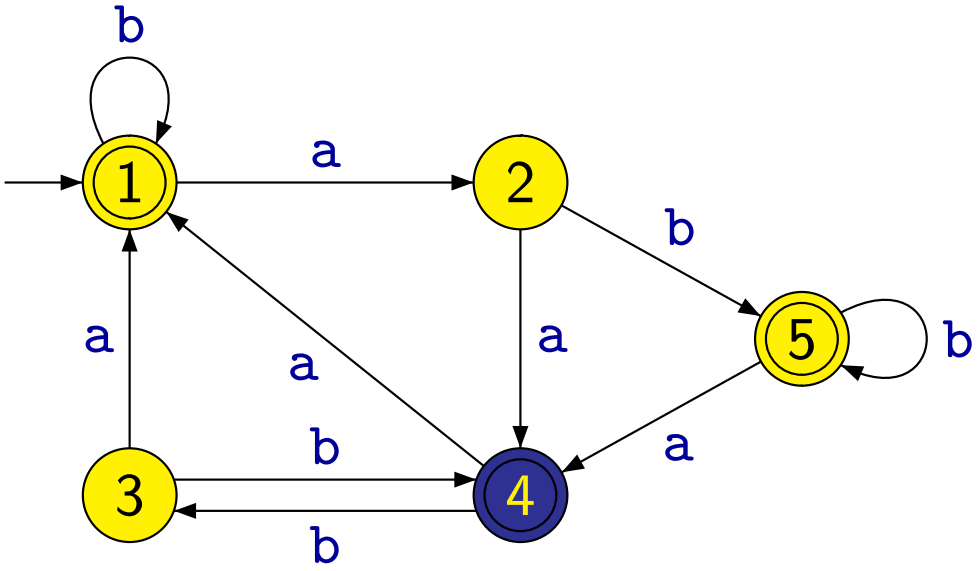


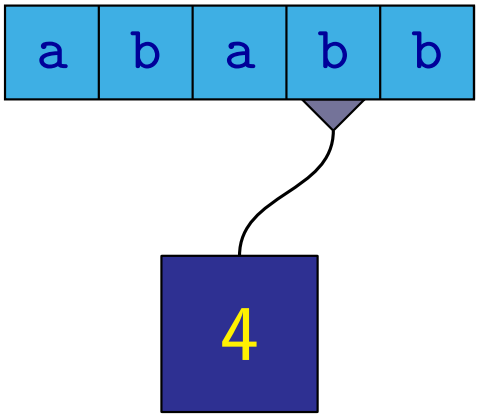
Deterministický konečný automat



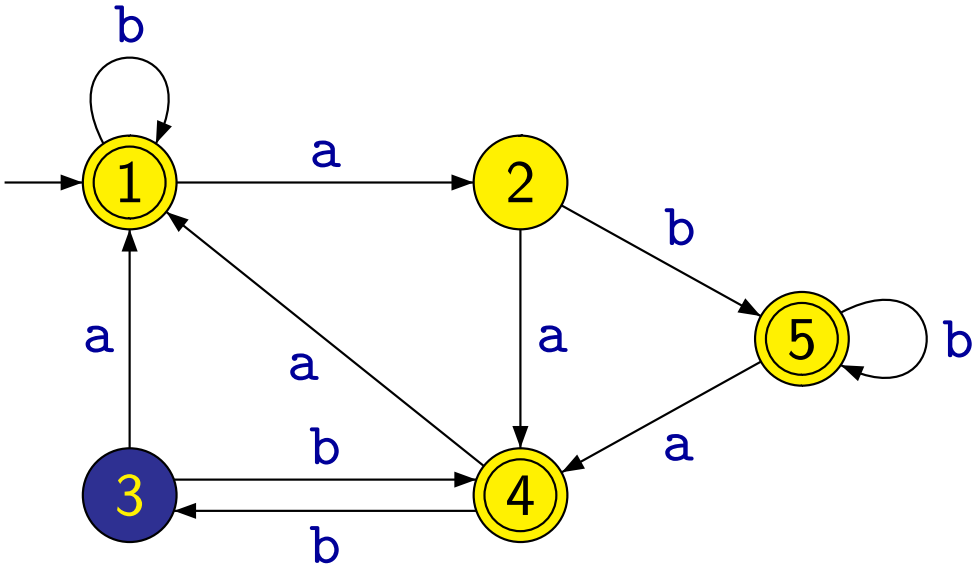


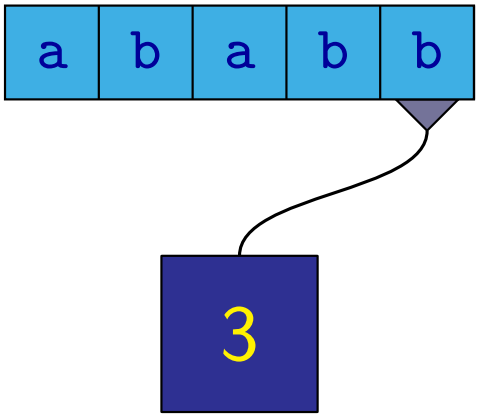
Deterministický konečný automat



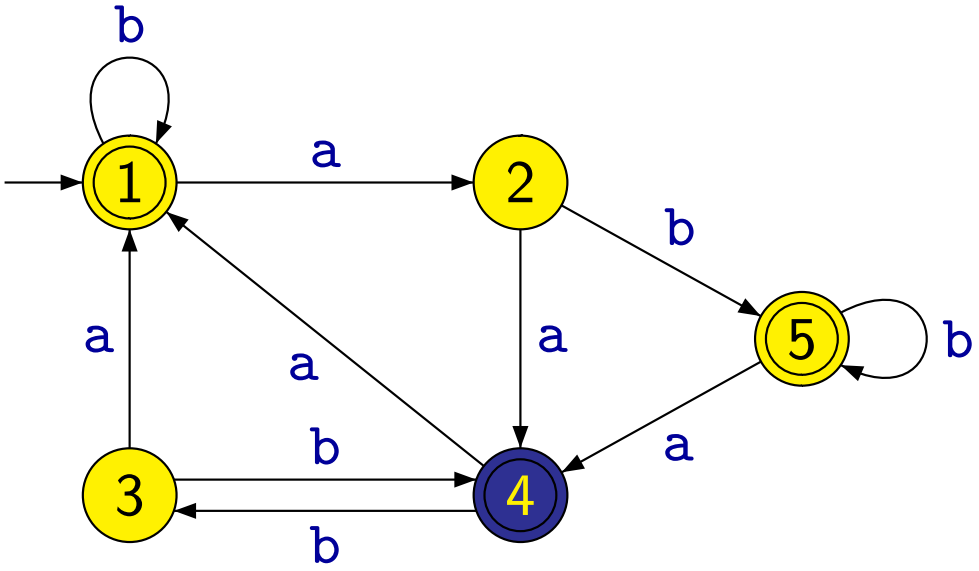


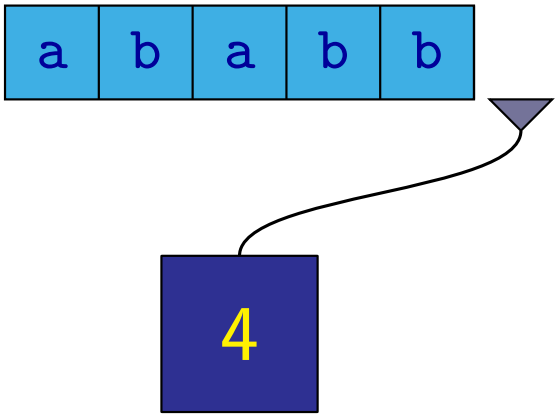
Deterministický konečný automat





Deterministický konečný automat





Deterministický konečný automat

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

Formálně můžeme konfiguraci definovat jako dvojici z množiny $Q \times \Sigma^*$.

Příklad: $(2, \mathtt{babb})$ je konfigurace

Na množině všech konfigurací můžeme definovat binární relaci $\vdash$ s následujícím významem: $C_1 \vdash C_2$ znamená, že automat 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 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.

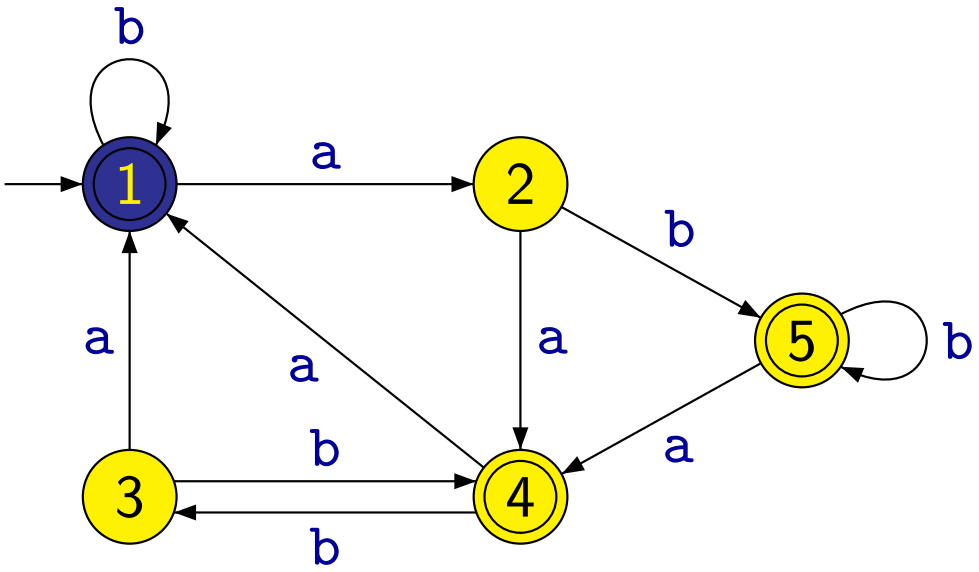
Definice

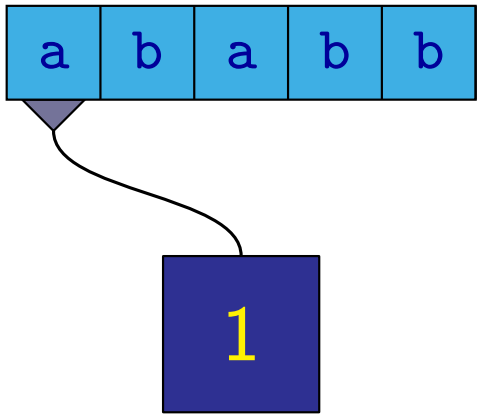
Výpočet automatu je posloupnost konfigurací

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

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 $C_{i-1} \vdash C_i$.

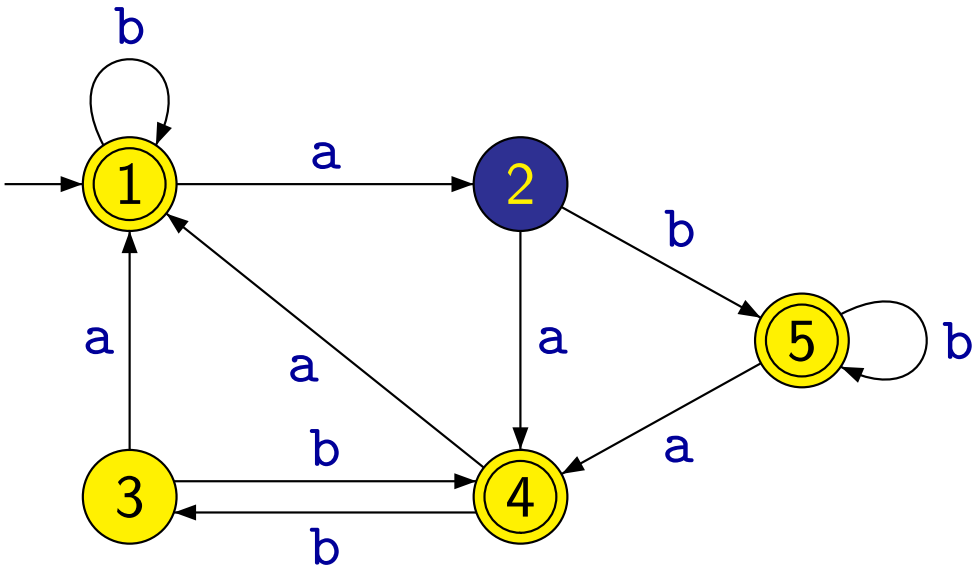
Deterministický konečný automat

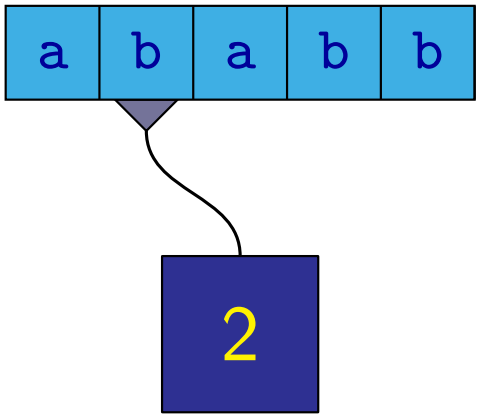




$(1, \mathtt{ababb})$

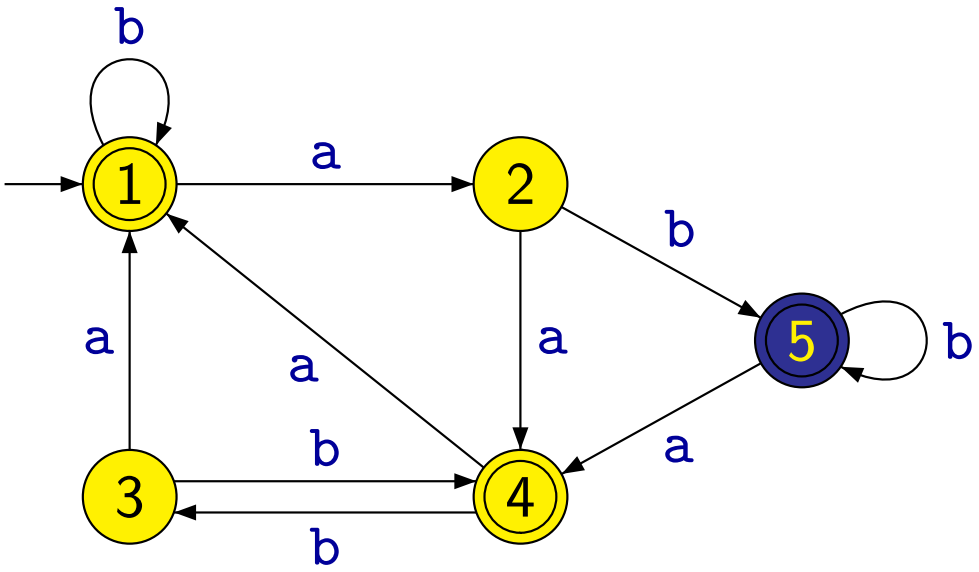
Deterministický konečný automat

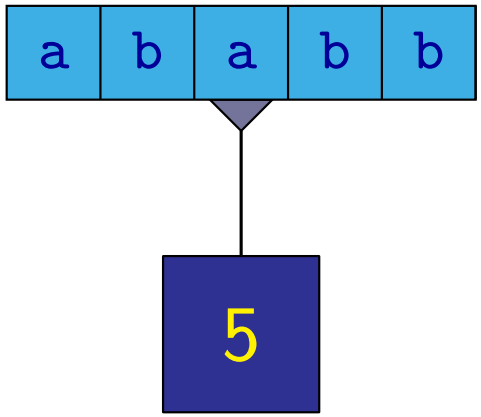




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb})$

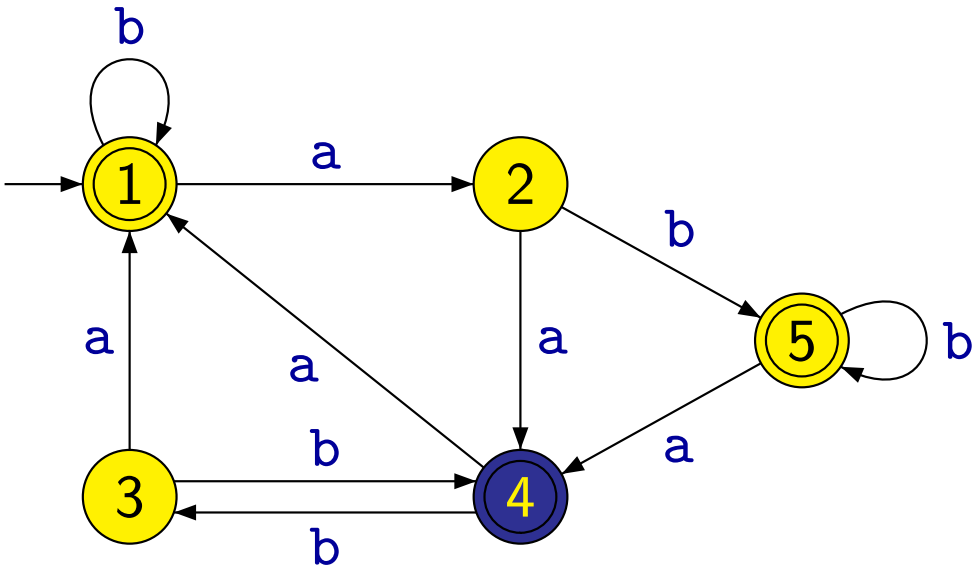
Deterministický konečný automat

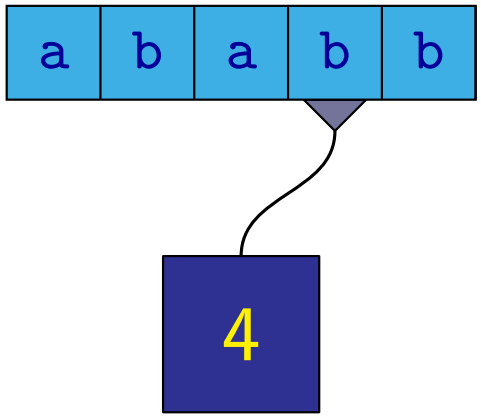




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash (5, \mathtt{abb})$

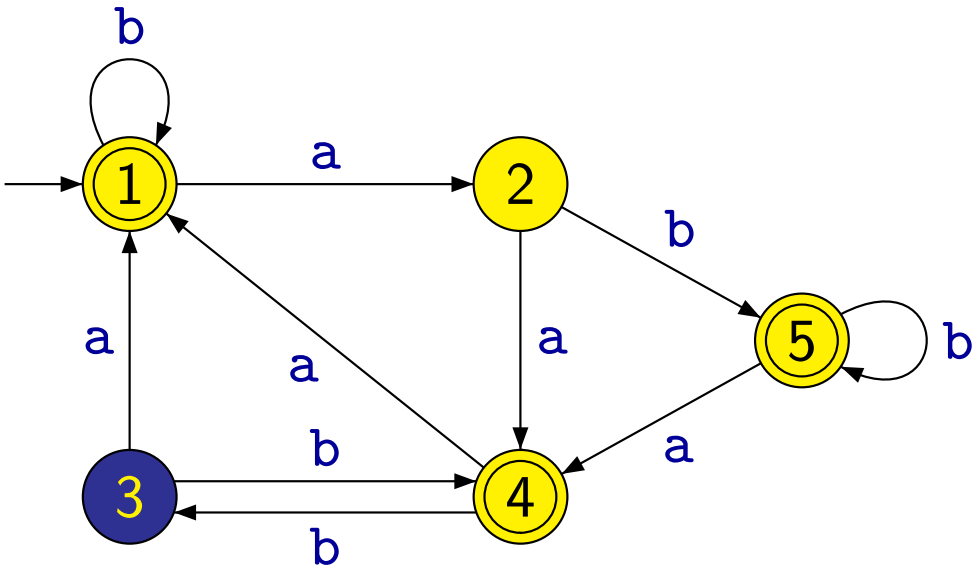
Deterministický konečný automat

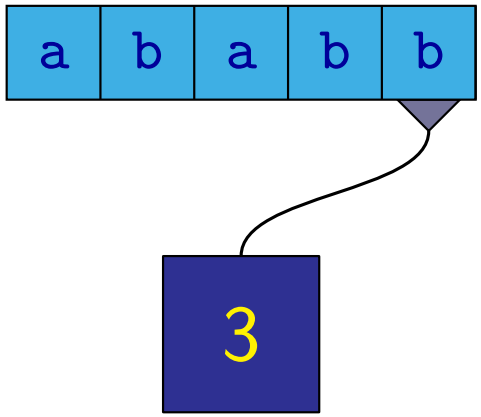




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash$
$(5, \mathtt{abb}) \vdash (4, \mathtt{bb})$

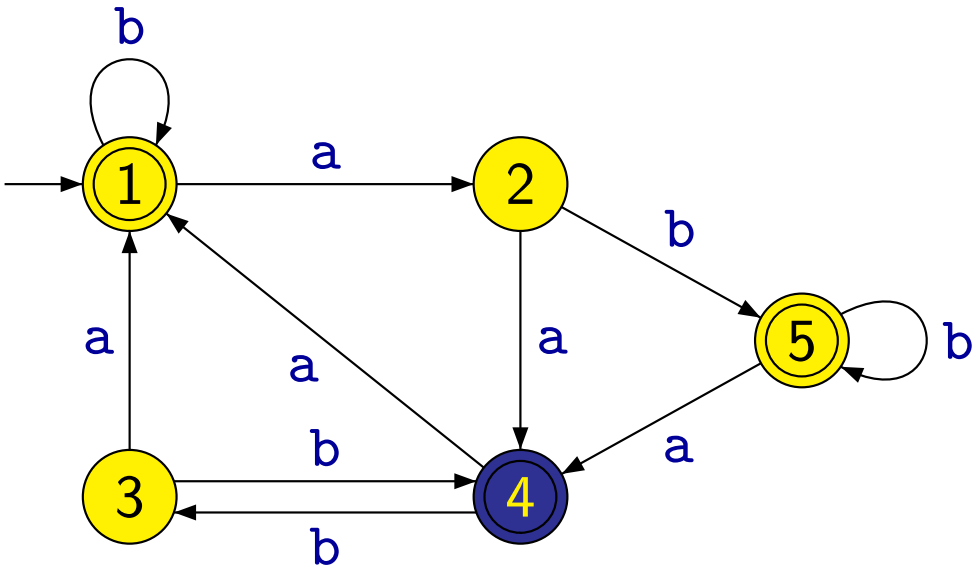
Deterministický konečný automat

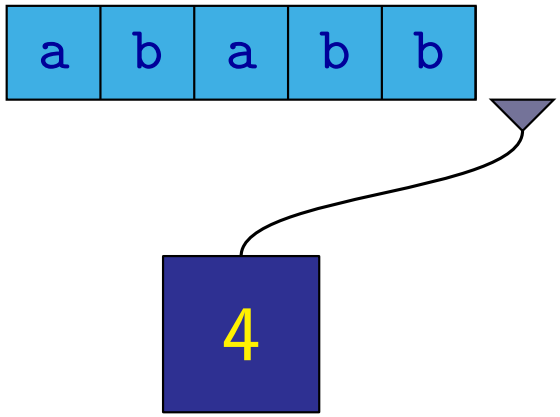




$(1, \mathtt{ababb}) \vdash (2, \mathtt{babb}) \vdash$
$(5, \mathtt{abb}) \vdash (4, \mathtt{bb}) \vdash$
$(3, \mathtt{b})$

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 relaci $\vdash^*$, jejíž význam je takový, 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

$$C_0 \vdash C_1 \vdash C_2 \vdash \cdots \vdash C_k$$

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

Definice

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

Definice

Automat **přijímá** slovo $w \in \Sigma^*$ právě tehdy, jestliže výpočet začínající v počáteční konfiguraci (q_0, w) skončí v přijímající koncové konfiguraci.

Poznámka: Formálně to můžeme definovat tak, že automat přijímá slovo $w \in \Sigma^*$ právě když $(q_0, w) \vdash^* (q, \varepsilon)$ pro nějaké $q \in F$.

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.

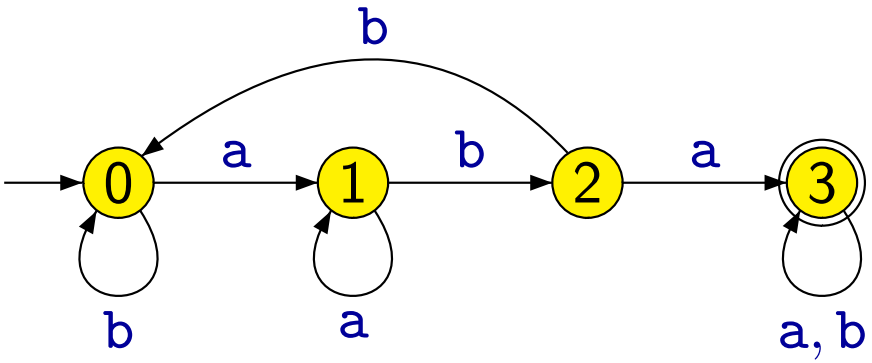
$$L(A) = \{w \in \Sigma^* \mid (q_0, w) \vdash^* (q, \varepsilon),\ q \in F\}$$

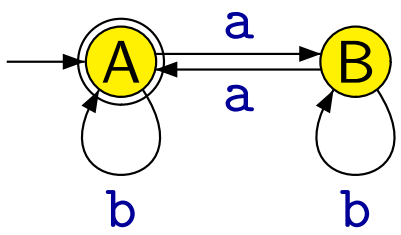
Definice

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

Automat pro průnik jazyků

Máme následující dva automaty:

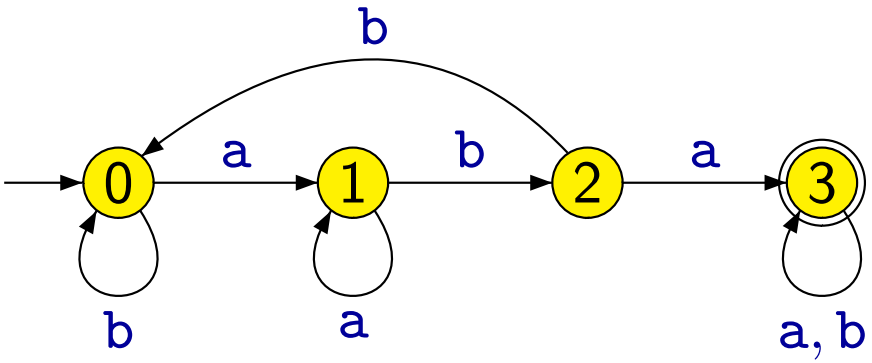


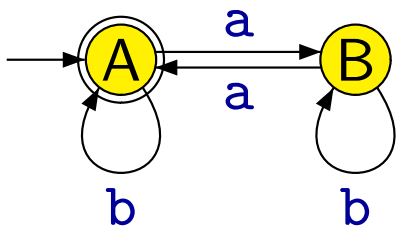


Příjmou oba slovo ababb?

Automat pro průnik jazyků

Máme následující dva automaty:



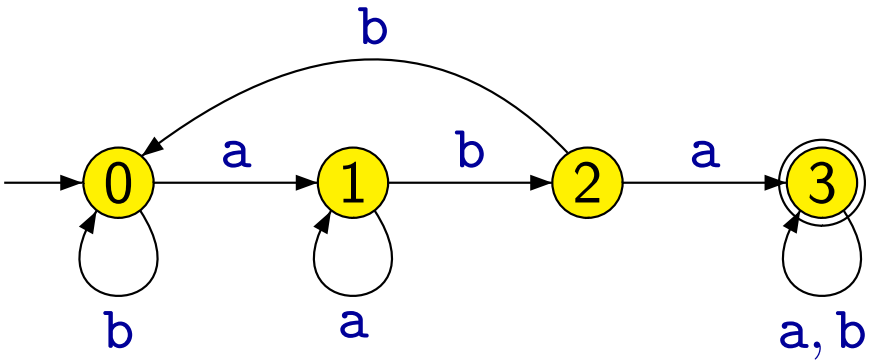


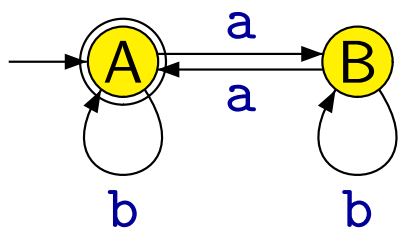
Příjmou oba slovo `ababb`?

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

Automat pro průnik jazyků

Máme následující dva automaty:



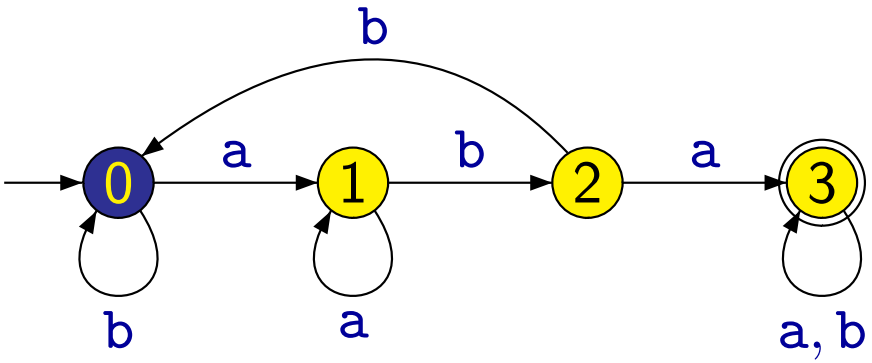


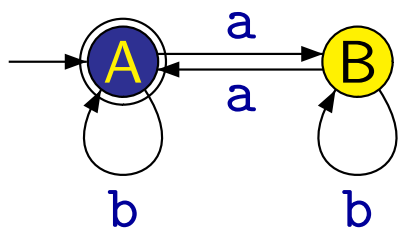
Příjmou oba slovo `ababb`?

Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ano, pokud oba odpoví ano.
Lepší je číst slovo současně oběma automaty.

Automat pro průnik jazyků

Máme následující dva automaty:



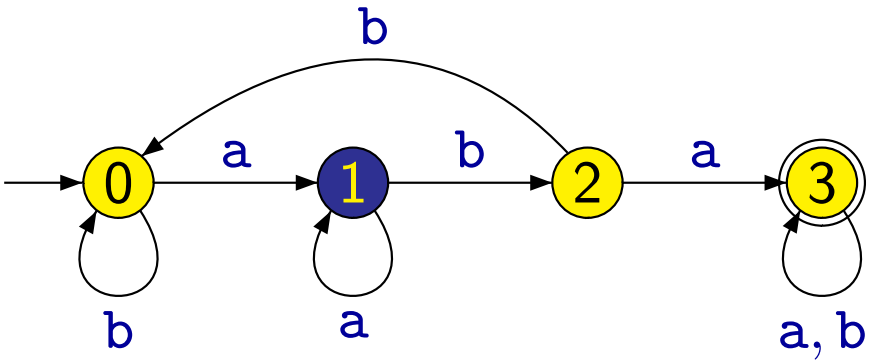


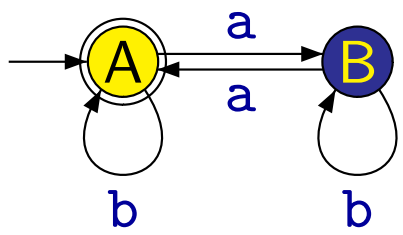
Příjmou oba slovo ababb?

Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ano, pokud oba odpoví ano.
Lepší je číst slovo současně oběma automaty.

Automat pro průnik jazyků

Máme následující dva automaty:



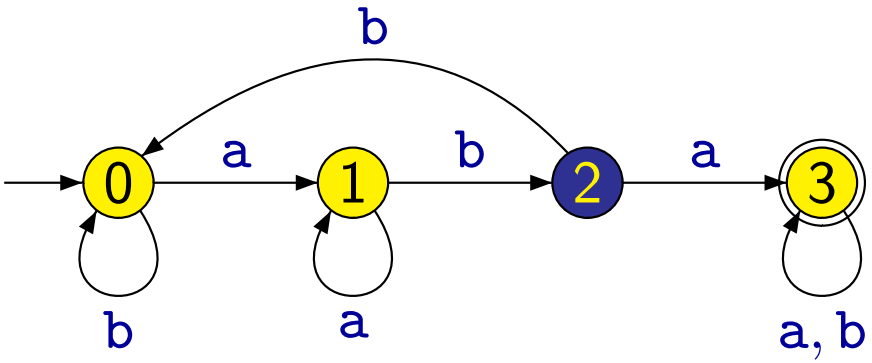


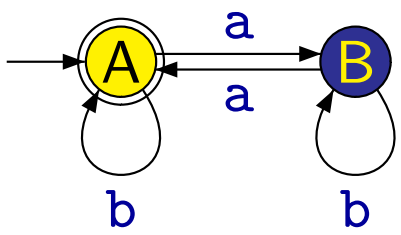
Příjmou oba slovo ababb?

Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ano, pokud oba odpoví ano.
Lepší je číst slovo současně oběma automaty.

Automat pro průnik jazyků

Máme následující dva automaty:



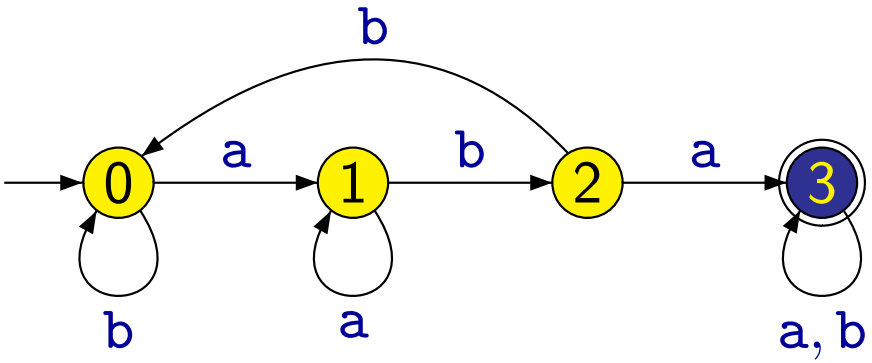


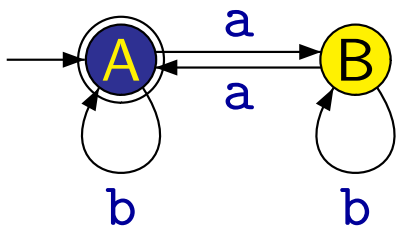
Příjmou oba slovo ababb?

Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ano, pokud oba odpoví ano.
Lepší je číst slovo současně oběma automaty.

Automat pro průnik jazyků

Máme následující dva automaty:



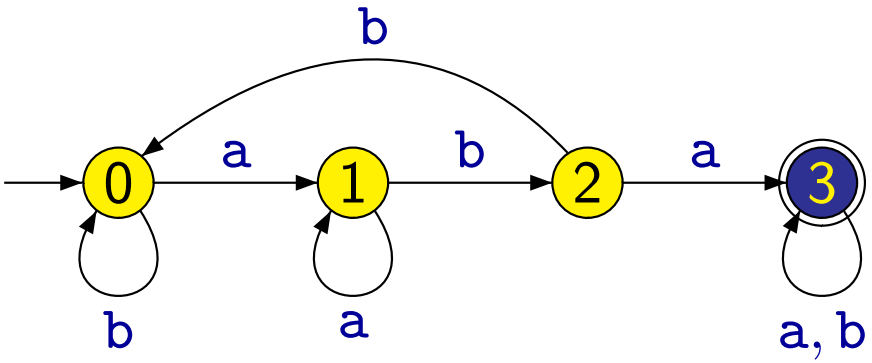


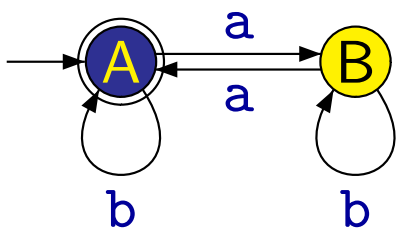
Příjmou oba slovo ababb?

Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ano, pokud oba odpoví ano.
Lepší je číst slovo současně oběma automaty.

Automat pro průnik jazyků

Máme následující dva automaty:



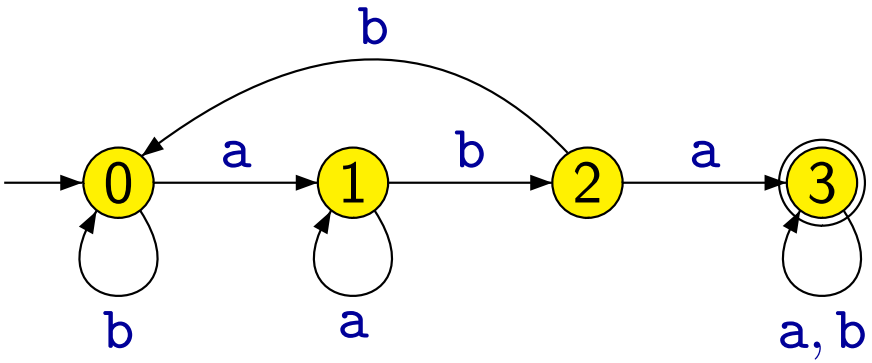


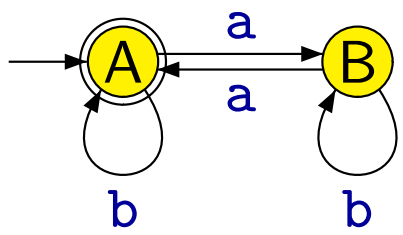
Příjmou oba slovo ababb?

Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ano, pokud oba odpoví ano.
Lepší je číst slovo současně oběma automaty.

Automat pro průnik jazyků

Máme následující dva automaty:

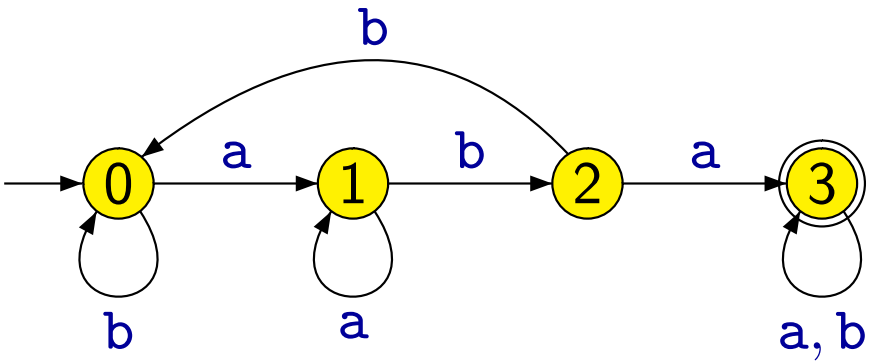


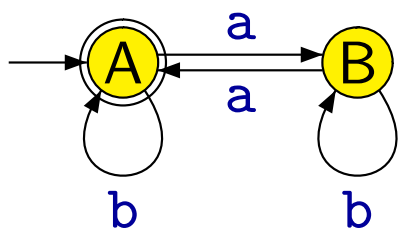


Příjmou oba slovo `ababb`?

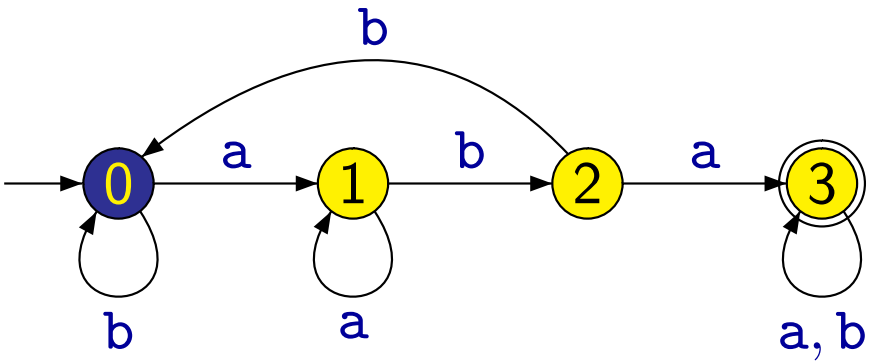
Můžeme postupně nechat přečíst slovo oběma automatům. Odpověď bude ano, pokud oba odpoví ano.
Lepší je číst slovo současně oběma automaty.
Situace při tomto postupu je dána nepřečtenou částí slova a aktuálními stavy obou automatů. Zkusíme vytvořit automat, který toto má jako své konfigurace.

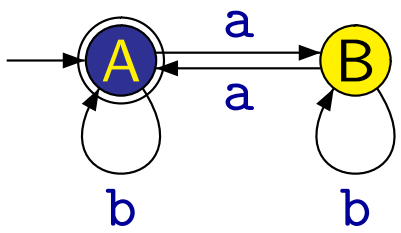
Automat pro průnik jazyků

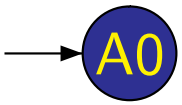




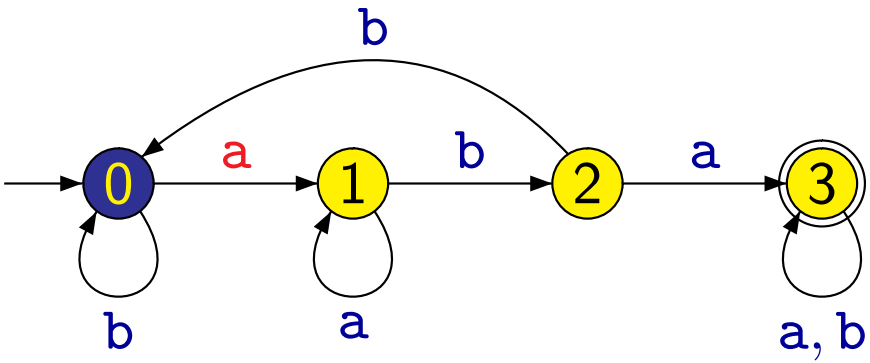
Automat pro průnik jazyků

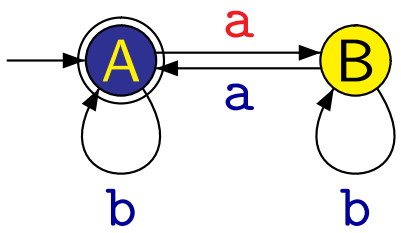




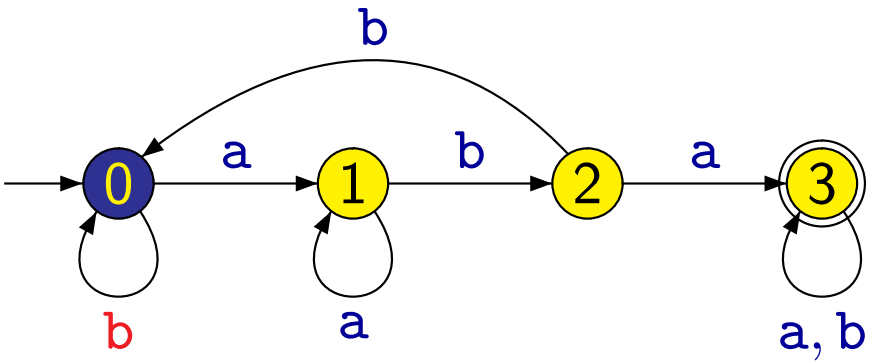


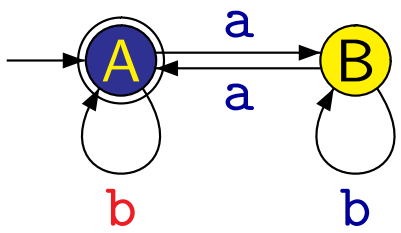
Automat pro průnik jazyků

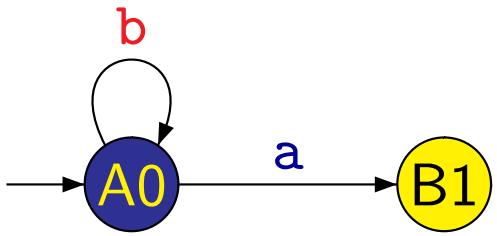




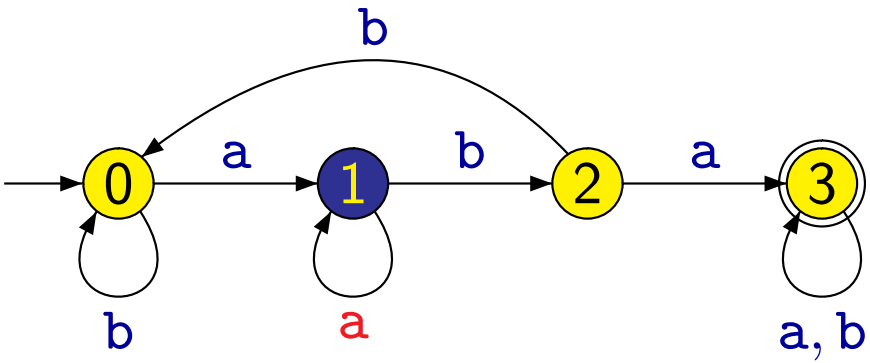
Automat pro průnik jazyků

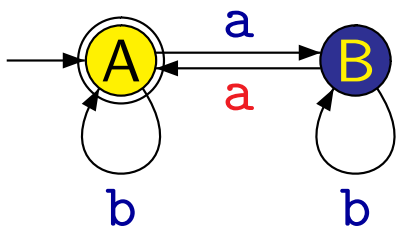


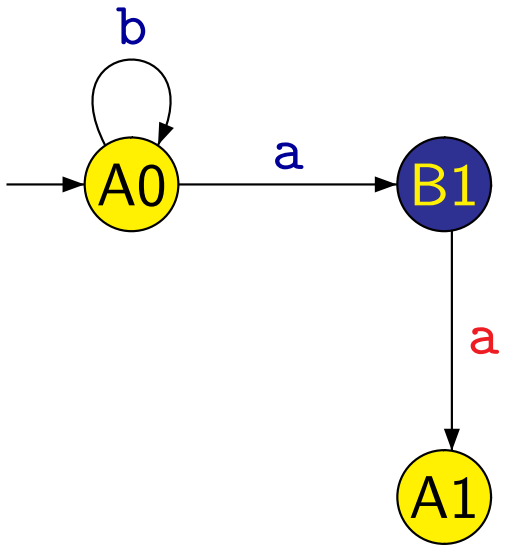




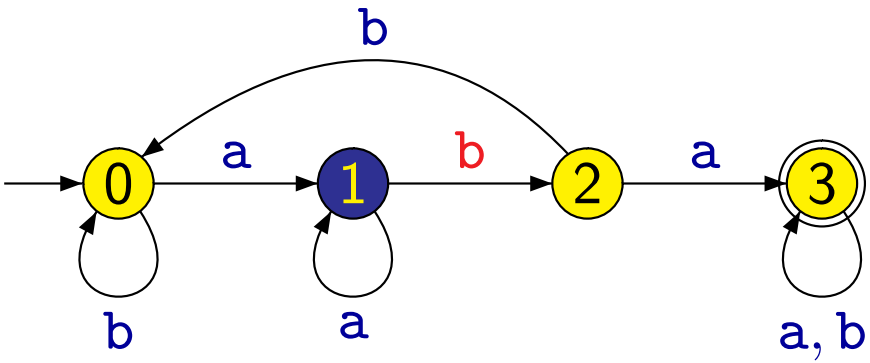
Automat pro průnik jazyků

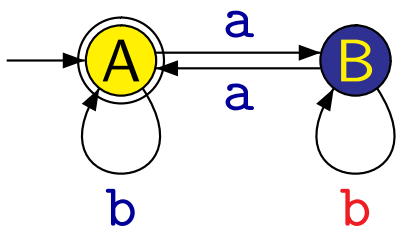


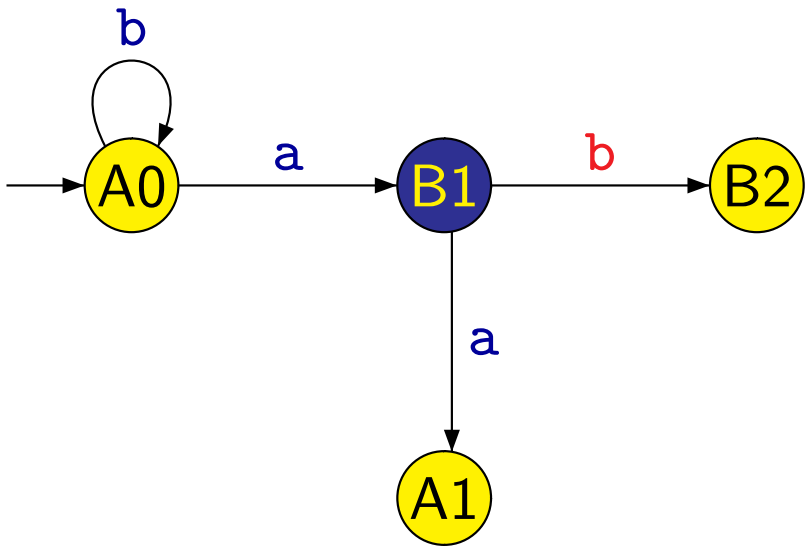




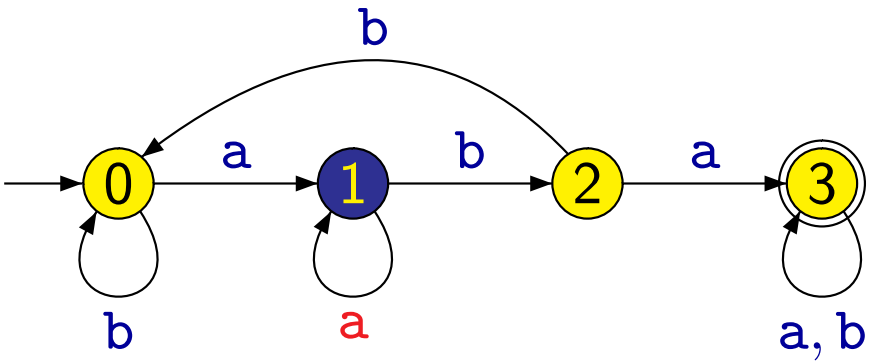
Automat pro průnik jazyků

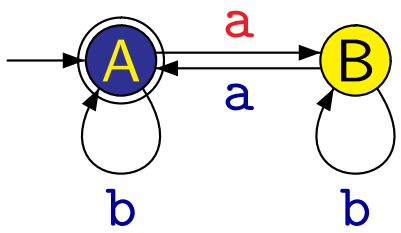


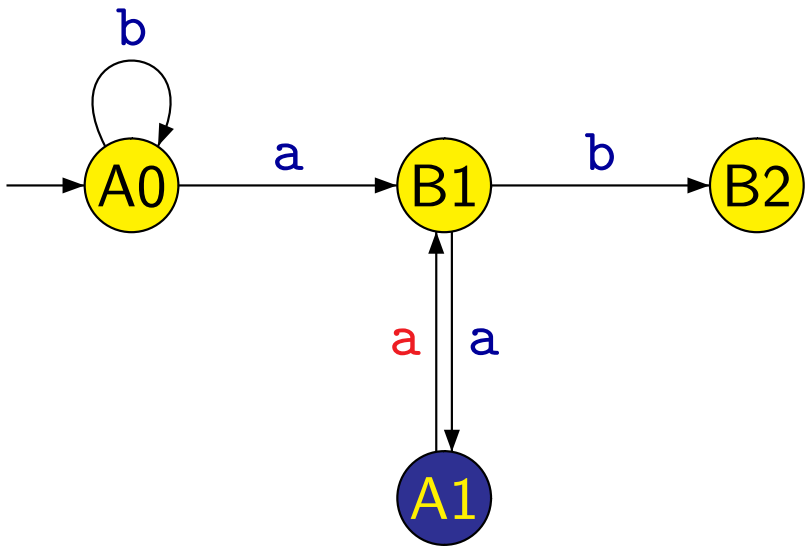




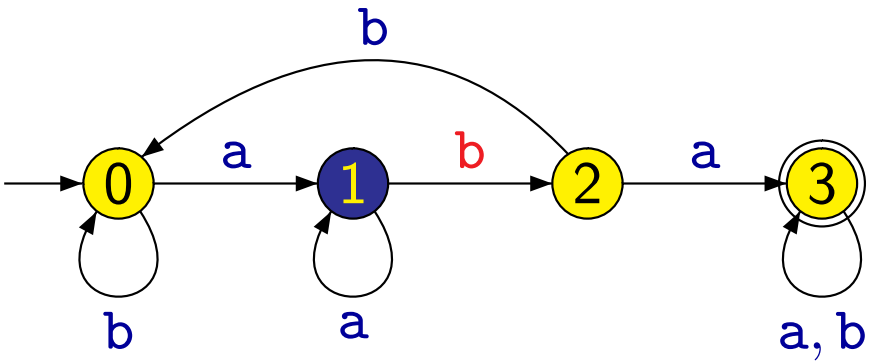
Automat pro průnik jazyků

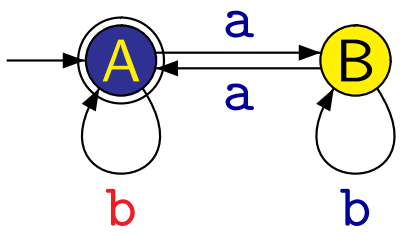


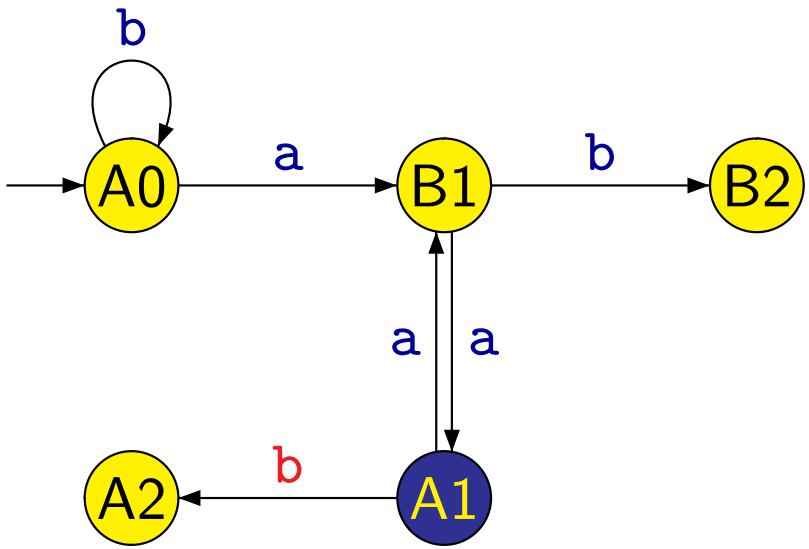




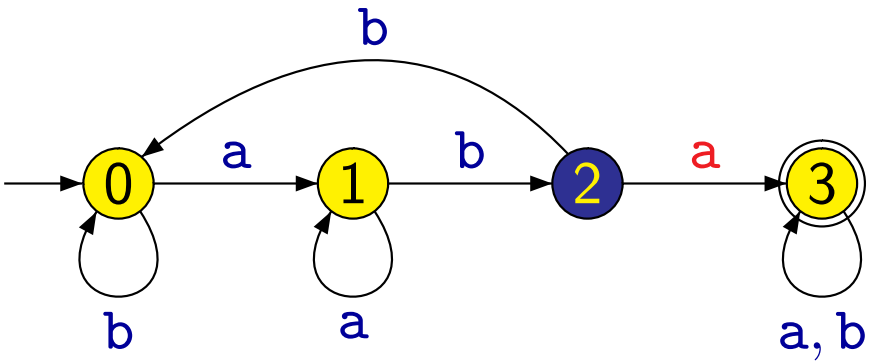
Automat pro průnik jazyků

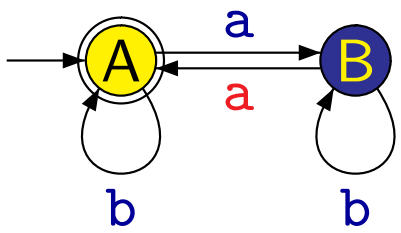


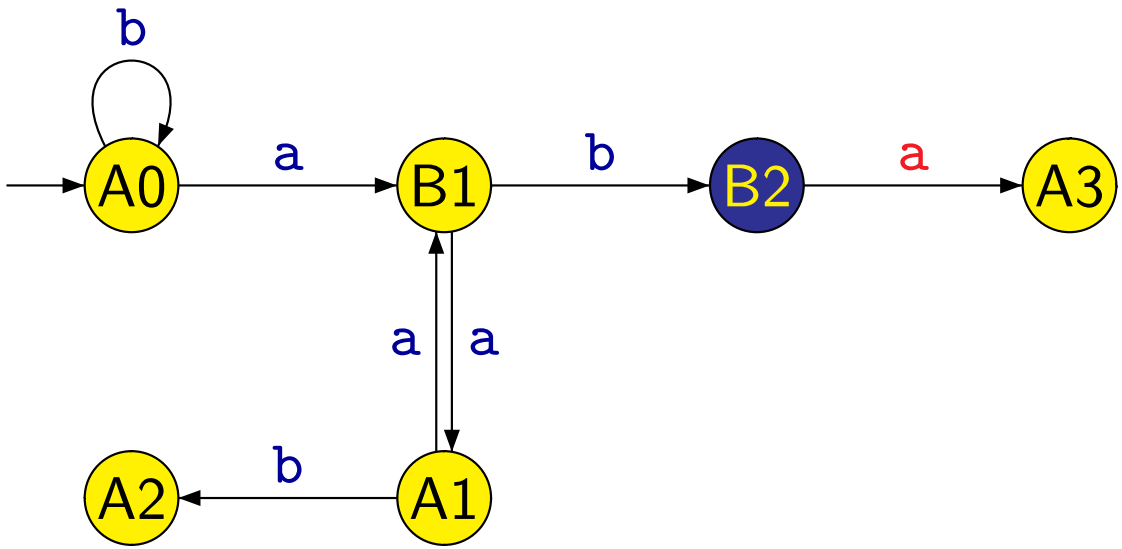




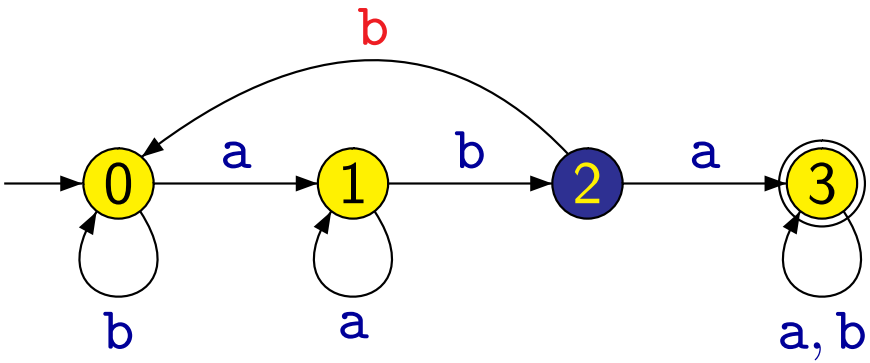
Automat pro průnik jazyků

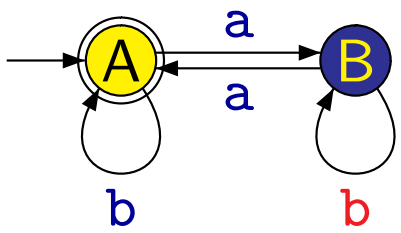


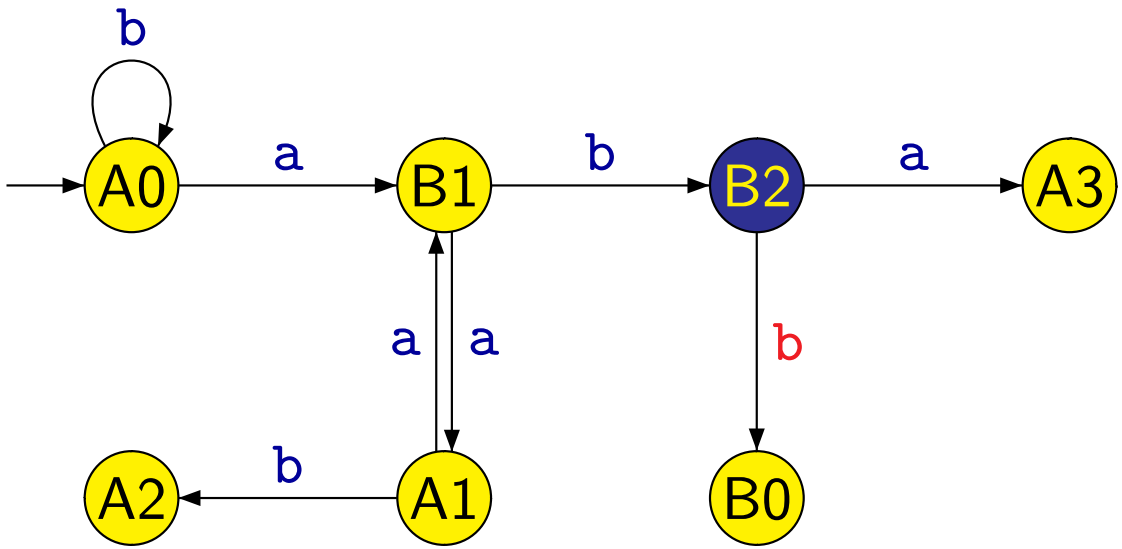




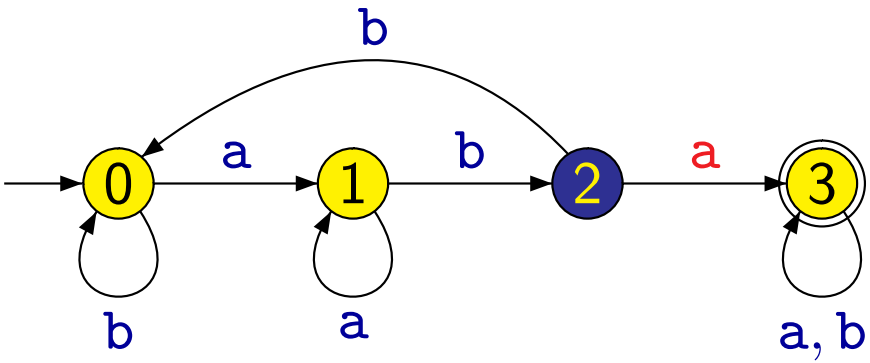
Automat pro průnik jazyků

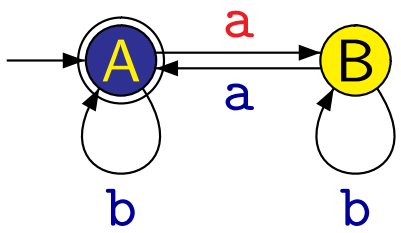


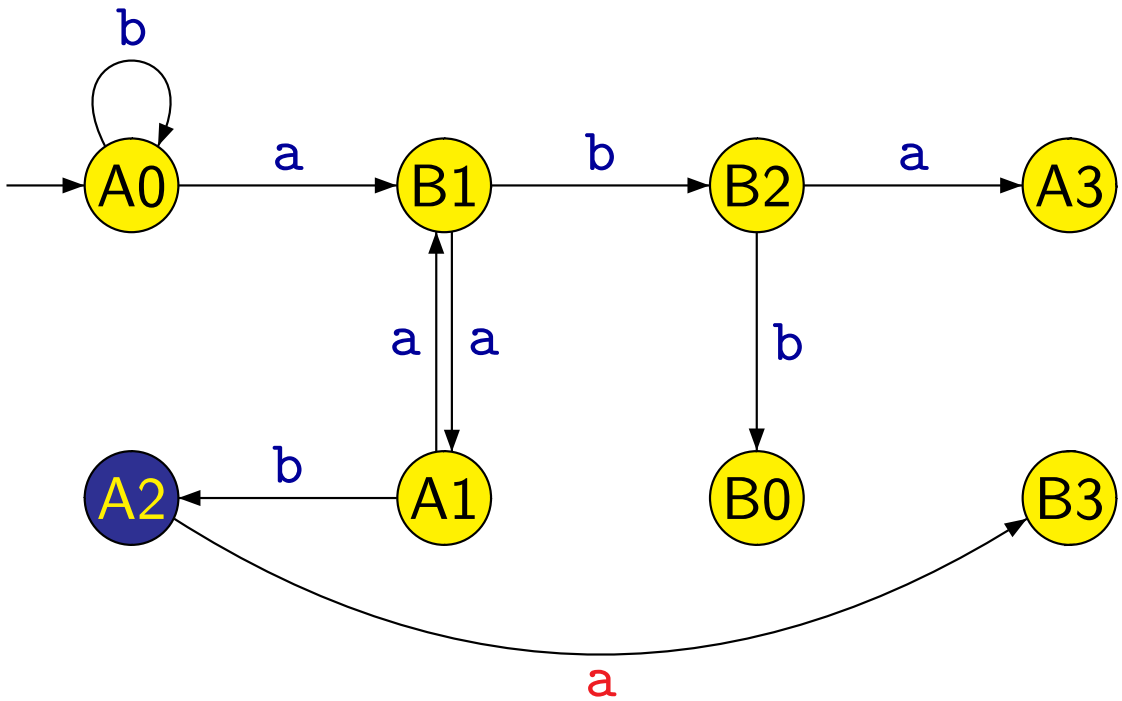




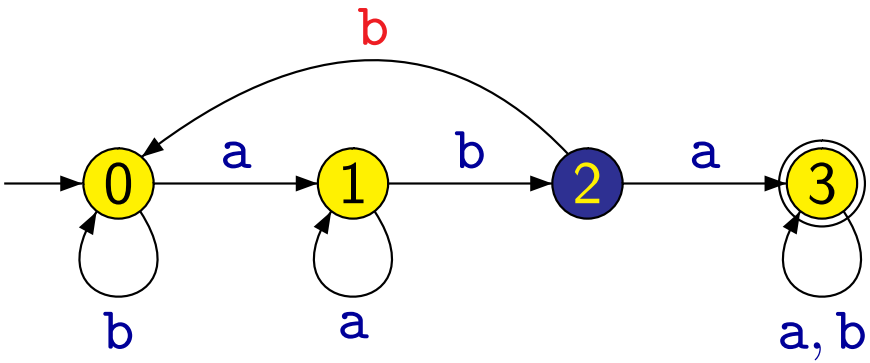
Automat pro průnik jazyků

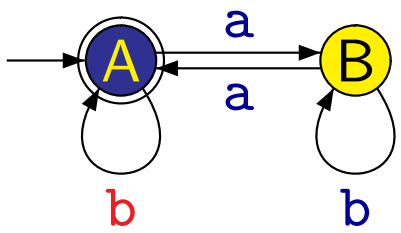


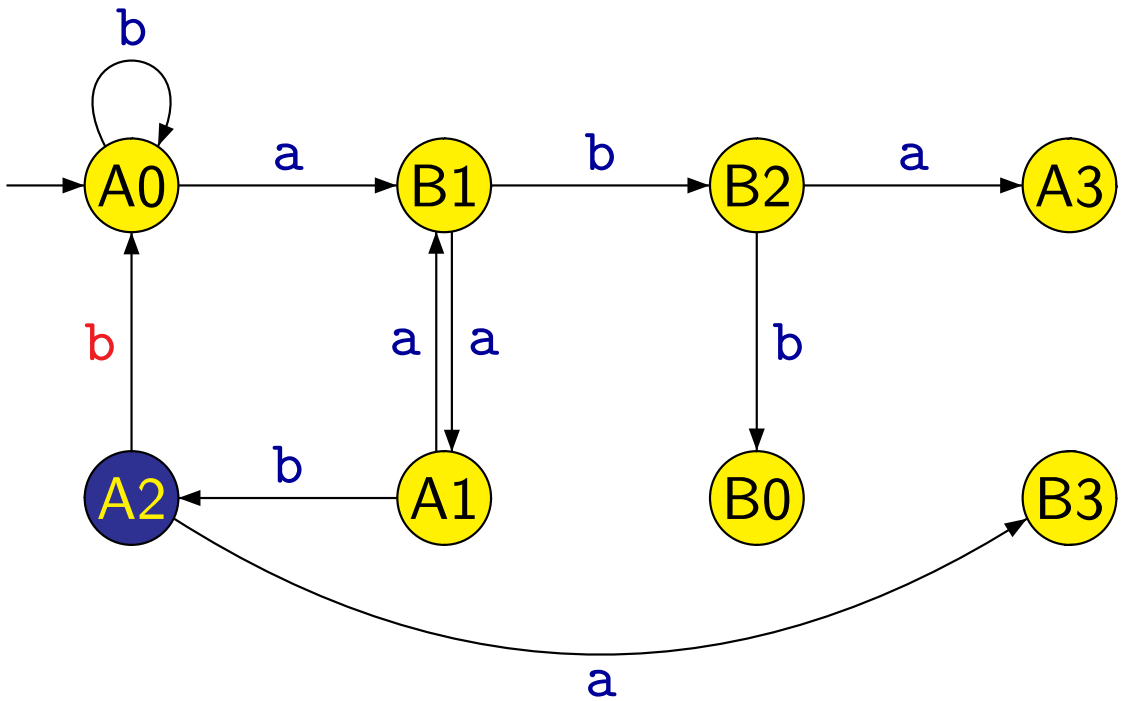




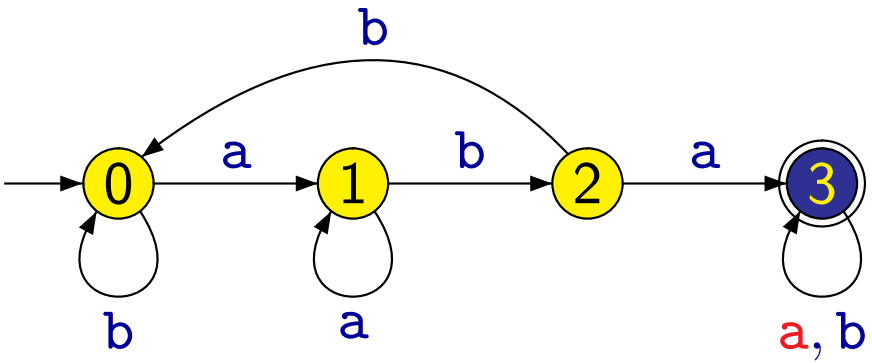
Automat pro průnik jazyků

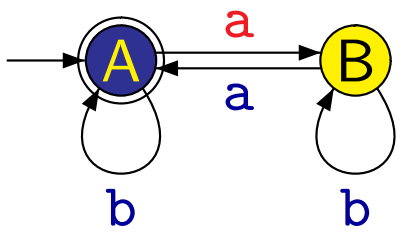


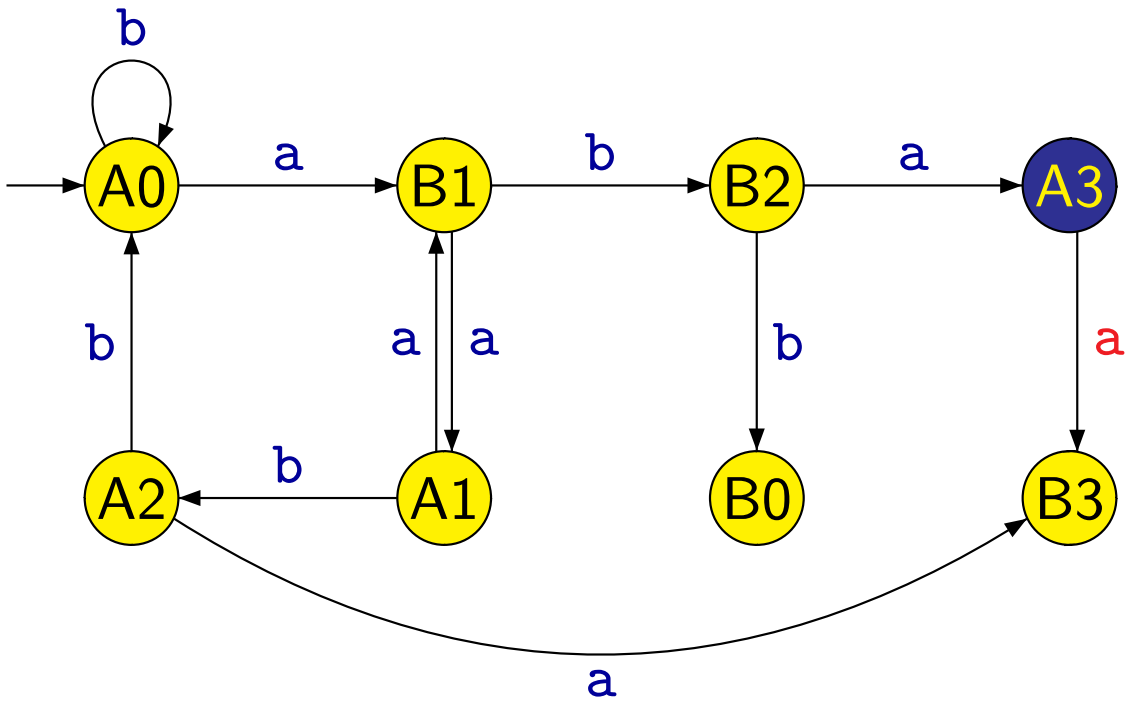




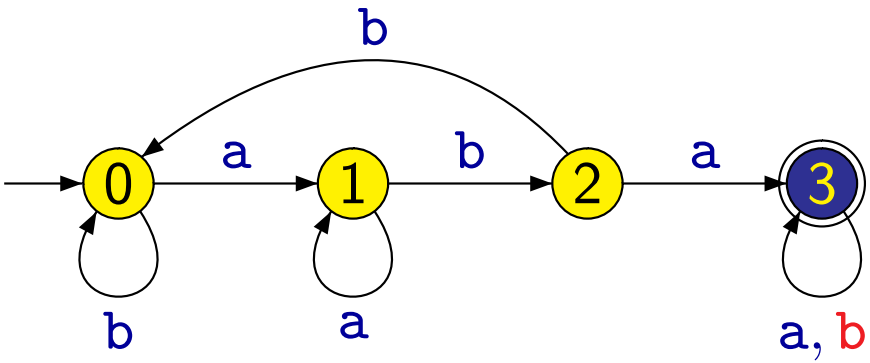
Automat pro průnik jazyků

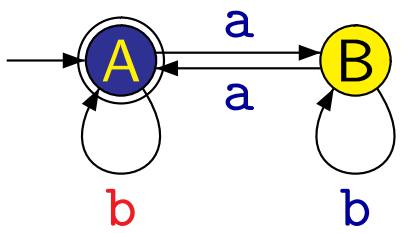


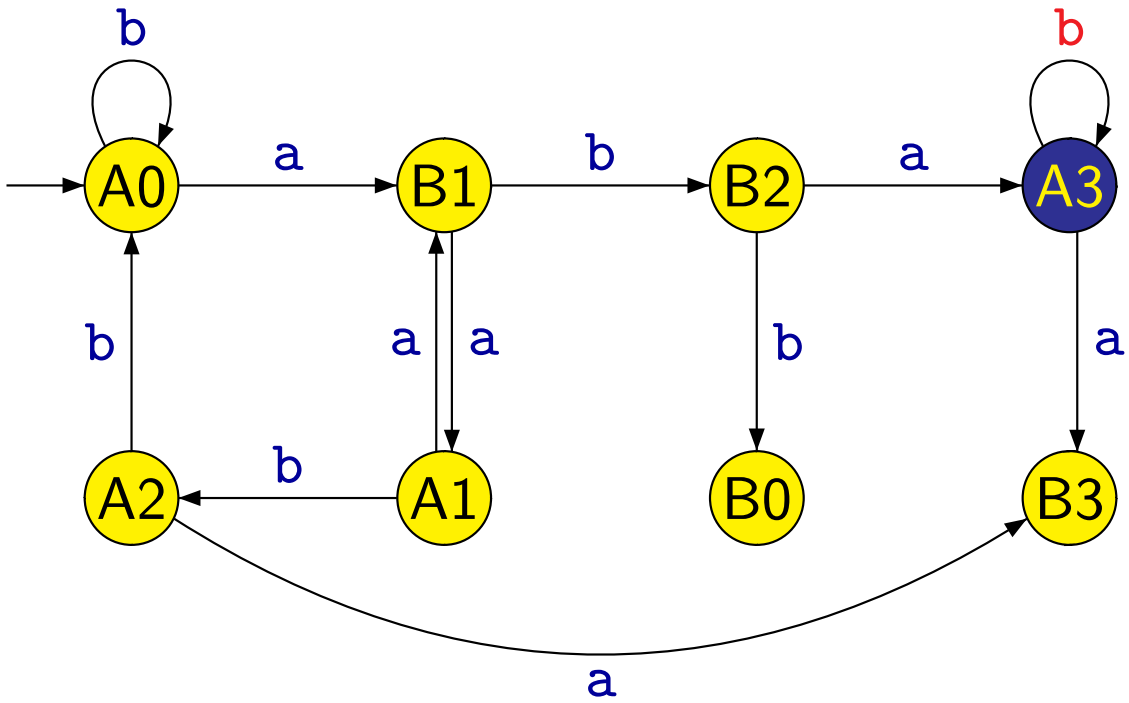




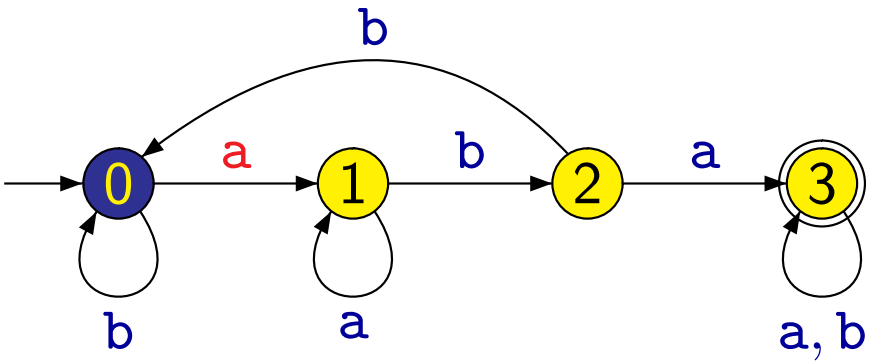
Automat pro průnik jazyků

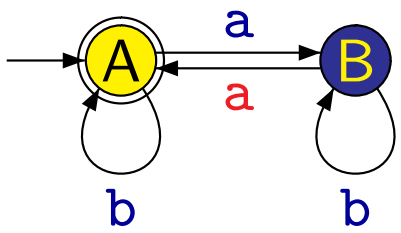


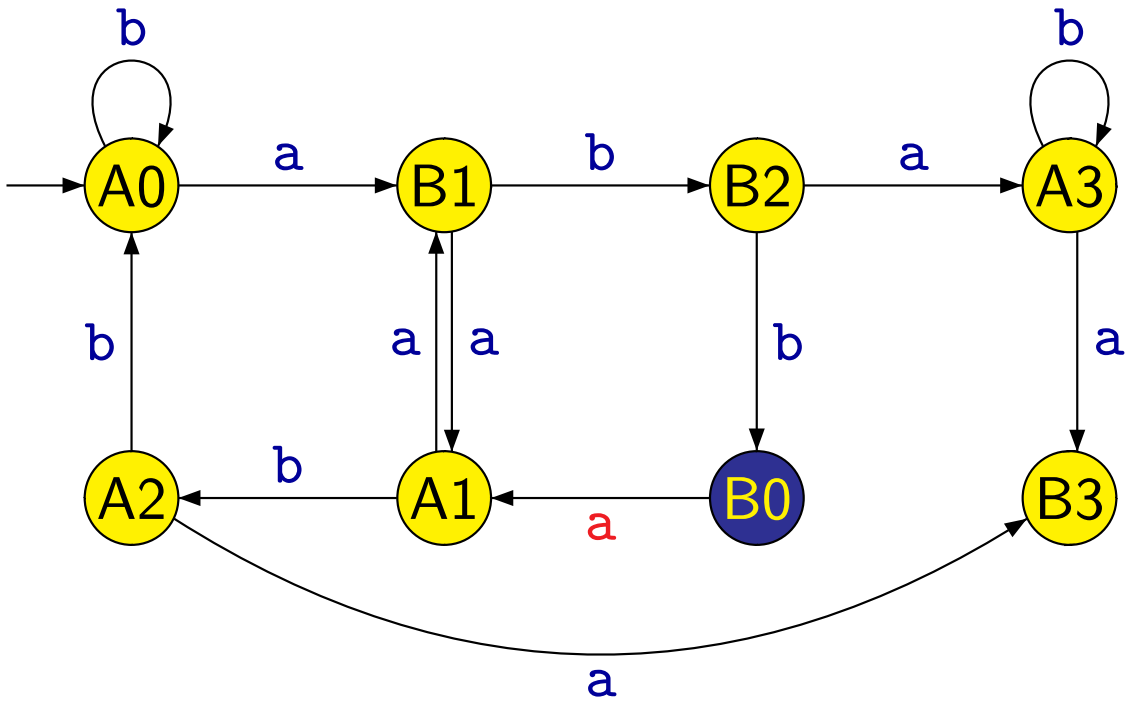




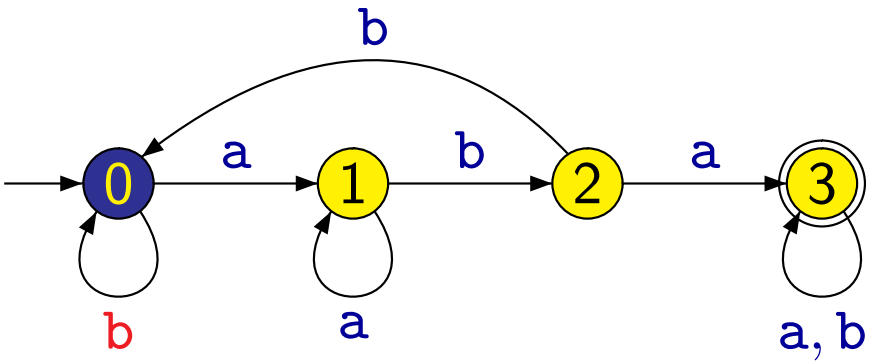
Automat pro průnik jazyků

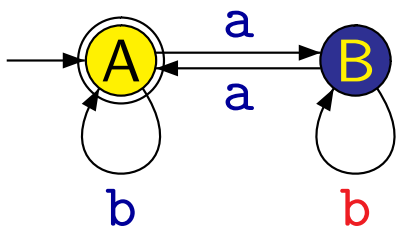


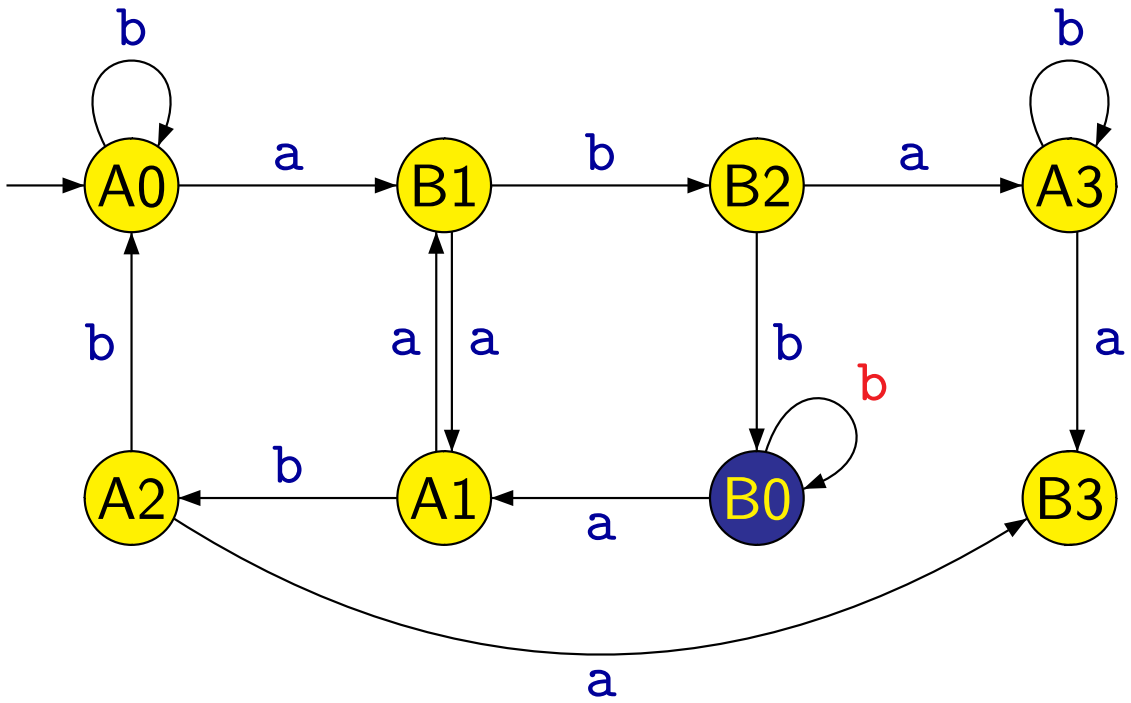




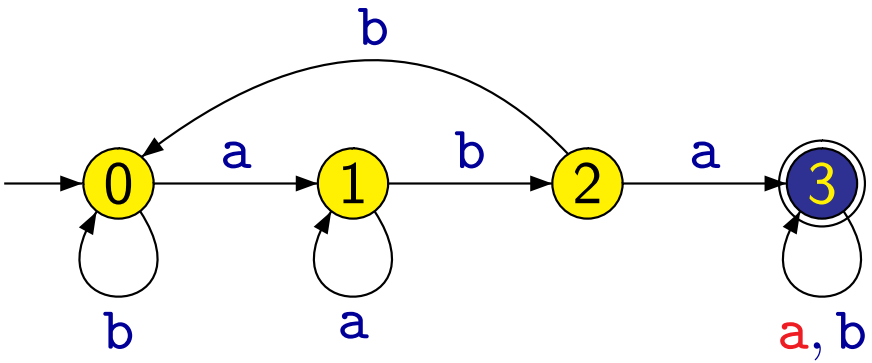
Automat pro průnik jazyků

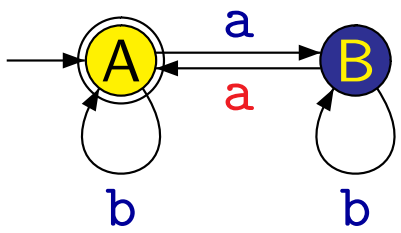


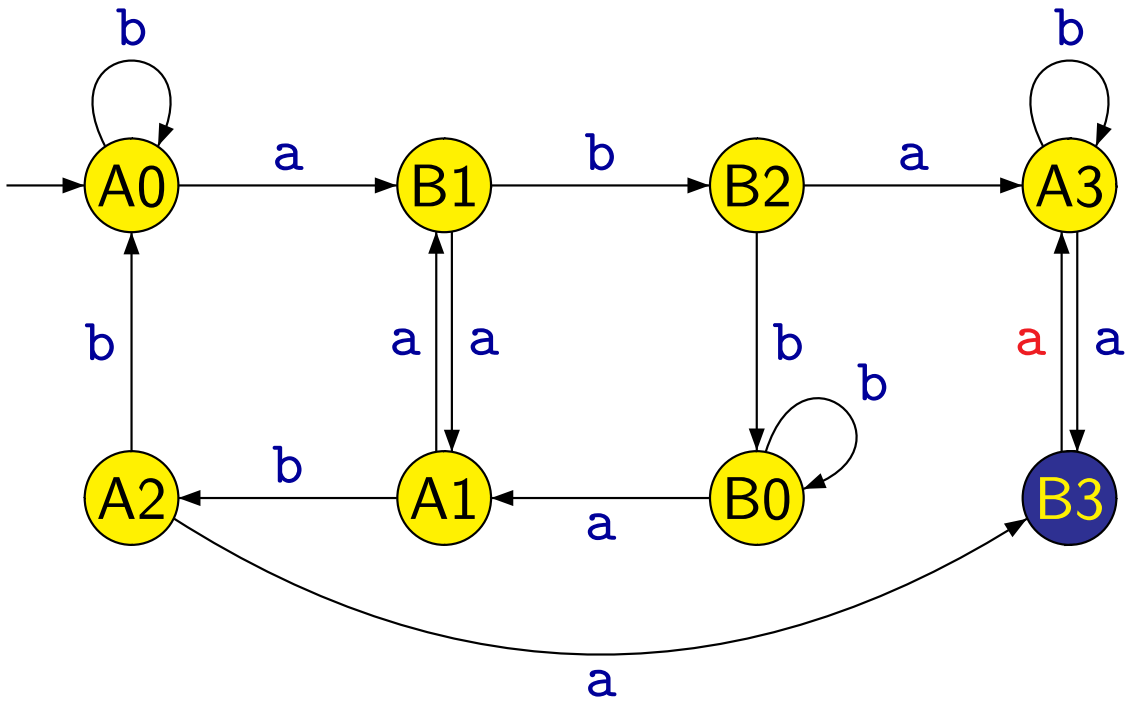




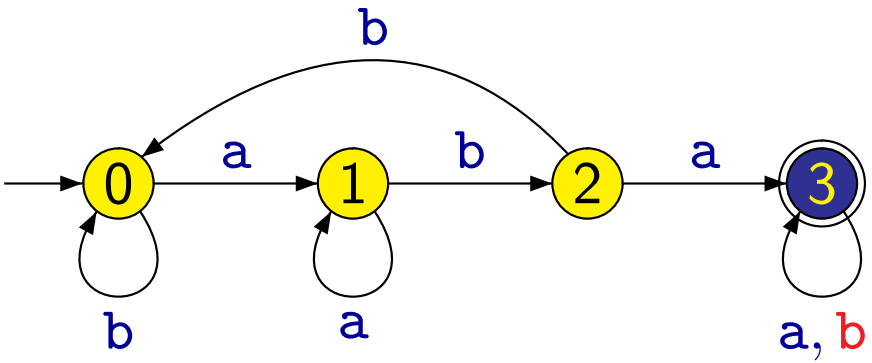
Automat pro průnik jazyků

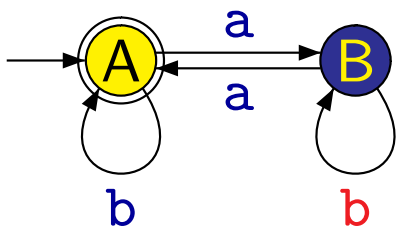


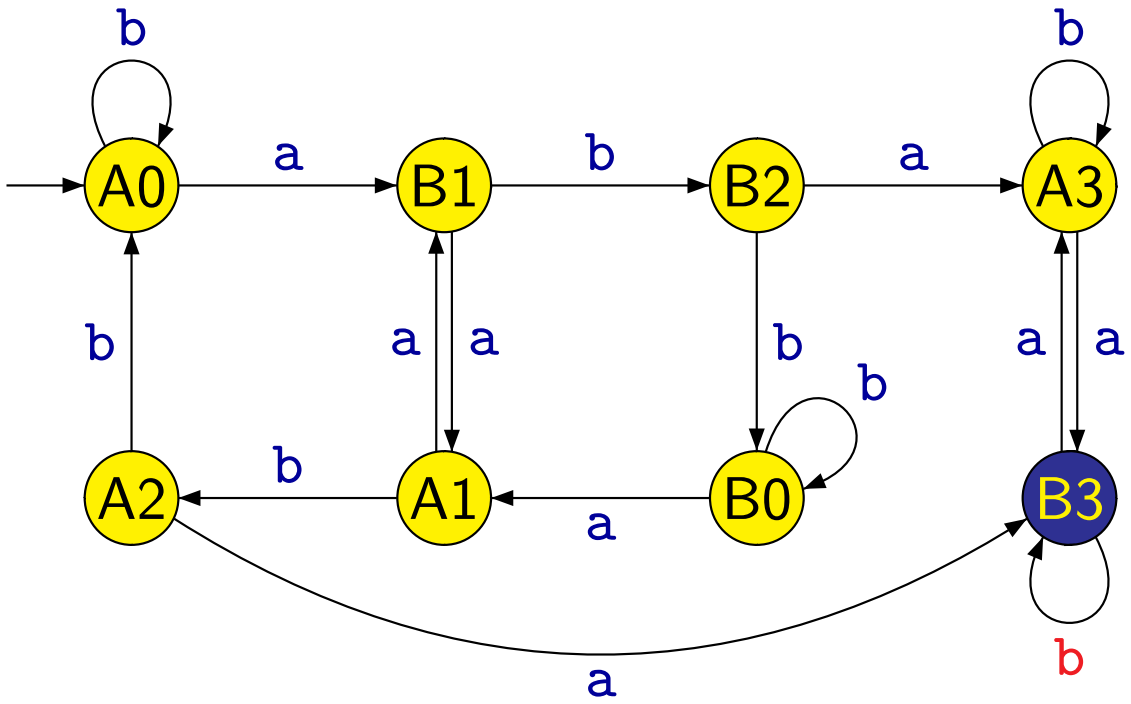




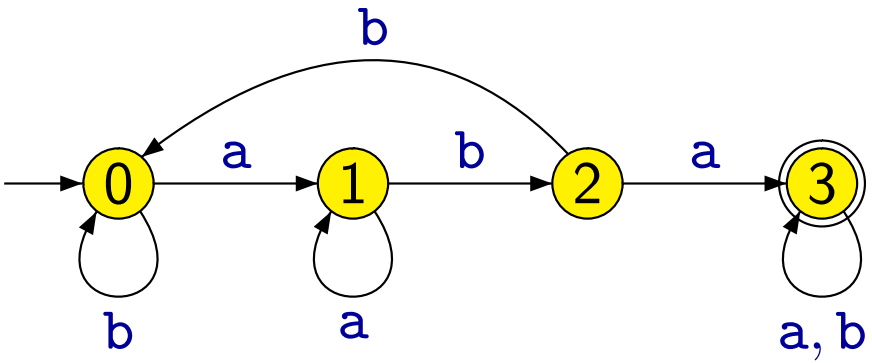
Automat pro průnik jazyků

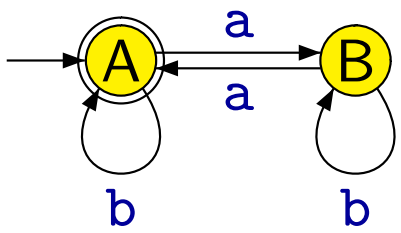


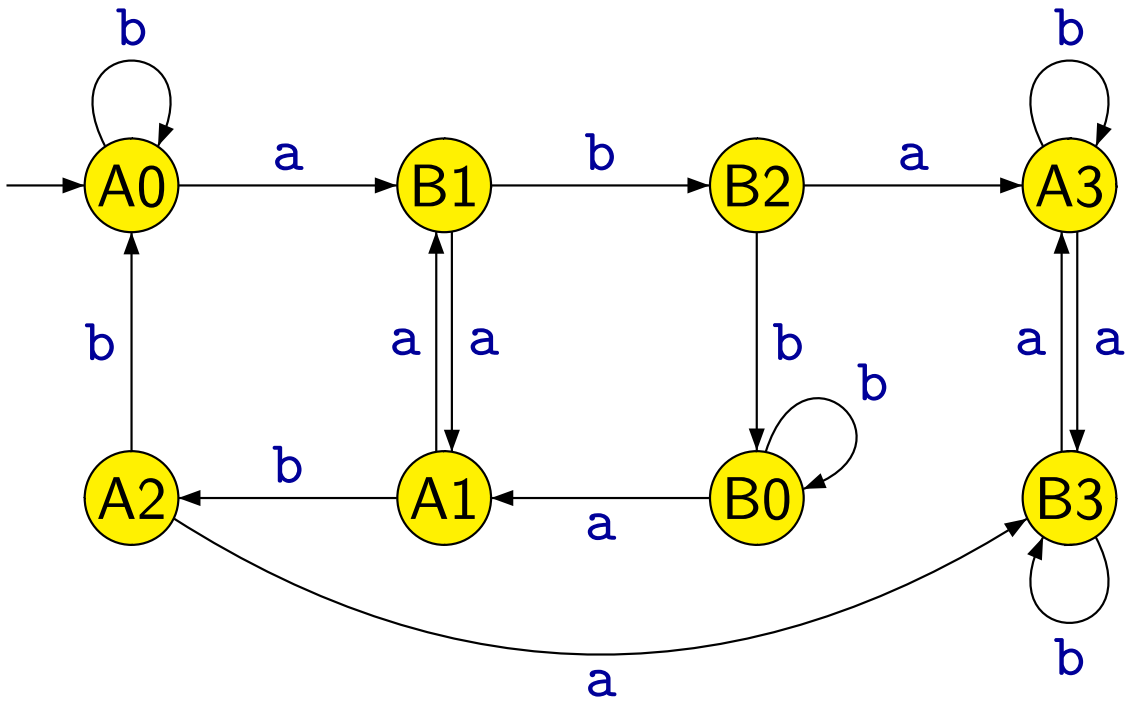




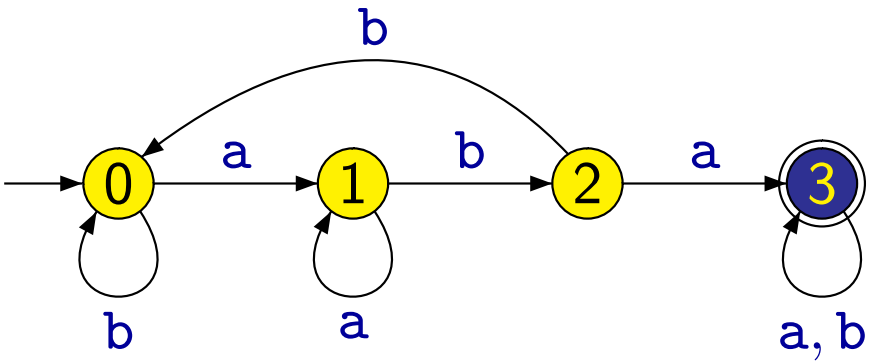
Automat pro průnik jazyků

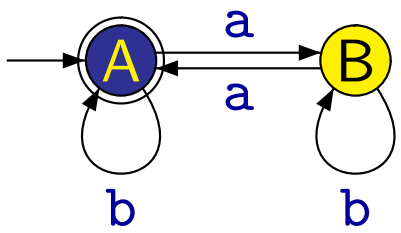


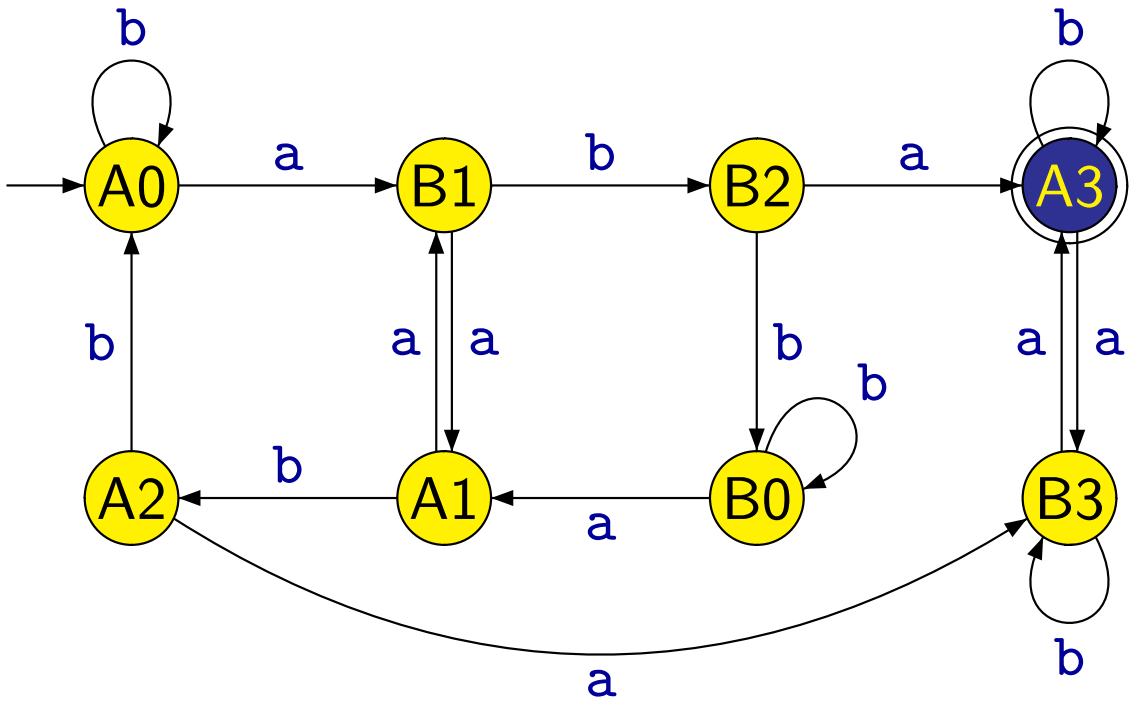




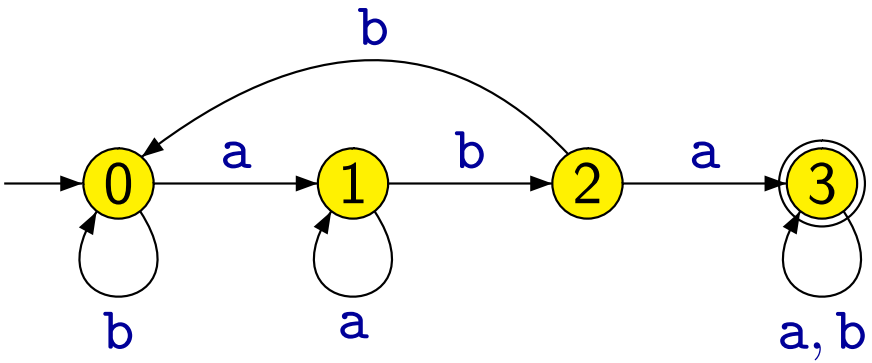
Automat pro průnik jazyků

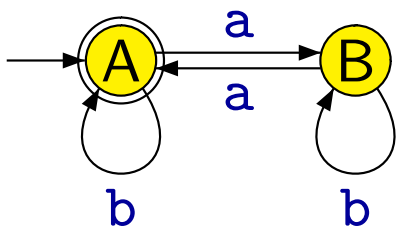


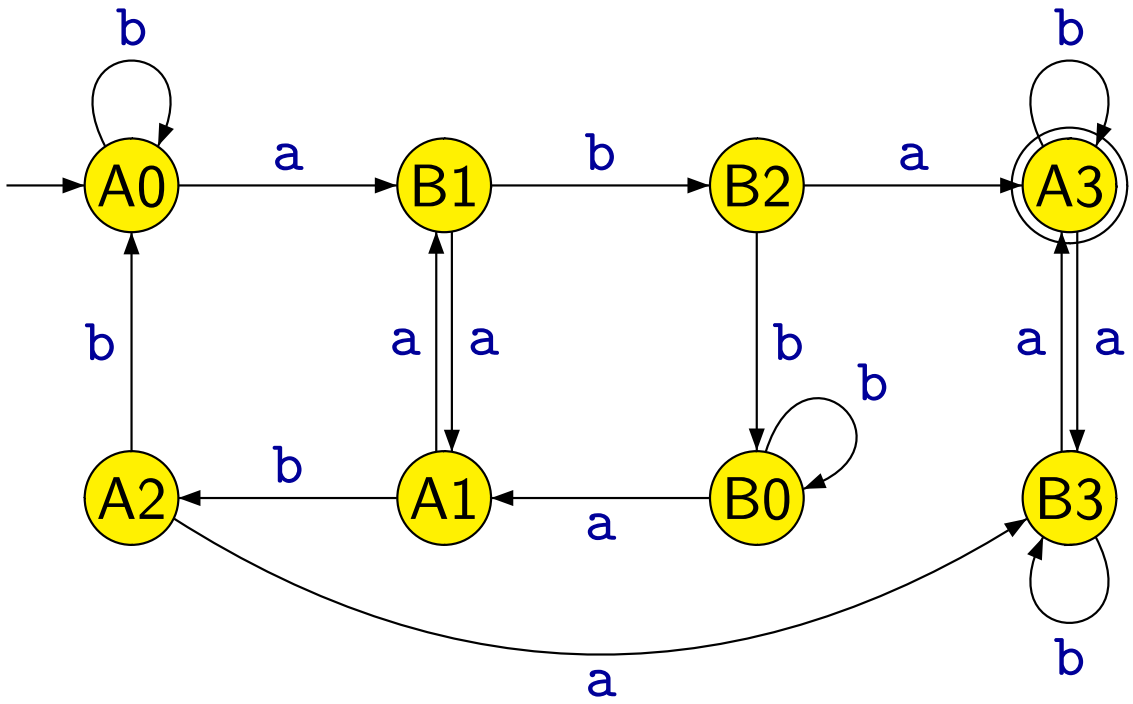




Automat pro průnik jazyků







Průnik regulárních jazyků

Věta

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

Průnik regulárních jazyků

Věta

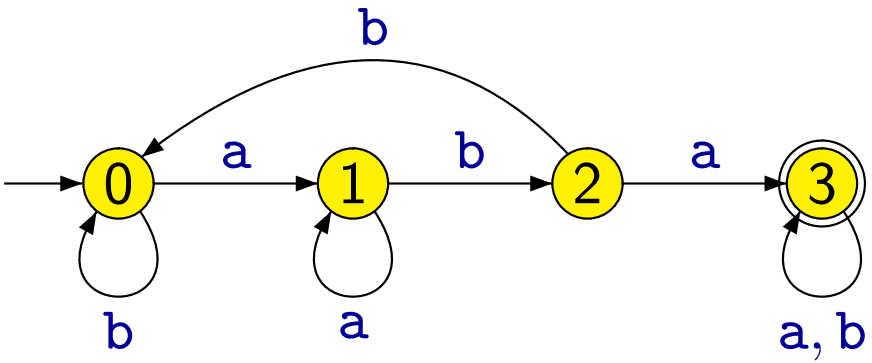
Jestliže jazyky $L_1, L_2 \subseteq \Sigma^*$ jsou regulární, pak také jazyk $L_1 \cap L_2$ je regulární.

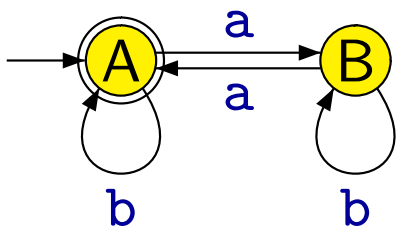
Důkaz: Nechť $L_1 = L(\mathcal{A}_1)$, $L_2 = L(\mathcal{A}_2)$ pro konečné automaty $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$. Definujeme automat $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ tž.

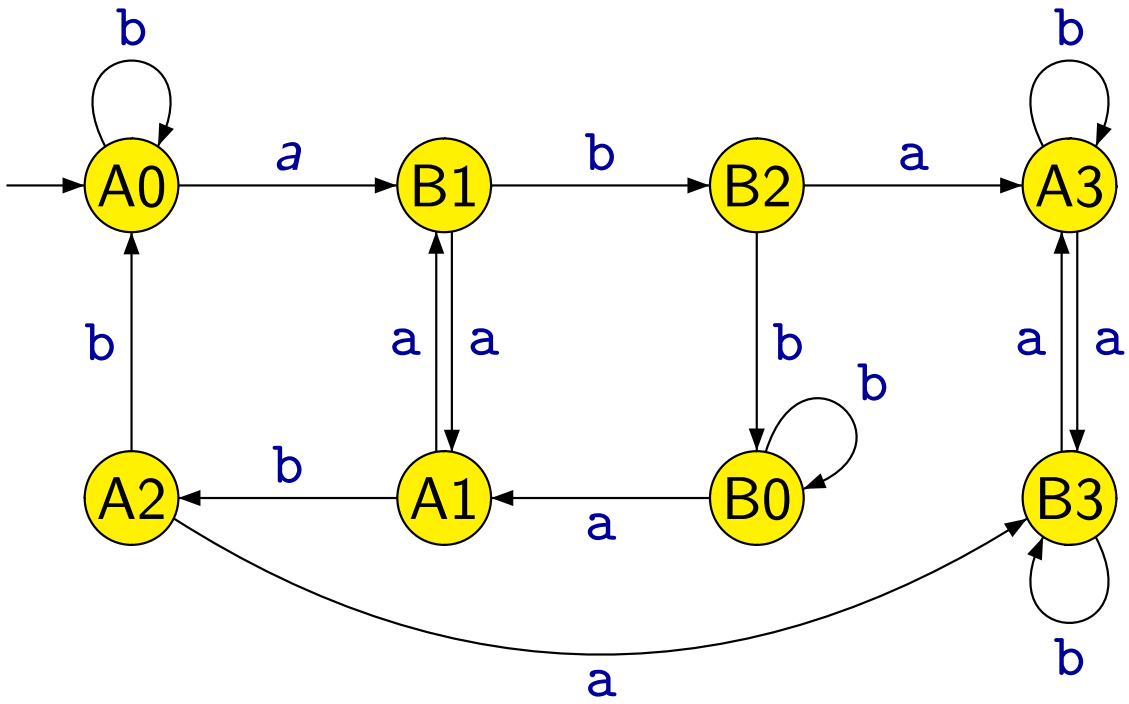
- $Q = Q_1 \times Q_2$,
- $\delta(\,(q_1, q_2), a\,) = (\,\delta_1(q_1, a), \delta_2(q_2, a)\,)$ pro všechna $q_1 \in Q_1$, $q_2 \in Q_2$, $a \in \Sigma$,
- $q_0 = (q_{01}, q_{02})$,
- $F = (F_1 \times F_2)$.

Od konstrukce pro sjednocení se tato liší jen množinou koncových stavů v sestrojeném automatu.

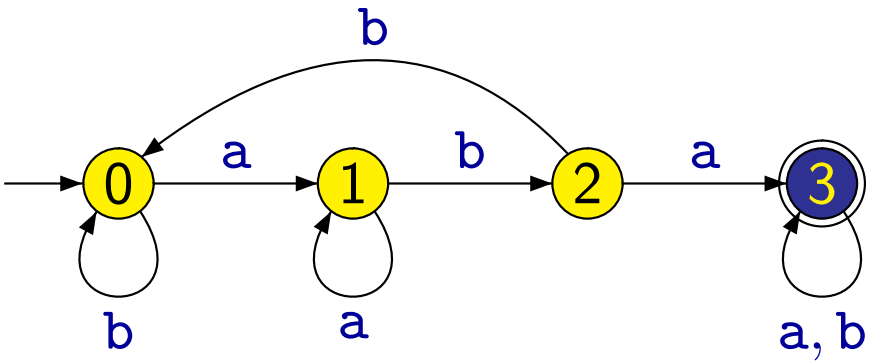
Automat pro sjednocení jazyků

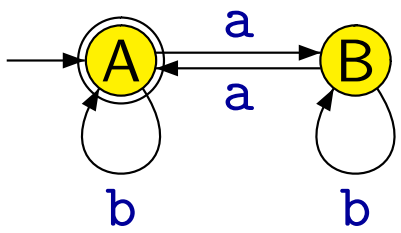


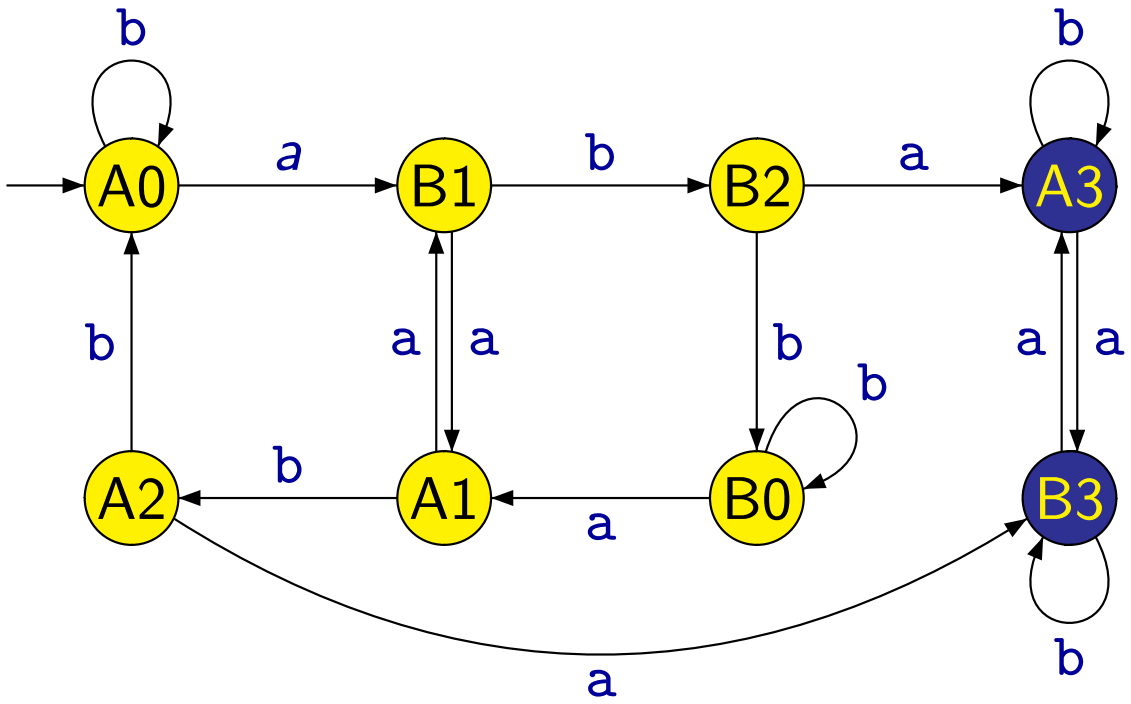




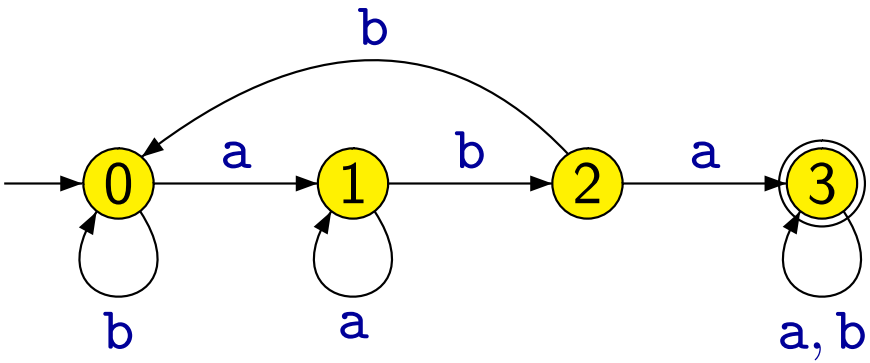
Automat pro sjednocení jazyků

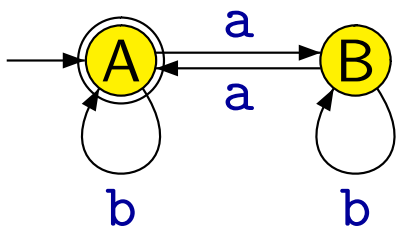


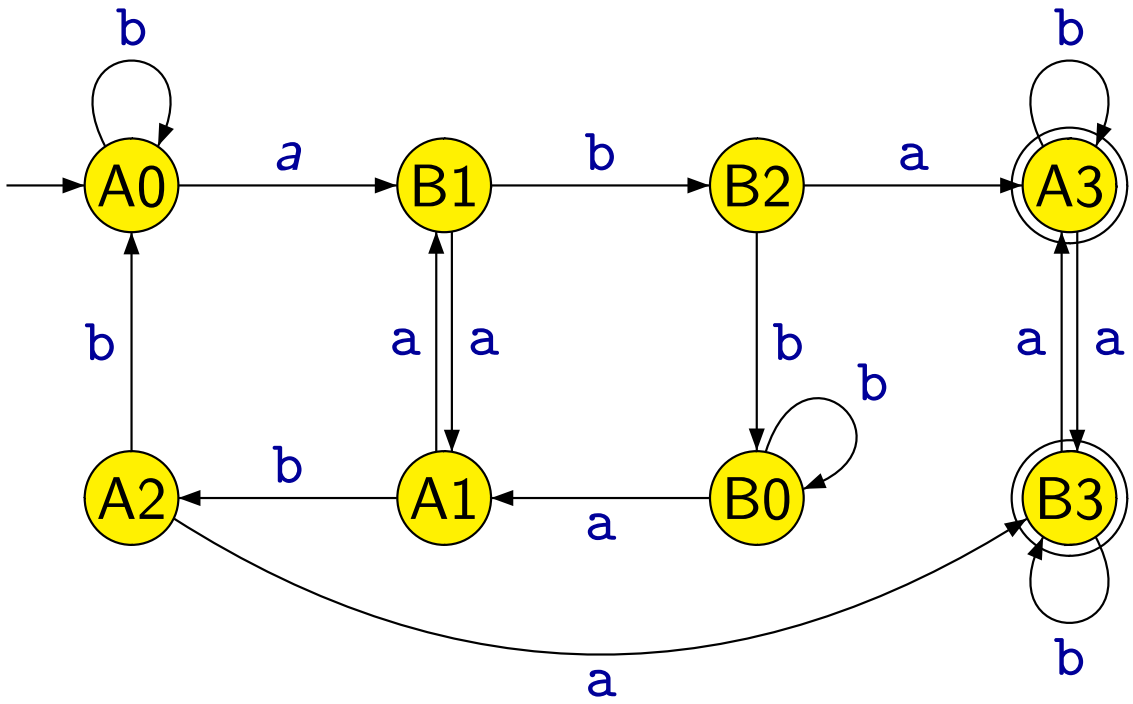




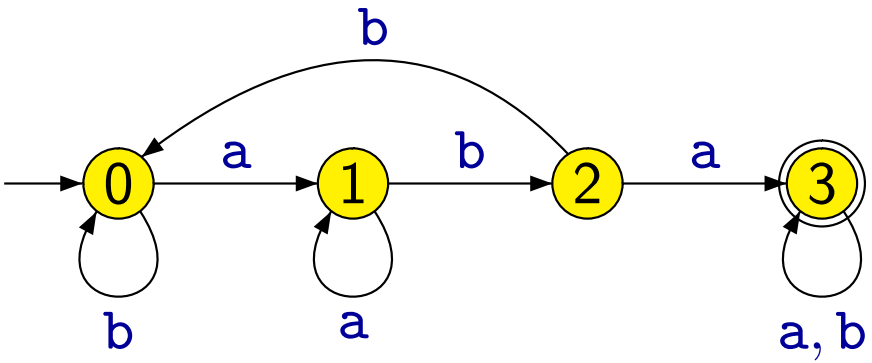
Automat pro sjednocení jazyků

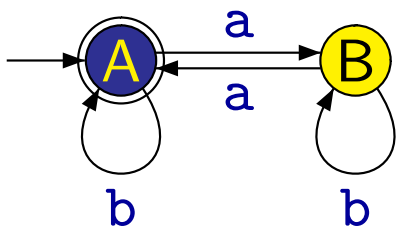


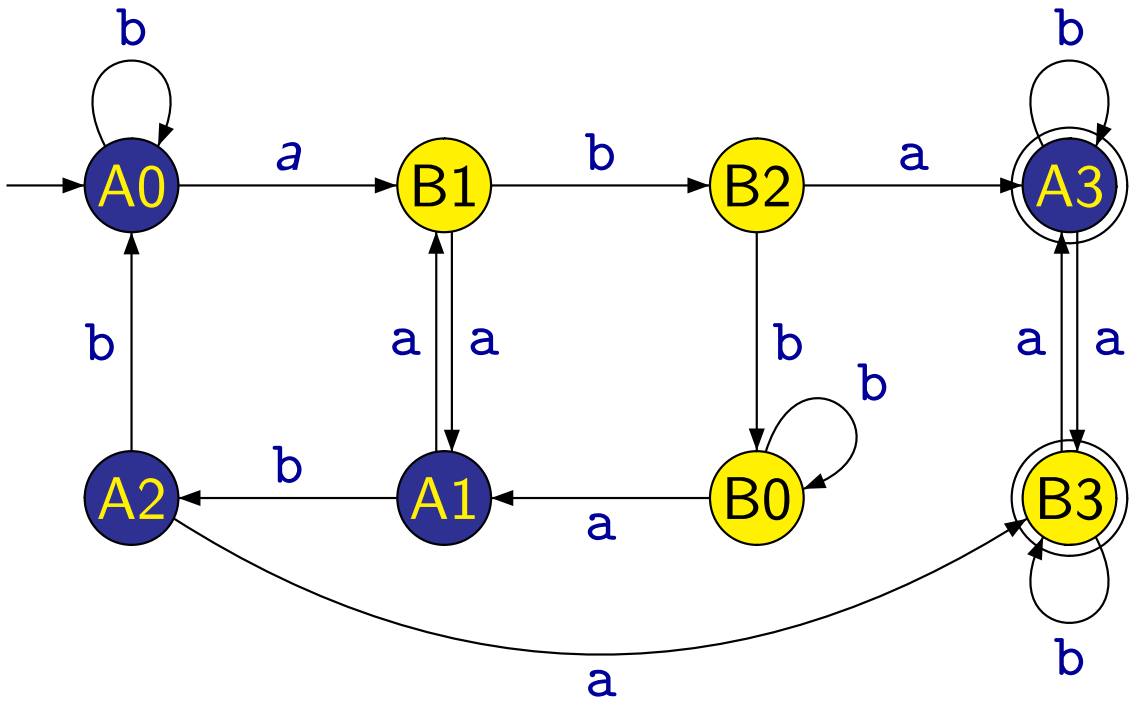




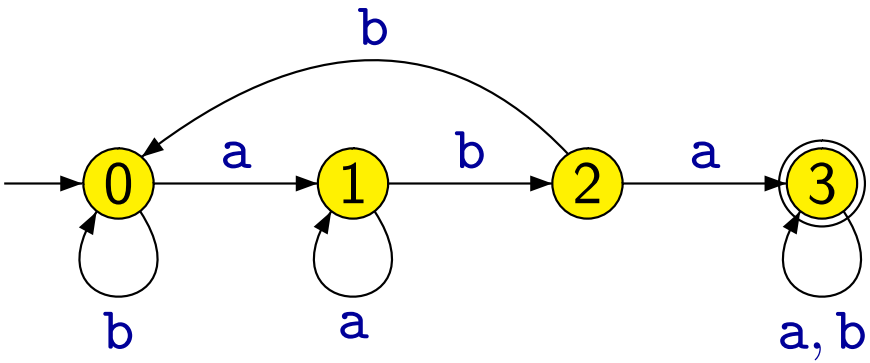
Automat pro sjednocení jazyků

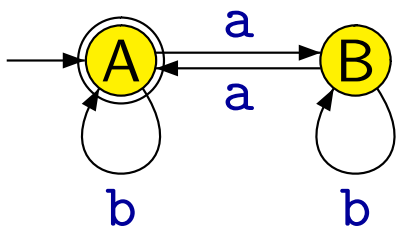


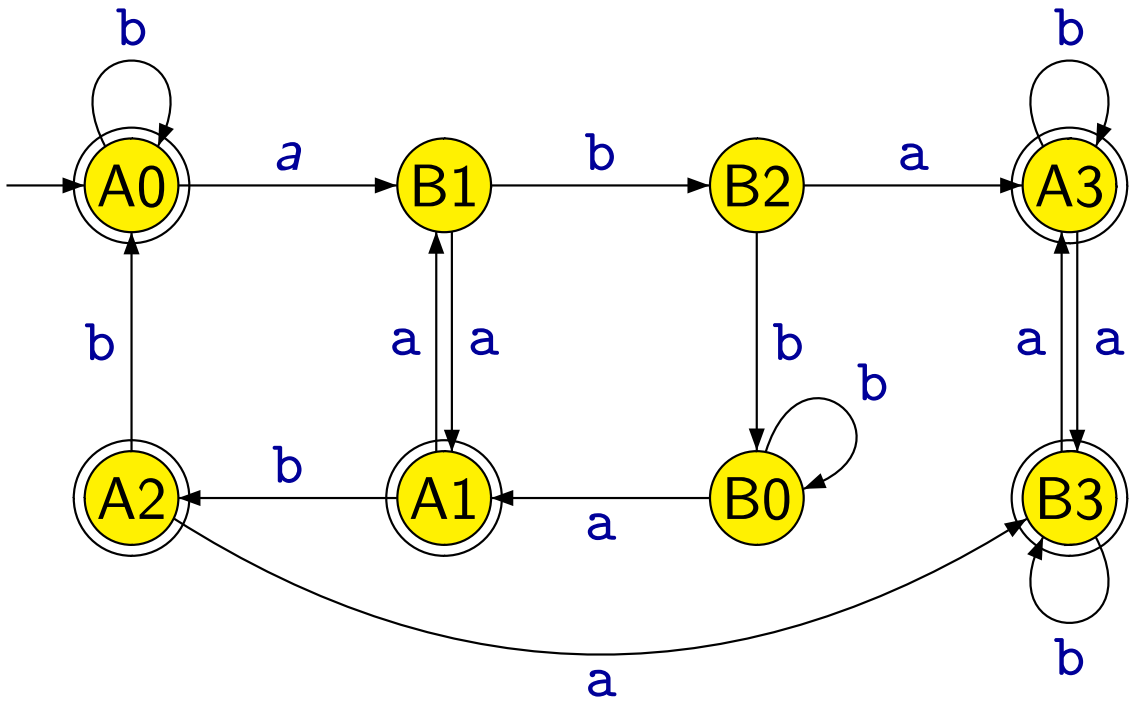




Automat pro sjednocení jazyků







Sjednocení regulárních jazyků

Věta

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

Sjednocení regulárních jazyků

Věta

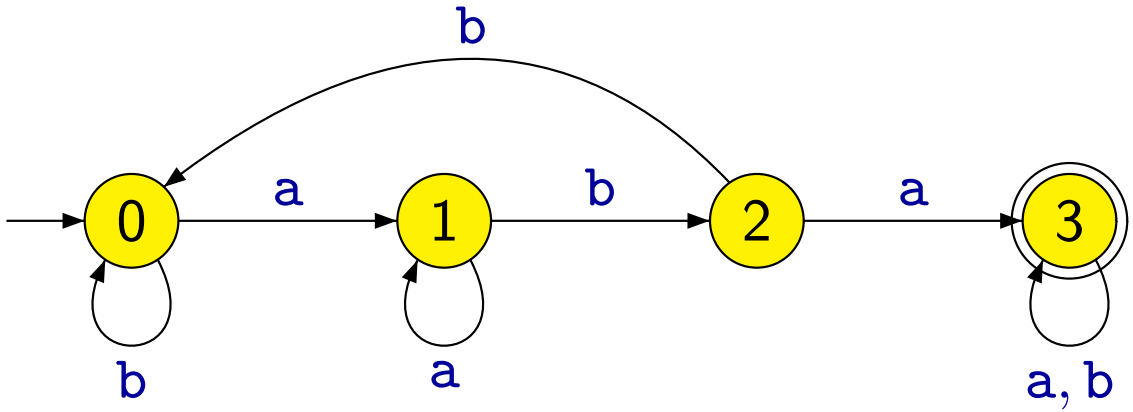
Jestliže jazyky $L_1, L_2 \subseteq \Sigma^*$ jsou regulární, pak také jazyk $L_1 \cup L_2$ je regulární.

Důkaz: Nechť $L_1 = L(\mathcal{A}_1)$, $L_2 = L(\mathcal{A}_2)$ pro konečné automaty $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$. Definujeme automat $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ tž.

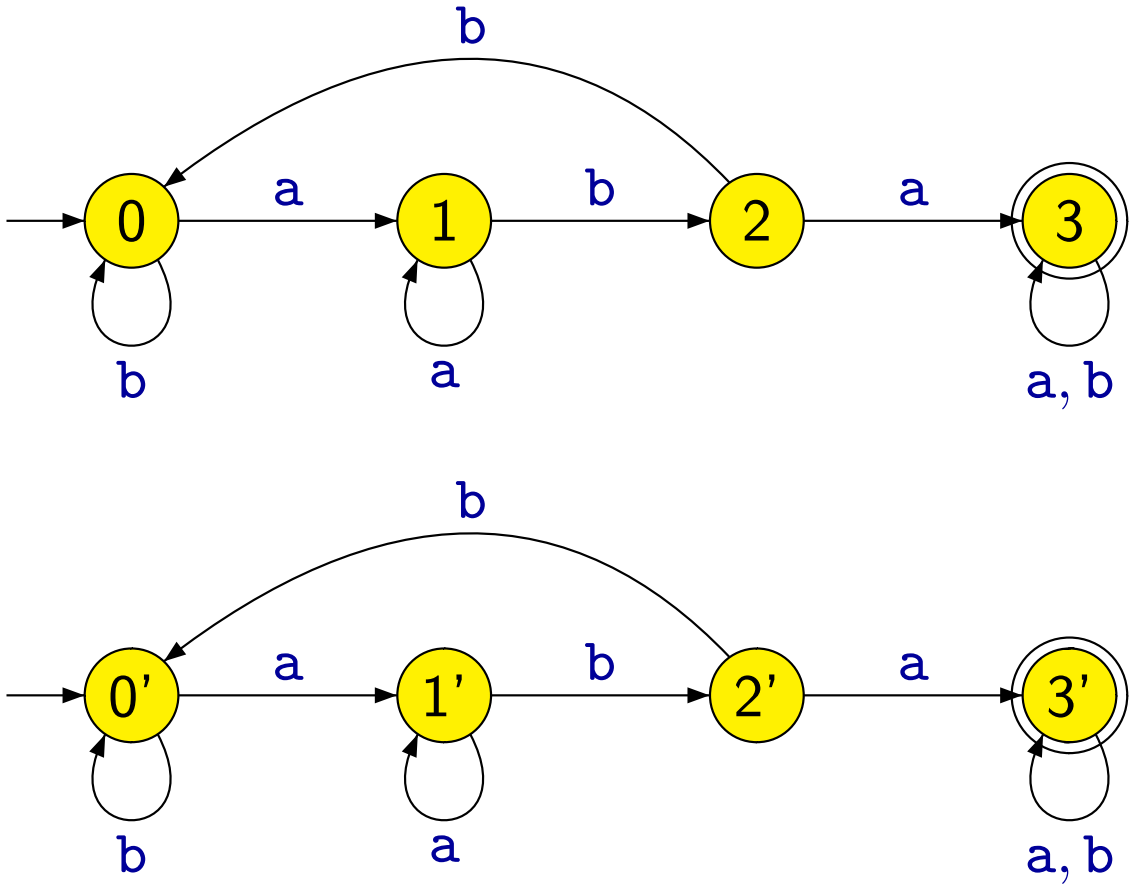
- $Q = Q_1 \times Q_2$,
- $\delta(\,(q_1, q_2), a\,) = (\,\delta_1(q_1, a), \delta_2(q_2, a)\,)$ pro všechna $q_1 \in Q_1$, $q_2 \in Q_2$, $a \in \Sigma$,
- $q_0 = (q_{01}, q_{02})$,
- $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Je zřejmé a např. indukcí podle délky $|w|$ je možno ukázat, že pro libovolné $q_1 \in Q_1$, $q_2 \in Q_2$ a $w \in \Sigma^*$ je $\delta^*(\,(q_1, q_2), w\,) = (\,\delta_1^*(q_1, w), \delta_2^*(q_2, w)\,)$.

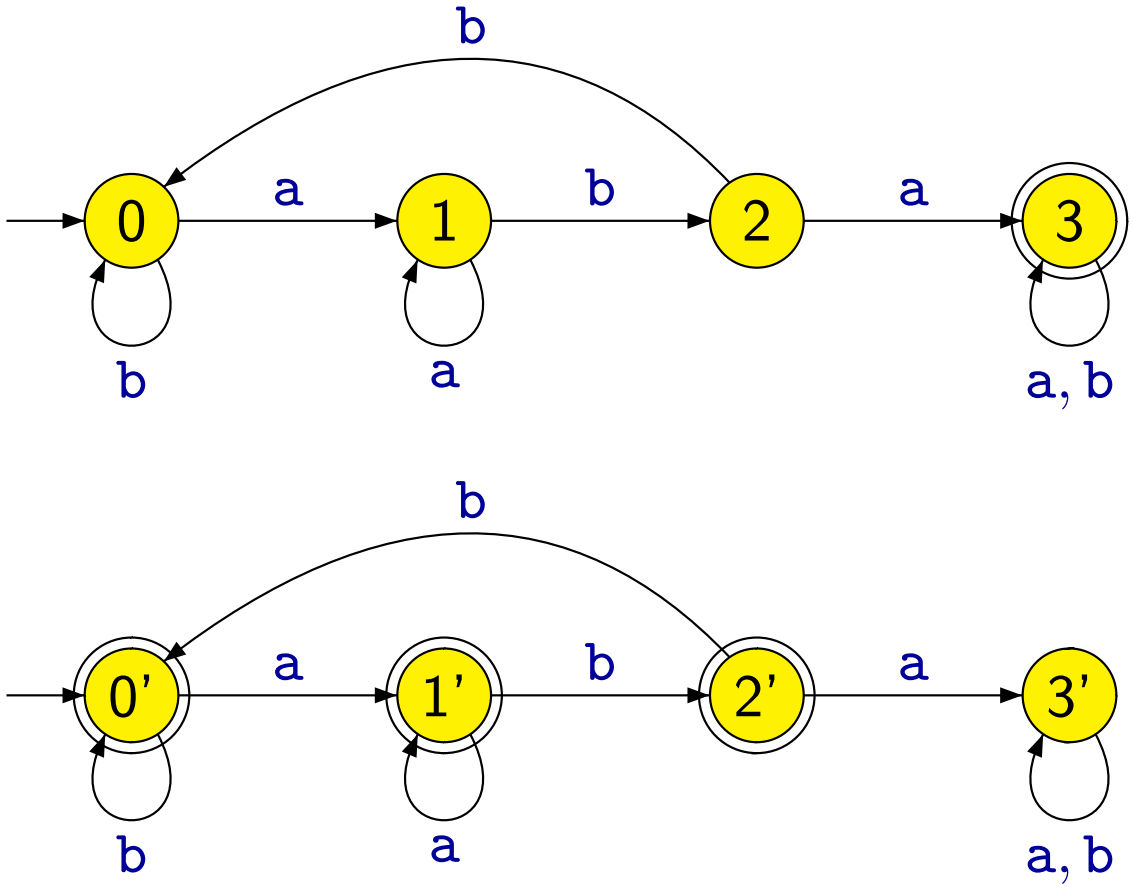
Automat pro doplněk jazyka



Automat pro doplněk jazyka



Automat pro doplněk jazyka



Doplněk regulárního jazyka

Věta

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

Doplněk regulárního jazyka

Věta

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

Důkaz: Nechť $L = L(\mathcal{A})$ pro konečný automat $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$. Definujeme automat $\mathcal{A}' = (Q, \Sigma, \delta, q_0, Q - F)$. Potom

- pro každé slovo w přijímané $\mathcal{A}$ platí $\delta^*(q_0, w) \in F$ a tedy $\delta^*(q_0, w) \notin Q - F$
- pro každé slovo w nepřijímané $\mathcal{A}$ platí $\delta^*(q_0, w) \notin F$ a tedy $\delta^*(q_0, w) \in Q - F$
- a tedy automat $\mathcal{A}'$ přijímá právě ta slova, která nepřijímá $\mathcal{A}$

Zřetězení regulárních jazyků

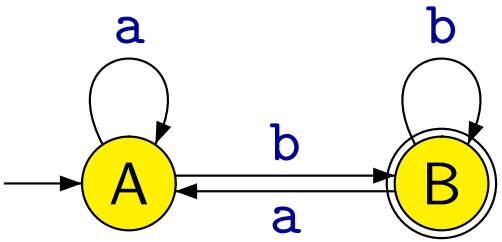
Je možné ke dvěma automatům $\mathcal{A}_\infty, \mathcal{A}_\in$ přijímajícím jazyky L_1, L_2 sestrojit automat $\mathcal{A}$ přijímající jazyk $L_1.L_2$?

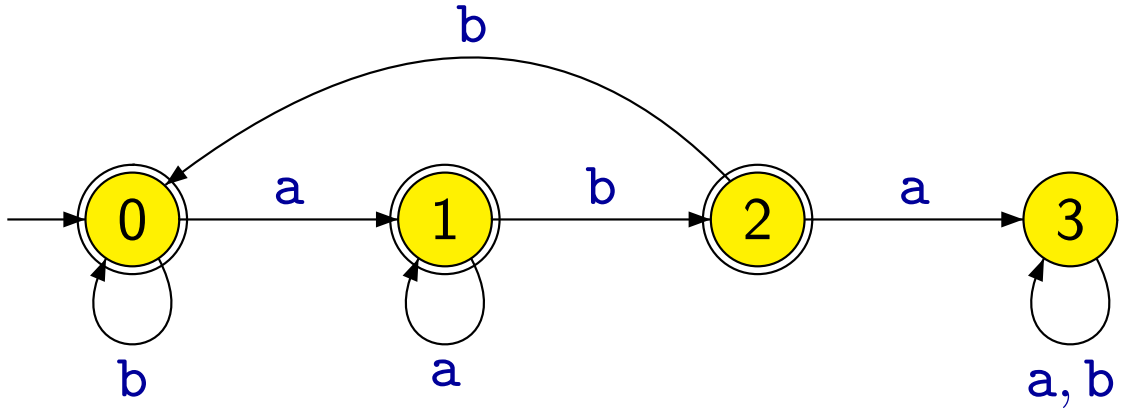
Příklad: Mějme jazyky $L_1 = \{wb \mid w \in \{a, b\}^*\}$ a $L_2 = \{w \in \{a, b\}^* \mid w \text{ neobsahuje } aba\}$.

Zřetězení regulárních jazyků

Je možné ke dvěma automatům $\mathcal{A}_\infty, \mathcal{A}_\in$ přijímajícím jazyky L_1, L_2 sestrojit automat $\mathcal{A}$ přijímající jazyk $L_1.L_2$?

Příklad: Mějme jazyky $L_1 = \{wb \mid w \in \{a,b\}^*\}$ a $L_2 = \{w \in \{a,b\}^* \mid w \text{ neobsahuje } aba\}$.



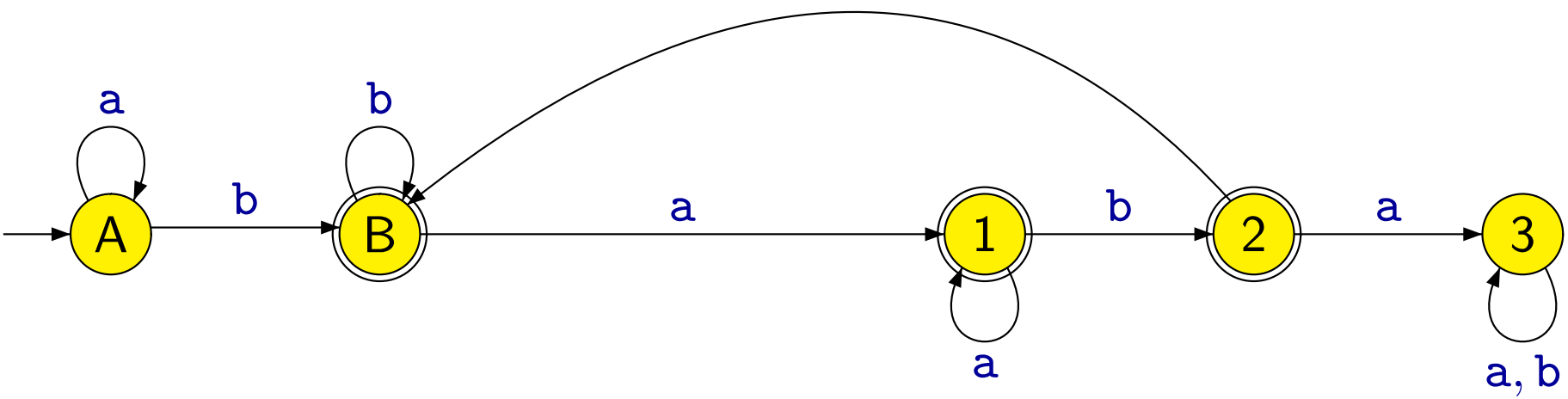


Jak tyto automaty spojit?

Zřetězení regulárních jazyků

Je možné ke dvěma automatům $\mathcal{A}_\infty, \mathcal{A}_\in$ přijímajícím jazyky L_1, L_2 sestrojit automat $\mathcal{A}$ přijímající jazyk $L_1.L_2$?

Příklad: Mějme jazyky $L_1 = \{wb \mid w \in \{a,b\}^*\}$ a $L_2 = \{w \in \{a,b\}^* \mid w \text{ neobsahuje } aba\}$



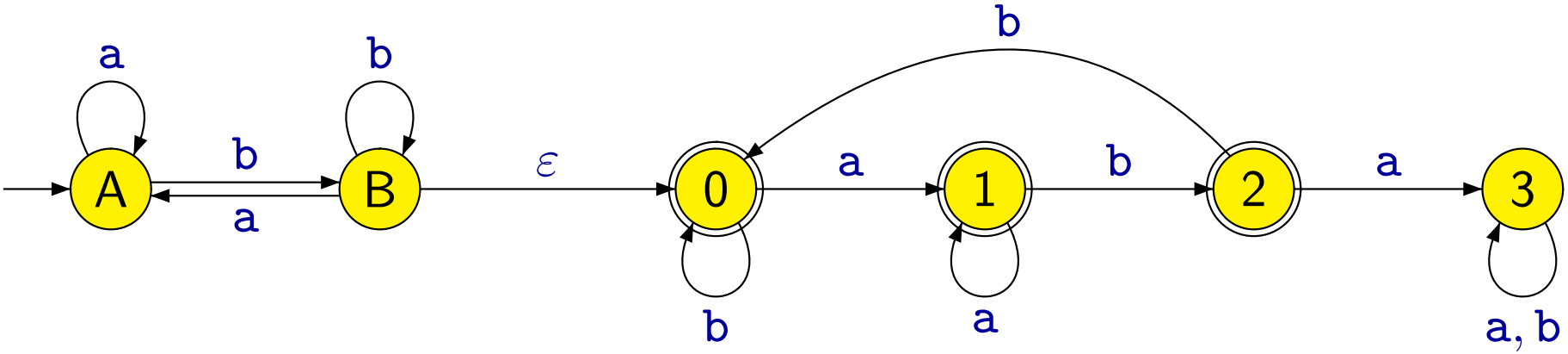
Ve chvíli, kdy přijímá první automat, zkusí začít rozpoznávat druhým automatem.

Správně příjme $abaaa$, kde $ab \in L_1$ a $aaa \in L_2$. Ale nepříjme $ababa$.

Zřetězení regulárních jazyků

Je možné ke dvěma automatům $\mathcal{A}_\infty, \mathcal{A}_\in$ přijímajícím jazyky L_1, L_2 sestrojit automat $\mathcal{A}$ přijímající jazyk $L_1.L_2$?

Příklad: Mějme jazyky $L_1 = \{wb \mid w \in \{a, b\}^*\}$ a $L_2 = \{w \in \{a, b\}^* \mid w \text{ neobsahuje } aba\}$.



Ve chvíli, kdy přijímá první automat, zkusí začít rozpoznávat druhým automatem a současně i zkouší pokračovat v prvním.

Jde o tzv. zobecněný nedeterministický automat rozpoznávající požadovaný jazyk

Nedeterministický konečný automat

Definice

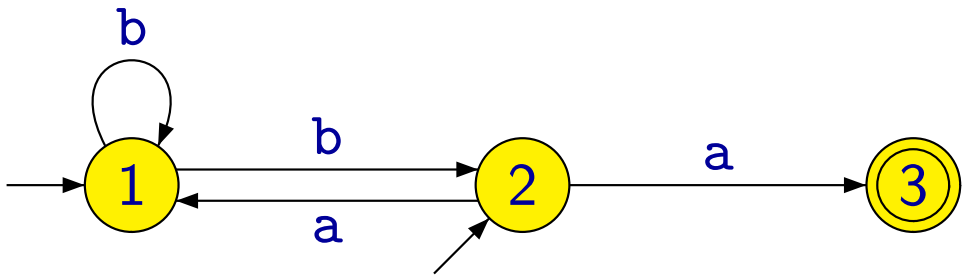
Nedeterministický konečný automat je uspořádaná pětice $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina zvaná vstupní abeceda
- $\delta : Q \times \Sigma \to 2^Q$ je (nedeterministická) přechodová funkce
- $I \subseteq Q$ je neprázdná množina počátečních stavů
- $F \subseteq Q$ je množina přijímajících (koncových) stavů

Rozdíly od deterministického konečného automatu:

- Z daného stavu je možno čtením vstupního symbolu přejít do více různých stavů, ale také přechod pro vstupní symbol nemusí existovat
- Je povolen více než jeden počáteční stav

Nedeterministický konečný automat



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

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

Nedeterministický konečný automat

Místo zápisu

$$\begin{aligned} \delta(1,\mathtt{a}) &= \emptyset & \qquad \delta(1,\mathtt{b}) &= \{1,2\} \\ \delta(2,\mathtt{a}) &= \{1,3\} & \qquad \delta(2,\mathtt{b}) &= \emptyset \\ \delta(3,\mathtt{a}) &= \emptyset & \qquad \delta(3,\mathtt{b}) &= \emptyset \end{aligned}$$

budeme raději používat stručnější tabulku nebo grafické znázornění:

δ	a	b
1		1, 2
2	1, 3	
3		

Nedeterministický konečný automat

Konfigurace nedeterministického konečného automatu je, stejně jako v případě deterministického konečného automatu, dána stavem jeho řídící jednotky a dosud nepřečteným obsahem pásky.

Formálně můžeme konfiguraci definovat jako dvojici z množiny $Q \times \Sigma^*$.

Opět můžeme na množině všech konfigurací definovat binární relaci $\vdash$ se stejným významem: $C_1 \vdash C_2$ znamená, že automat může přejít jedním krokem z konfigurace C_1 do konfigurace C_2. Liší se ovšem formální definice této relace.

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

Nedeterministický konečný automat

Konfigurace (q, w) se nazývá **počáteční konfigurace**, jestliže $q \in I$.

Poznámka: Pro stejné slovo tedy může existovat více různých počátečních konfigurací

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

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 a pro všechna $i \in \{1, 2, \ldots, k\}$ platí, že $C_{i-1} \vdash C_i$.
Výpočet je **úplný**, je-li C_k koncová konfigurace.

Nedeterministický konečný automat

Opět můžeme definovat relaci $\vdash^*$, jejíž význam je takový, 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

$$C_0 \vdash C_1 \vdash C_2 \vdash \cdots \vdash C_k$$

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

Nedeterministický konečný automat

Definice

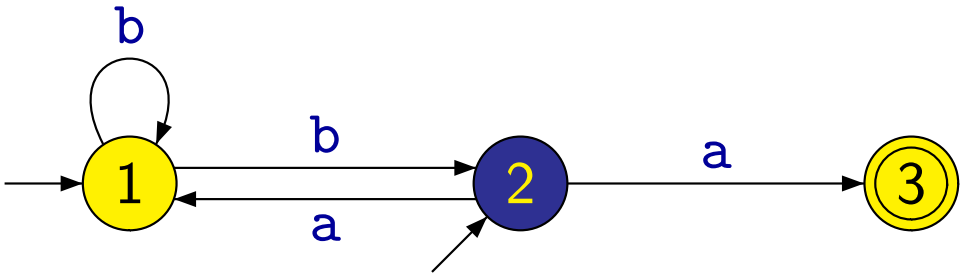
Koncová konfigurace (q, ε) je **přijímající**, jestliže $q \in F$.

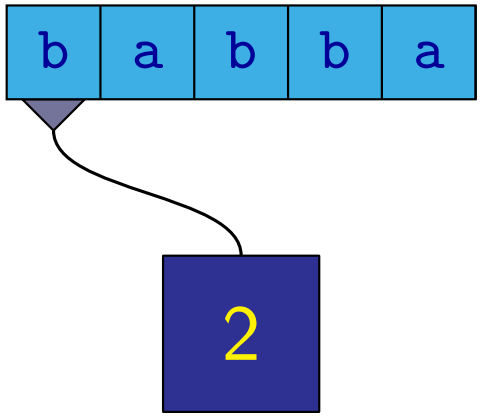
Definice

Automat **přijímá** slovo $w \in \Sigma^*$ právě tehdy, jestliže existuje nějaký (úplný) výpočet začínající v počáteční konfiguraci (q, w) pro nějaké $q \in I$ a končící v přijímající koncové konfiguraci.

Poznámka: Formálně to můžeme definovat tak, že automat přijímá slovo $w \in \Sigma^*$ právě když $(q_0, w) \vdash^* (q, \varepsilon)$ pro nějaké $q \in F$ a $q_0 \in I$.

Nedeterministický konečný automat

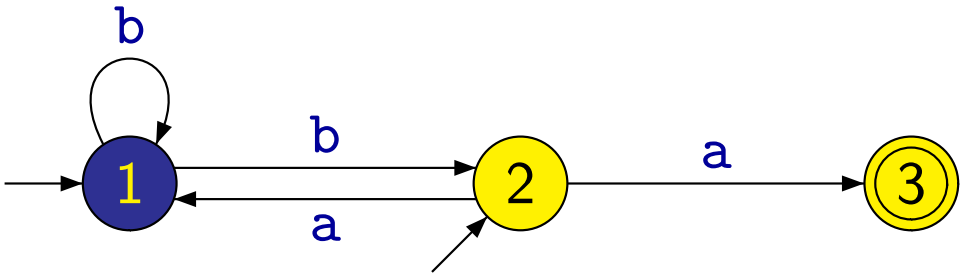


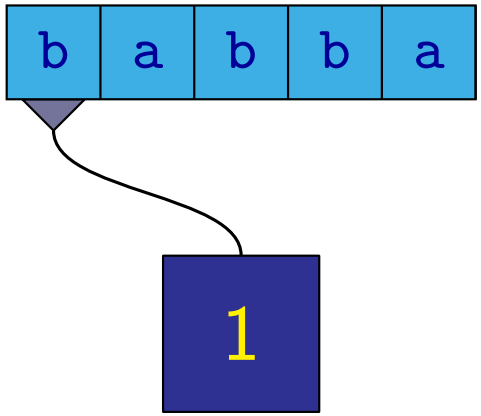


$(2, \mathtt{babba})$

Neúplný výpočet

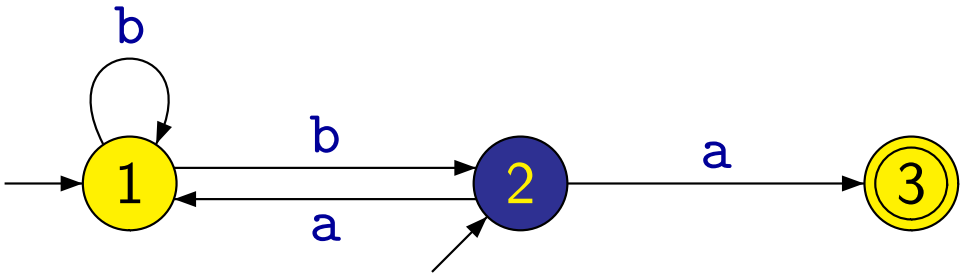
Nedeterministický konečný automat

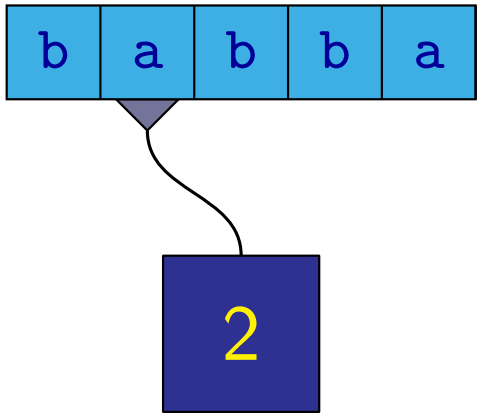




$(1, \mathtt{babba})$

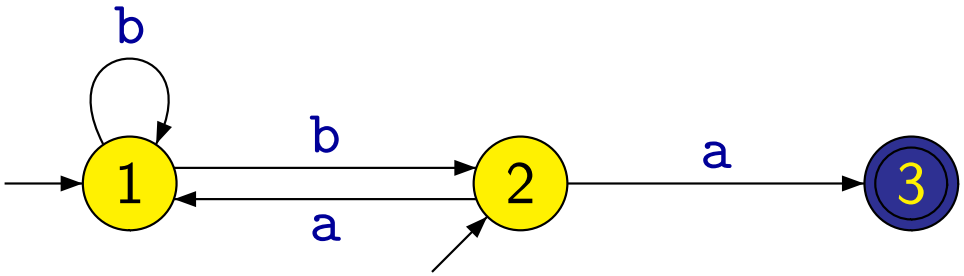
Nedeterministický konečný automat

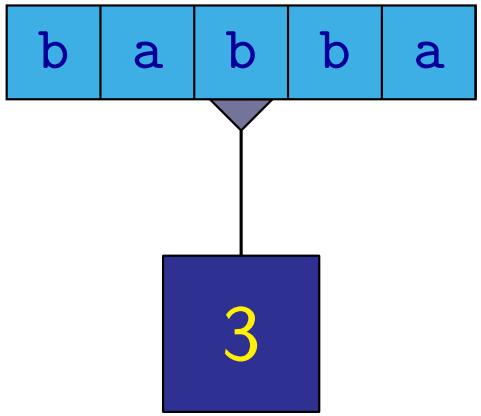




$$(1, \mathtt{babba}) \vdash (2, \mathtt{abba})$$

Nedeterministický konečný automat

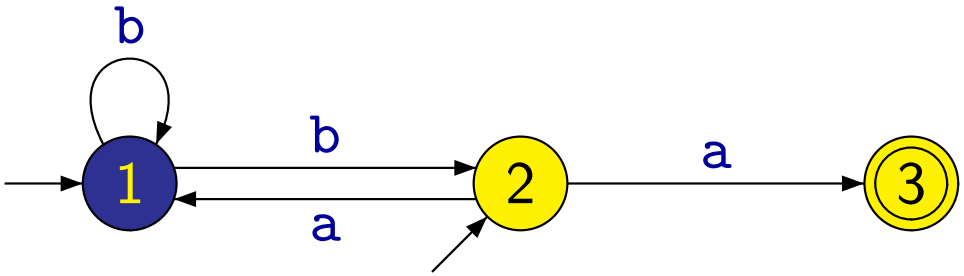


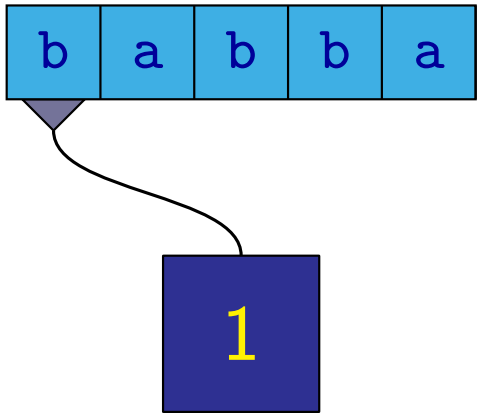


$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash (3, \mathtt{bba})$

Neúplný výpočet

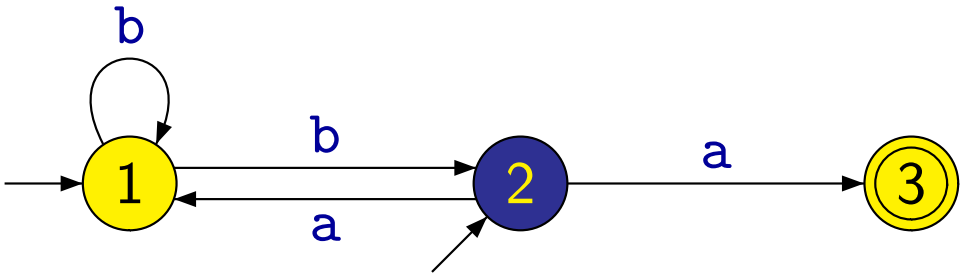
Nedeterministický konečný automat

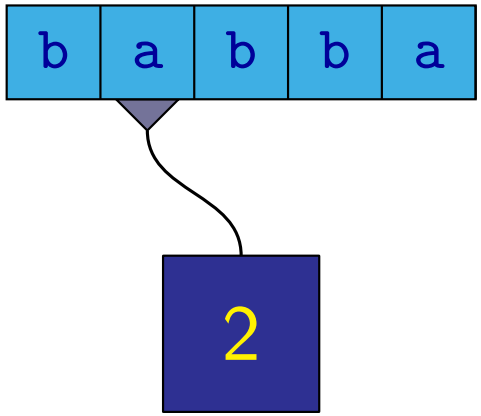




$(1, \mathtt{babba})$

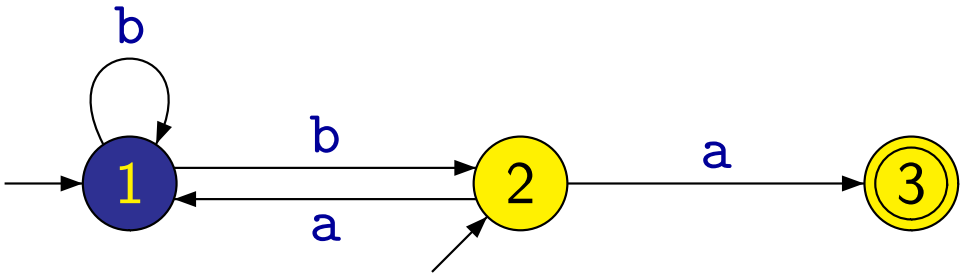
Nedeterministický konečný automat

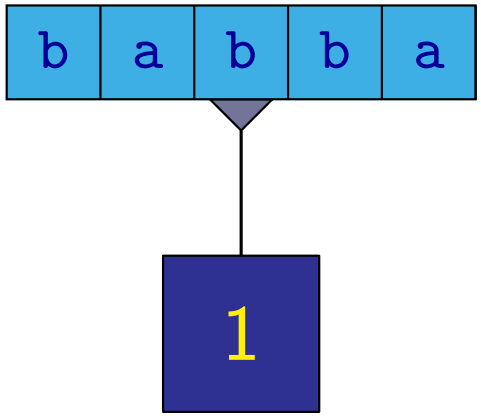




$$(1, \mathtt{babba}) \vdash (2, \mathtt{abba})$$

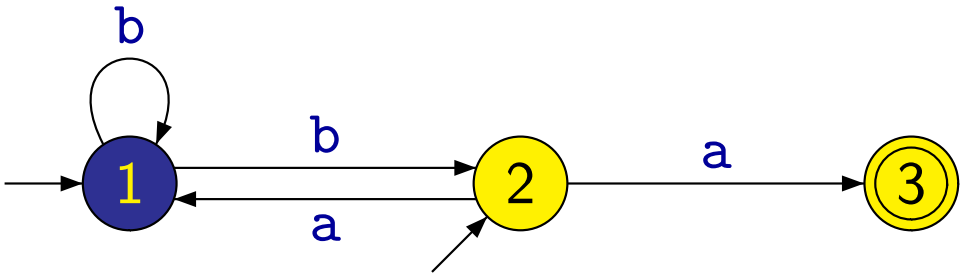
Nedeterministický konečný automat

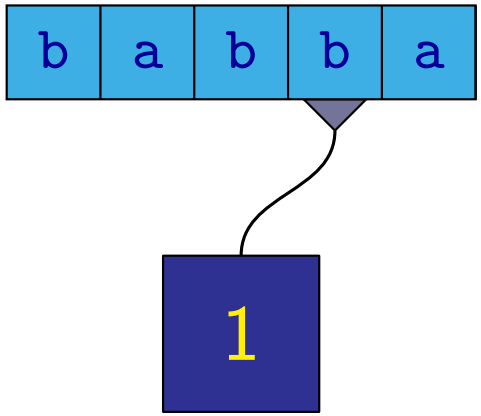




$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash (1, \mathtt{bba})$

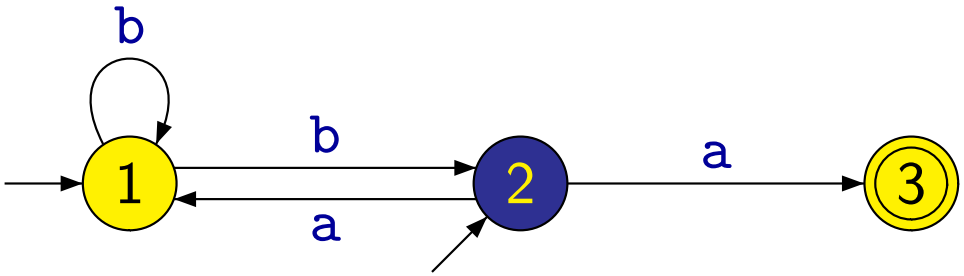
Nedeterministický konečný automat

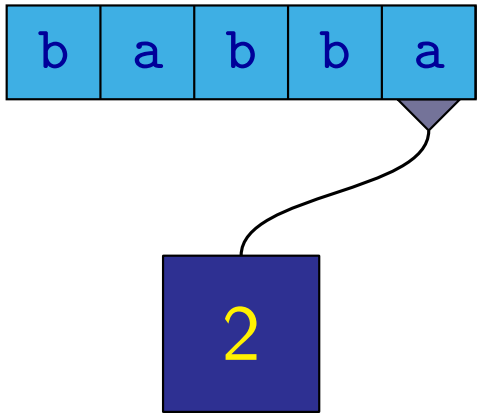




$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash$
$(1, \mathtt{bba}) \vdash (1, \mathtt{ba})$

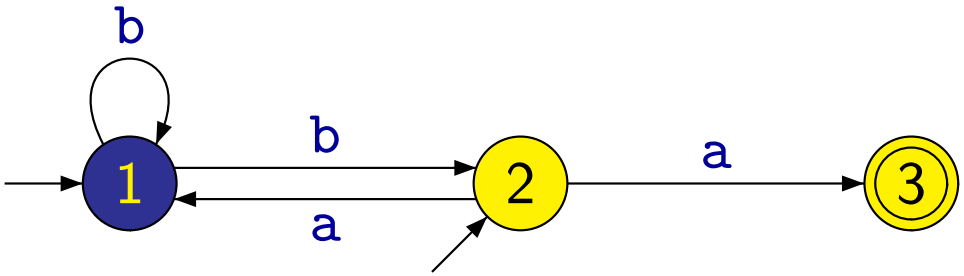
Nedeterministický konečný automat

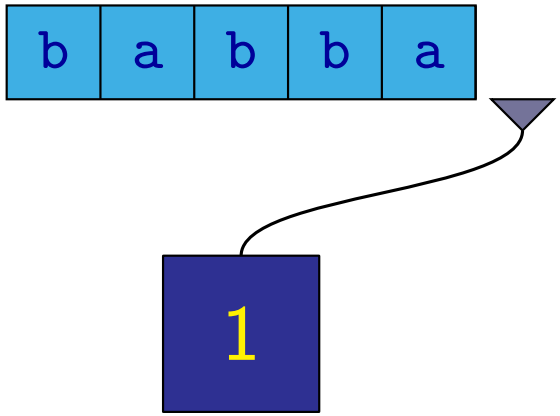




$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash$
$(1, \mathtt{bba}) \vdash (1, \mathtt{ba}) \vdash$
$(2, \mathtt{a})$

Nedeterministický konečný automat

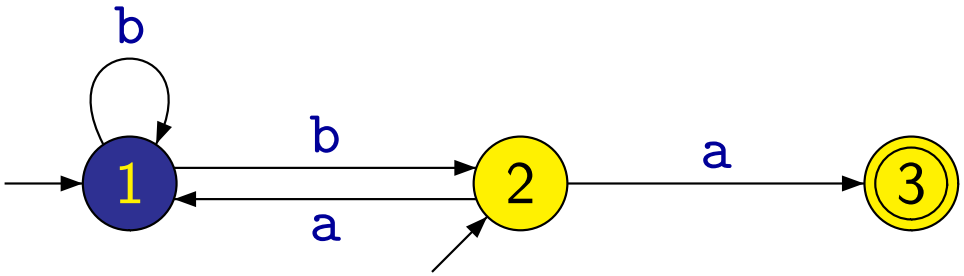


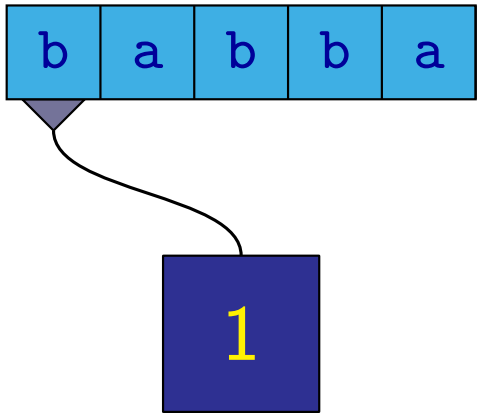


$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash$
$(1, \mathtt{bba}) \vdash (1, \mathtt{ba}) \vdash$
$(2, \mathtt{a}) \vdash (1, \varepsilon)$

Úplný, ale nepřijímající výpočet

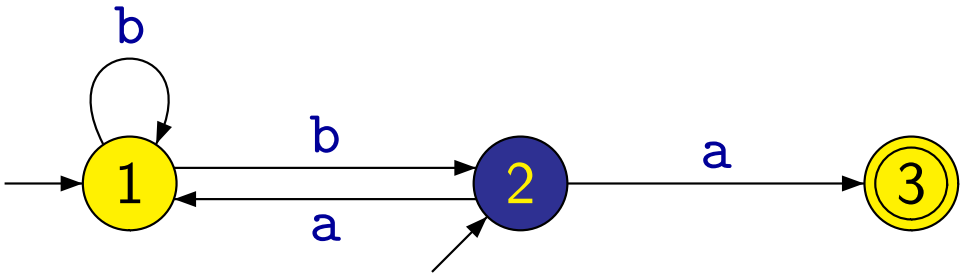
Nedeterministický konečný automat

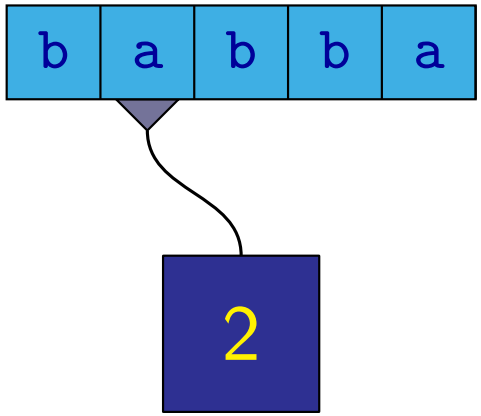




$(1, \mathtt{babba})$

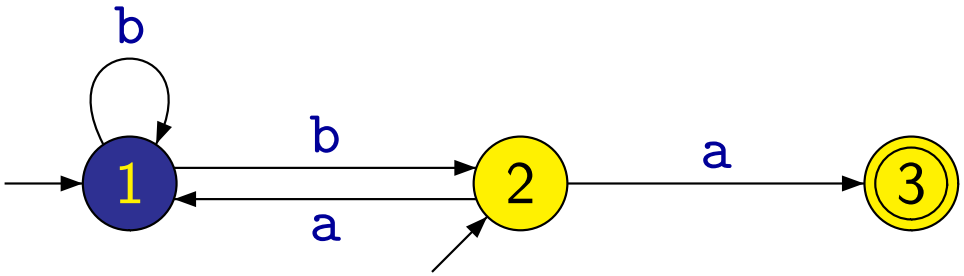
Nedeterministický konečný automat

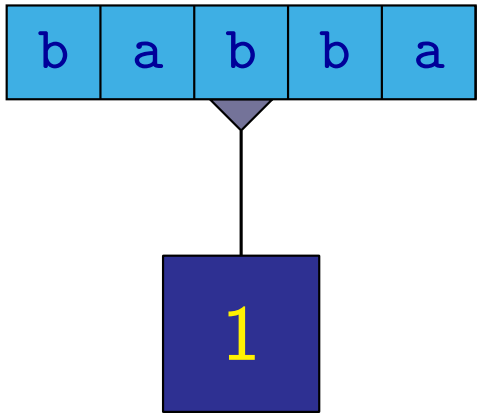




$$(1, \mathtt{babba}) \vdash (2, \mathtt{abba})$$

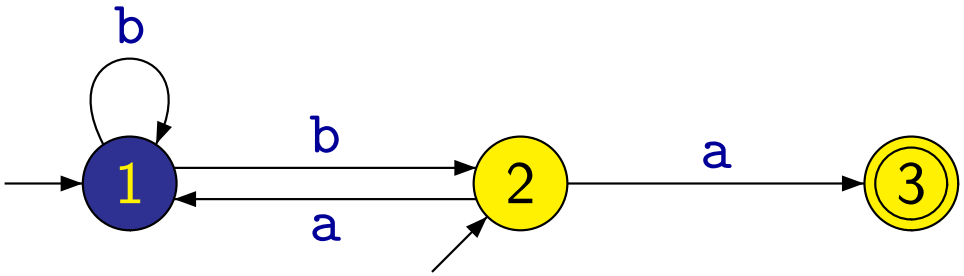
Nedeterministický konečný automat

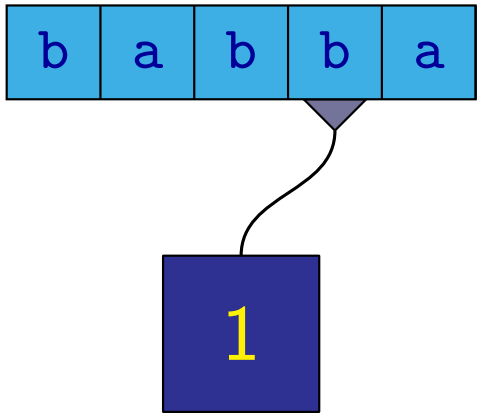




$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash (1, \mathtt{bba})$

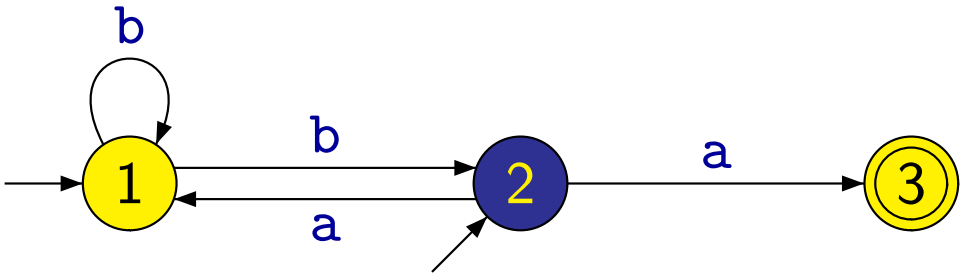
Nedeterministický konečný automat

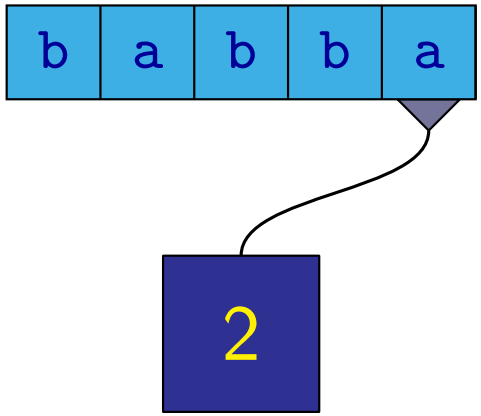




$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash$
$(1, \mathtt{bba}) \vdash (1, \mathtt{ba})$

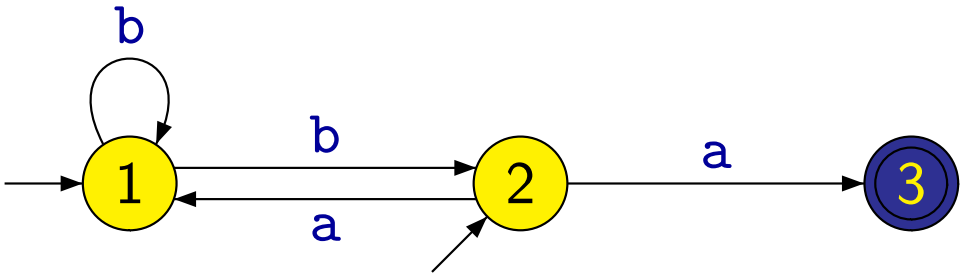
Nedeterministický konečný automat

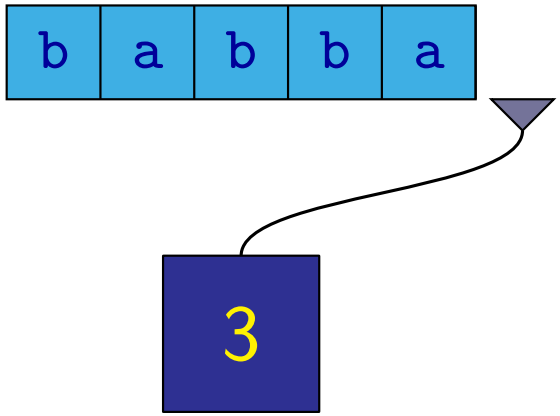




$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash$
$(1, \mathtt{bba}) \vdash (1, \mathtt{ba}) \vdash$
$(2, \mathtt{a})$

Nedeterministický konečný automat





$(1, \mathtt{babba}) \vdash (2, \mathtt{abba}) \vdash$
$(1, \mathtt{bba}) \vdash (1, \mathtt{ba}) \vdash$
$(2, \mathtt{a}) \vdash (3, \varepsilon)$

Úplný přijímající výpočet

Zobecněný nedeterministický konečný automat

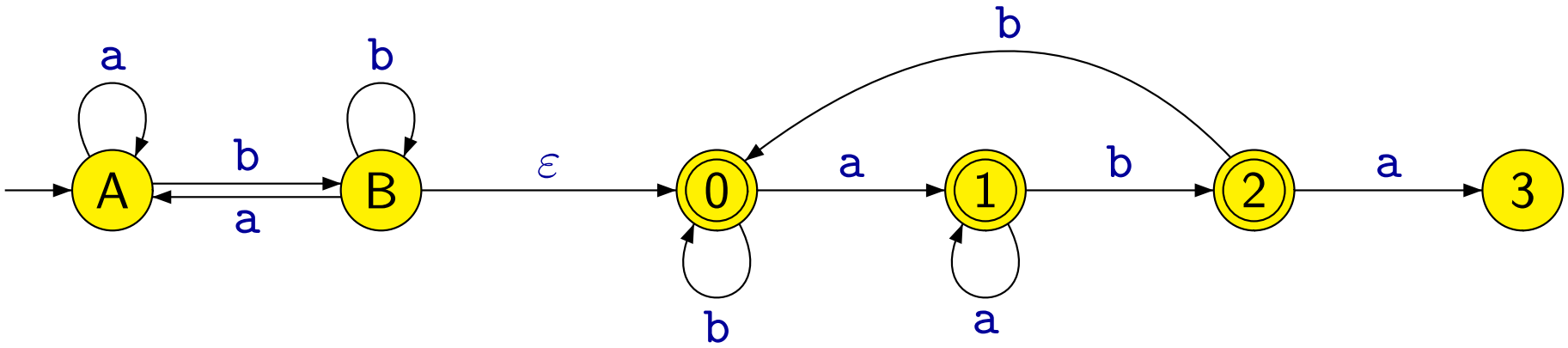
- Někdy je užitečná možnost nechat automat změnit stav bez čtení vstupního symbolu (např. motivační příklad na zřetězení)
- Přidáme tzv. ε-**přechody** - mohou být provedeny kdykoliv se automat nachází ve stavu, ze kterého vycházejí a neposouvá se čtecí hlava na další symbol

Definice

(Zobecněný) nedeterministický konečný automat je uspořádaná pětice $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina zvaná vstupní abeceda
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ je (nedeterministická) přechodová funkce
- $I \subseteq Q$ je neprázdná množina počátečních stavů
- $F \subseteq Q$ je množina přijímajících (koncových) stavů

Zobecněný nedeterministický konečný automat



- $Q = \{A, B, 1, 2, 3\}$
- $\Sigma = \{\mathtt{a}, \mathtt{b}\}$
- $I = \{A\}$
- $F = \{0, 1, 2\}$

$$\begin{array}{ll}
\delta(A, \mathtt{a}) = \{A\} & \delta(A, \mathtt{b}) = \{B\} \\
\delta(B, \mathtt{a}) = \{A\} & \delta(B, \mathtt{b}) = \{B\} \\
\delta(B, \varepsilon) = \{0\} & \\
\delta(0, \mathtt{a}) = \{1\} & \delta(0, \mathtt{b}) = \{0\} \\
\delta(1, \mathtt{a}) = \{1\} & \delta(1, \mathtt{b}) = \{2\} \\
\delta(2, \mathtt{a}) = \{3\} & \delta(2, \mathtt{b}) = \{0\} \\
\delta(3, \mathtt{a}) = \emptyset & \delta(3, \mathtt{b}) = \emptyset
\end{array}$$

Zobecněný nedeterministický konečný automat

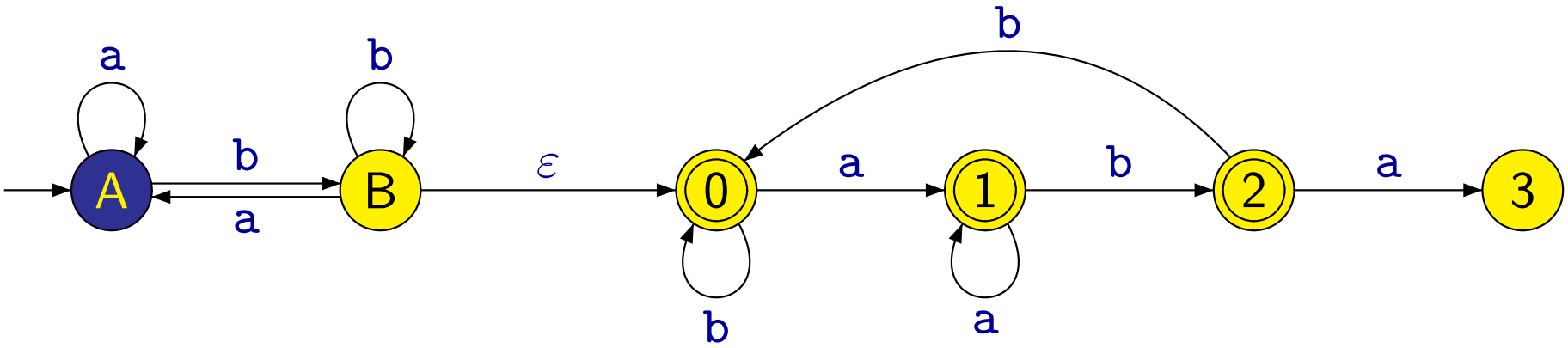
Opět je vhodné zapsat přechodovou funkci tabulkou

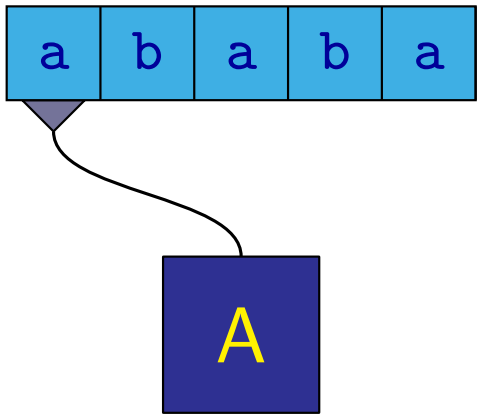
δ	a	b	ε
A	A	B	
B	A	B	0
0	1	0	
1	1	2	
2	3	0	
3			

Zobecněný nedeterministický konečný automat

- Konfigurace ZNKA je definována stejně jako pro NKA.
- Relaci $\vdash$ mezi konfiguracemi definujeme tak, že $(q, w) \vdash (q', w')$ právě když $w = aw'$ a $q' \in \delta(q, a)$ pro nějaké $a \in \Sigma$ nebo $w = w'$ a $q' \in \delta(q, \varepsilon)$.
- Počáteční a koncová konfigurace, výpočet, přijímání atd. jsou definovány stejně jako pro NKA

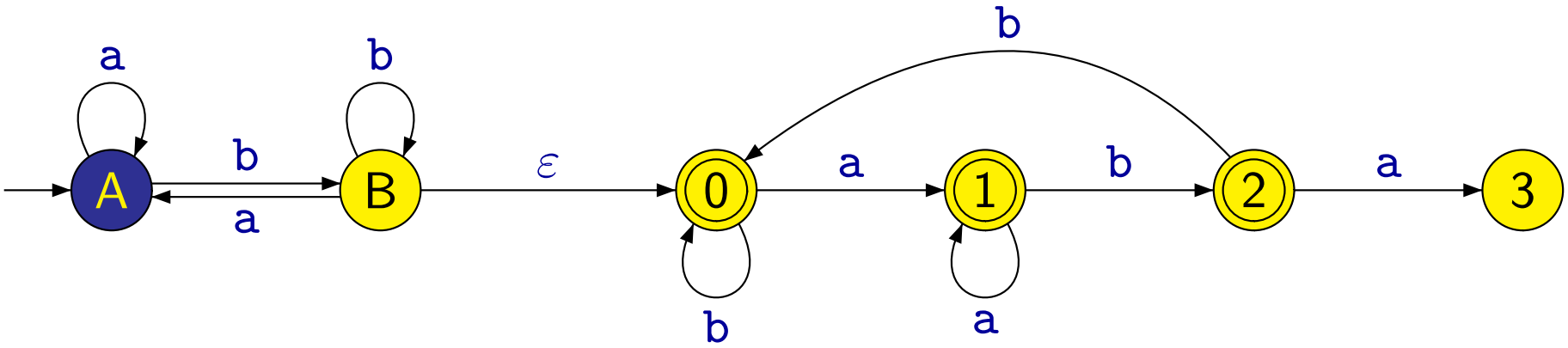
Zobecněný nedeterministický konečný automat

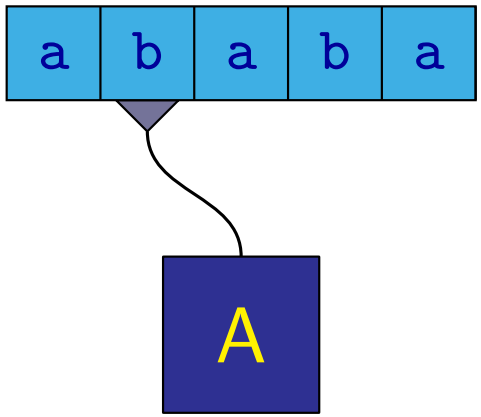




$(A, \mathtt{ababa})$

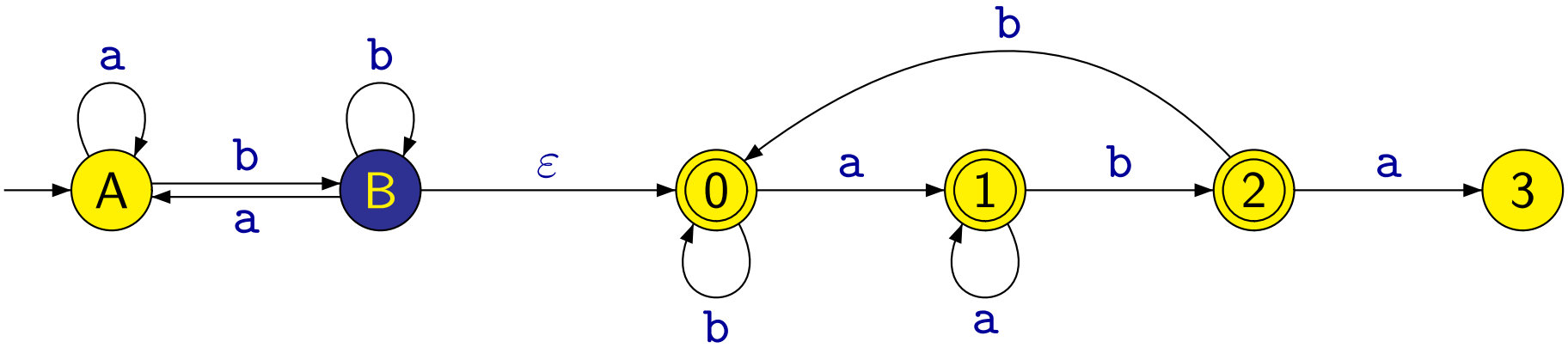
Zobecněný nedeterministický konečný automat

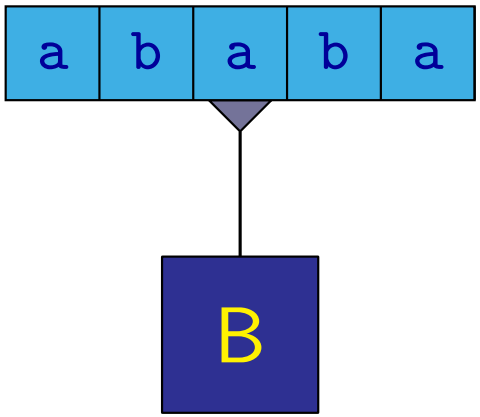




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba})$

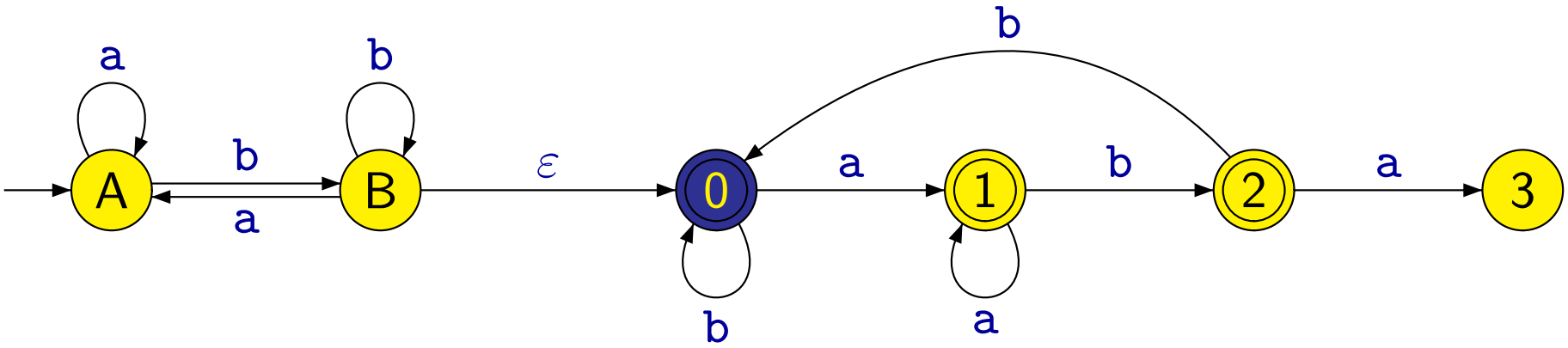
Zobecněný nedeterministický konečný automat

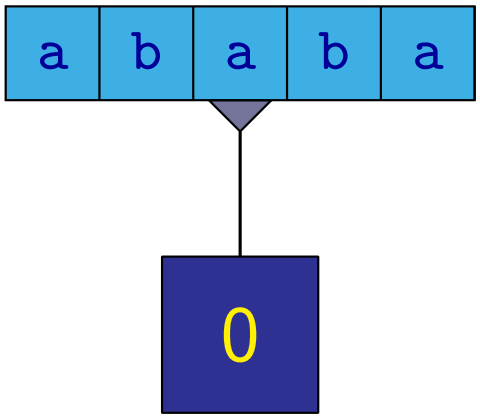




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash (B, \mathtt{aba})$

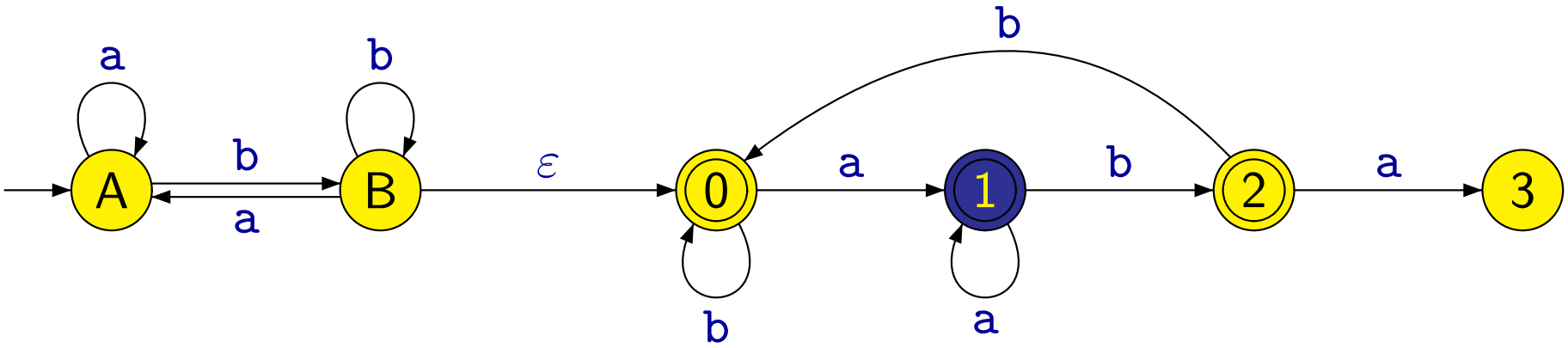
Zobecněný nedeterministický konečný automat

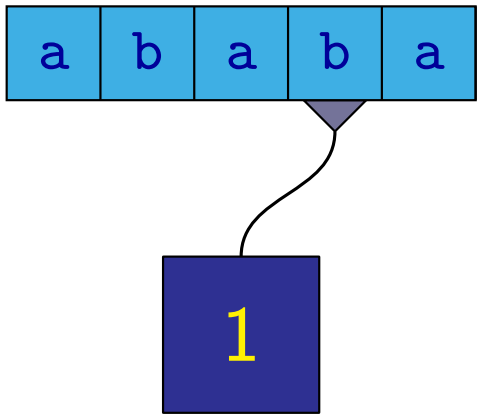




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (0, \mathtt{aba})$

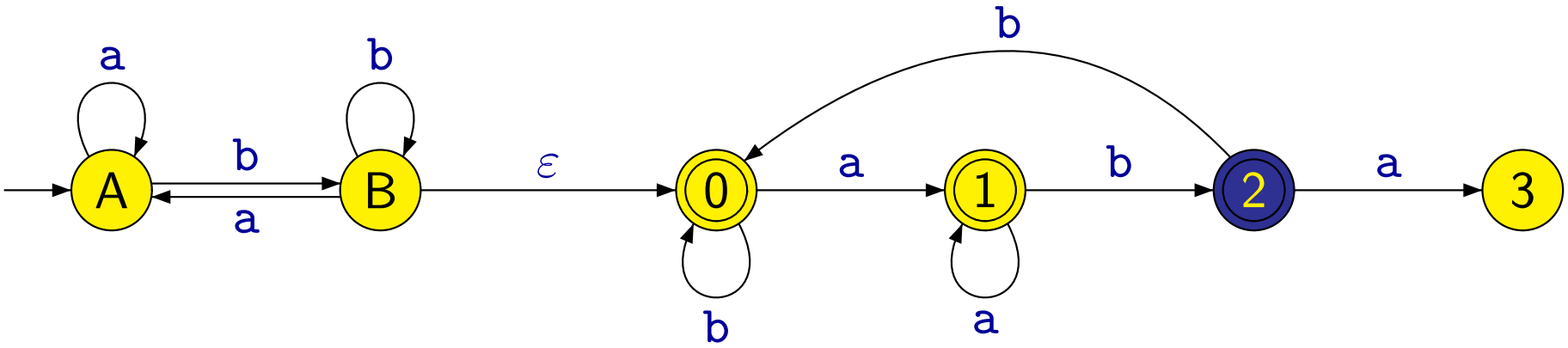
Zobecněný nedeterministický konečný automat

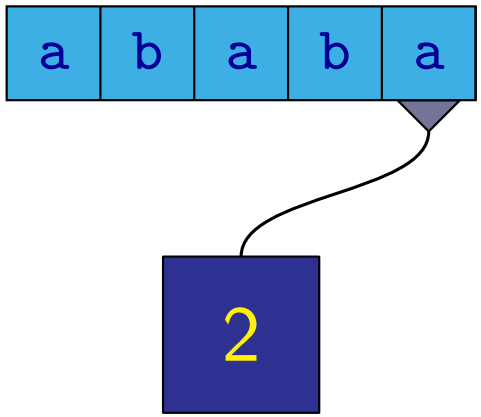




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (0, \mathtt{aba}) \vdash$
$(1, \mathtt{ba})$

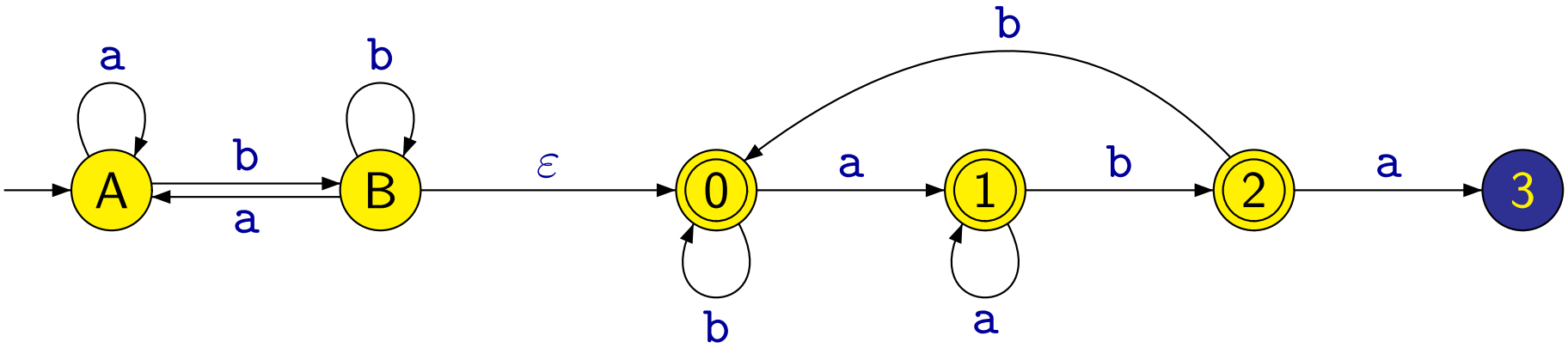
Zobecněný nedeterministický konečný automat

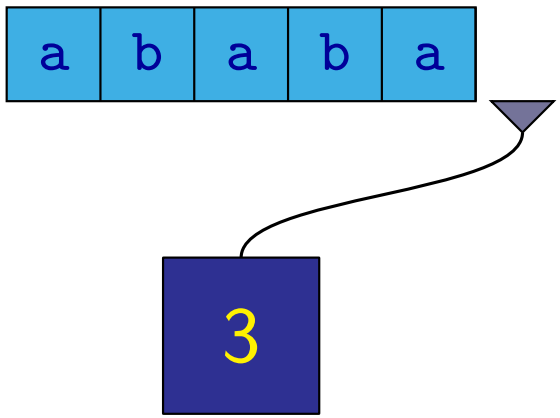




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (0, \mathtt{aba}) \vdash$
$(1, \mathtt{ba}) \vdash (2, \mathtt{a})$

Zobecněný nedeterministický konečný automat

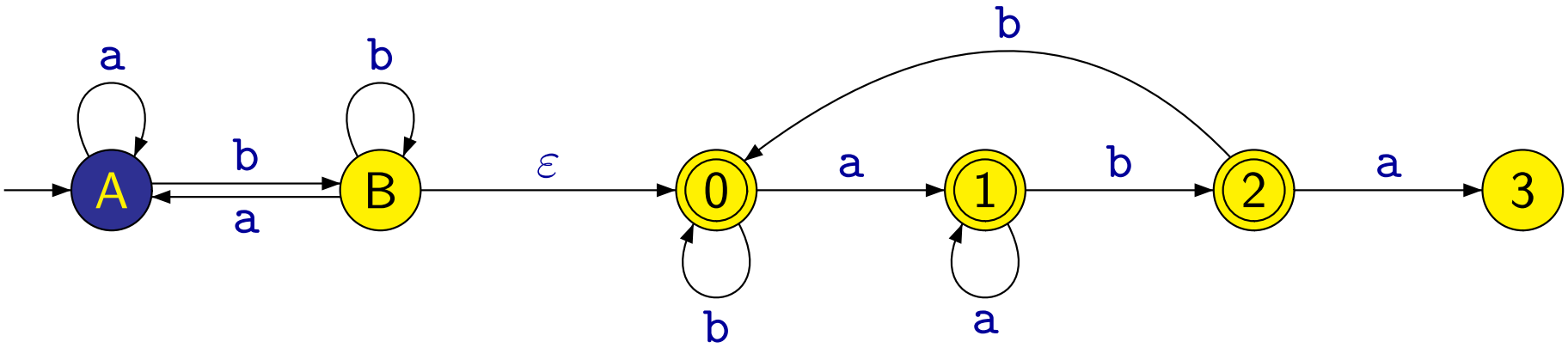


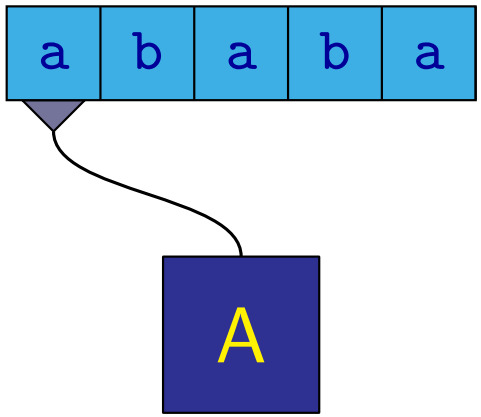


$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (0, \mathtt{aba}) \vdash$
$(1, \mathtt{ba}) \vdash (2, \mathtt{a}) \vdash (3, \varepsilon)$

Úplný, ale nepřijímající výpočet

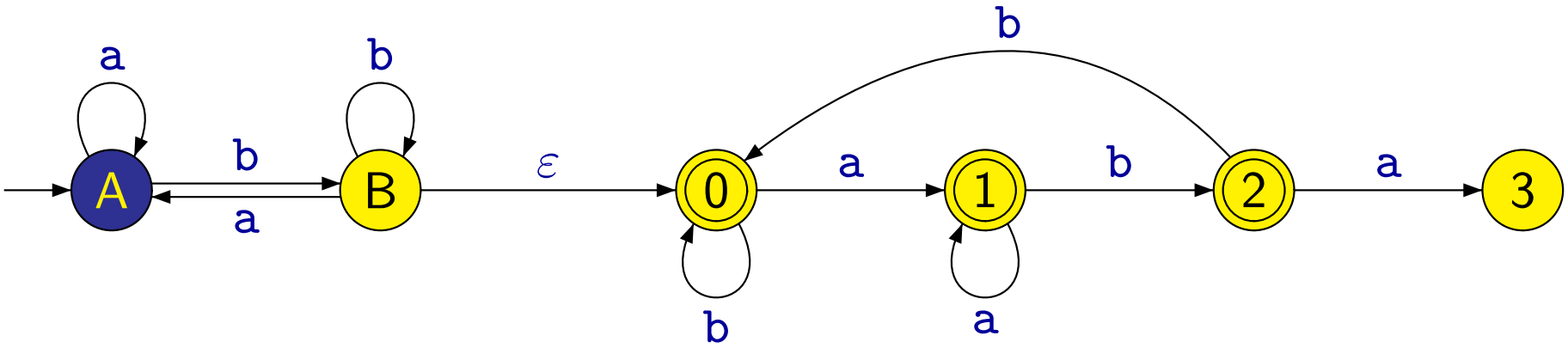
Zobecněný nedeterministický konečný automat

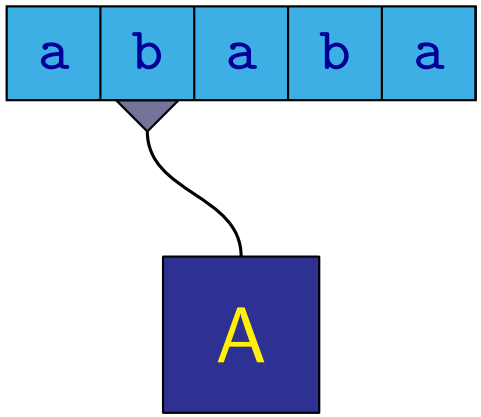




$(A, \mathtt{ababa})$

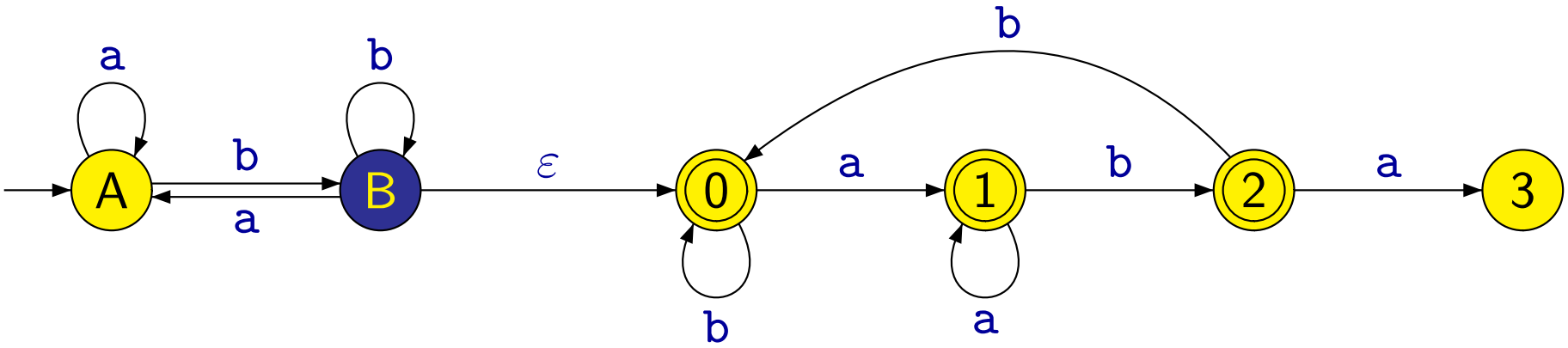
Zobecněný nedeterministický konečný automat

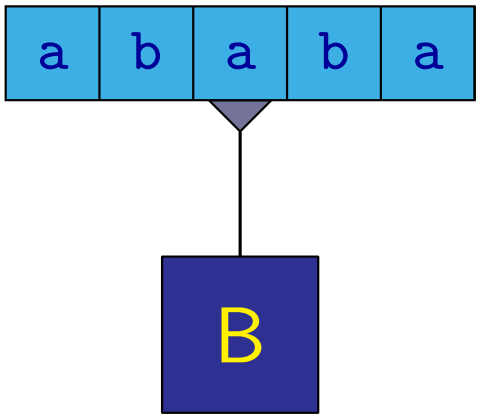




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba})$

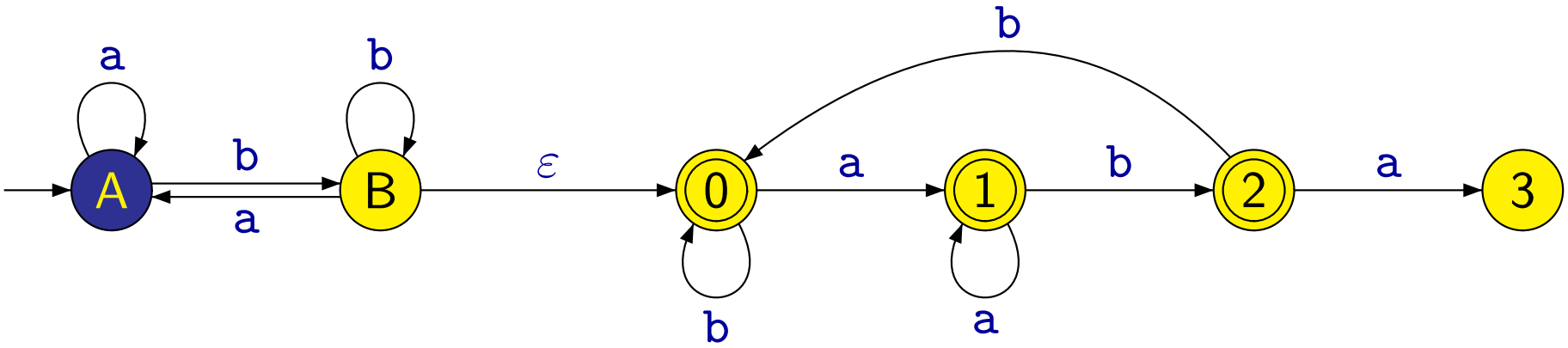
Zobecněný nedeterministický konečný automat

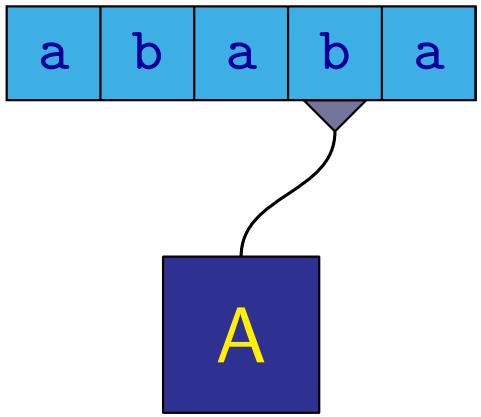




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash (B, \mathtt{aba})$

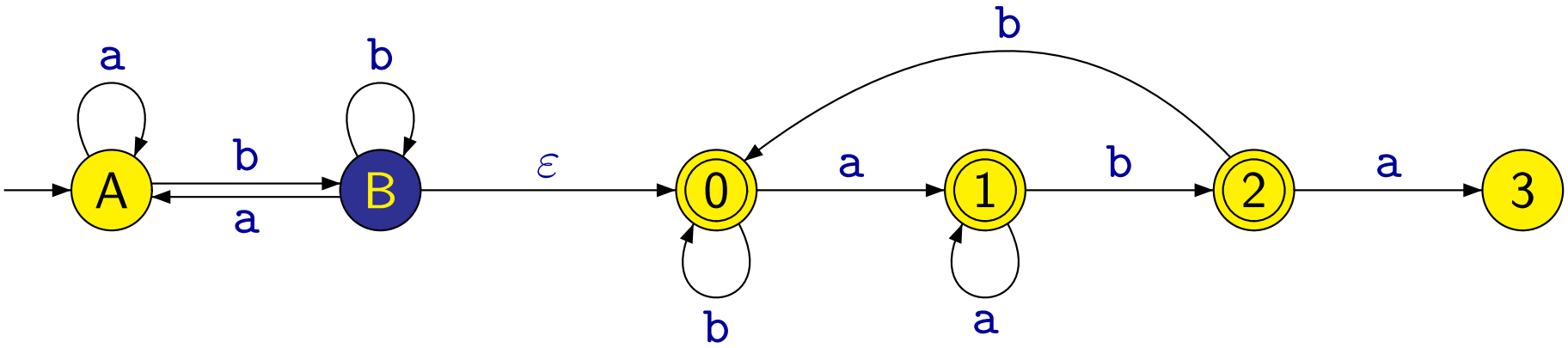
Zobecněný nedeterministický konečný automat

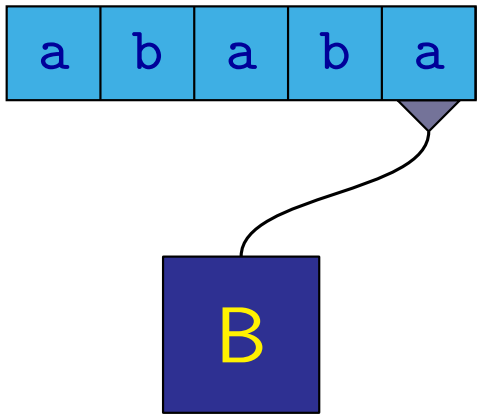




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (A, \mathtt{ba})$

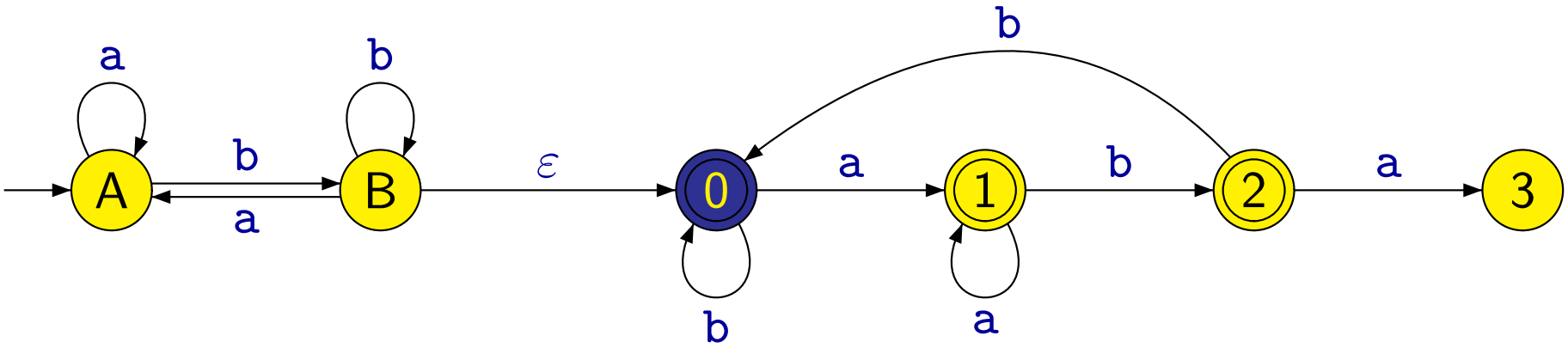
Zobecněný nedeterministický konečný automat

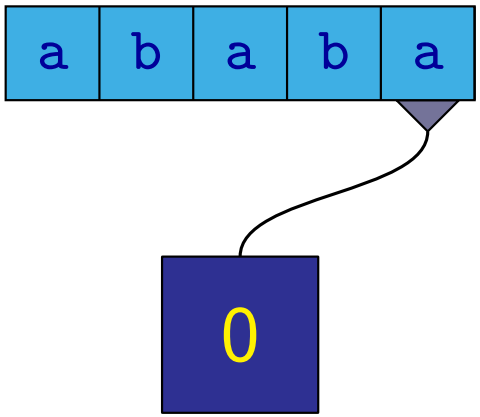




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (A, \mathtt{ba}) \vdash$
$(B, \mathtt{a})$

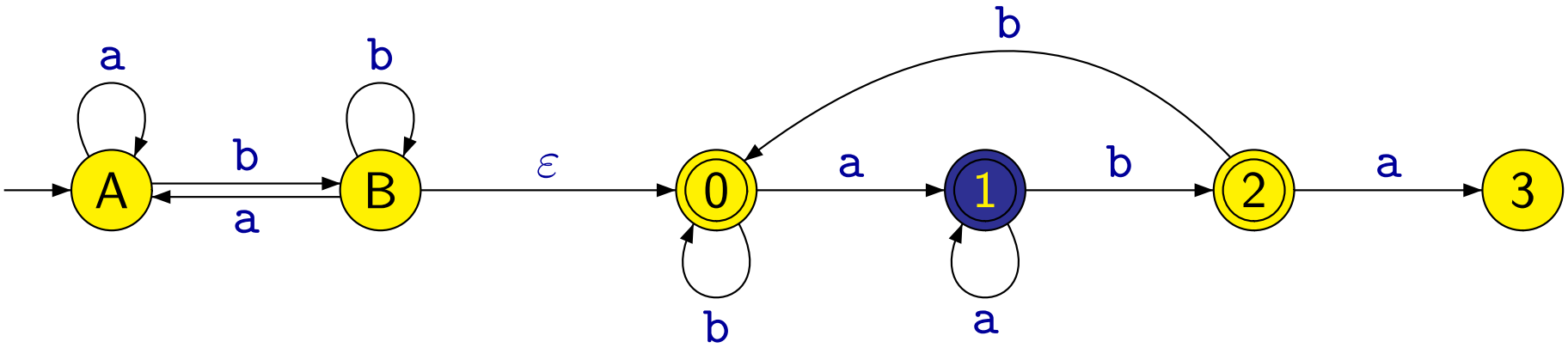
Zobecněný nedeterministický konečný automat

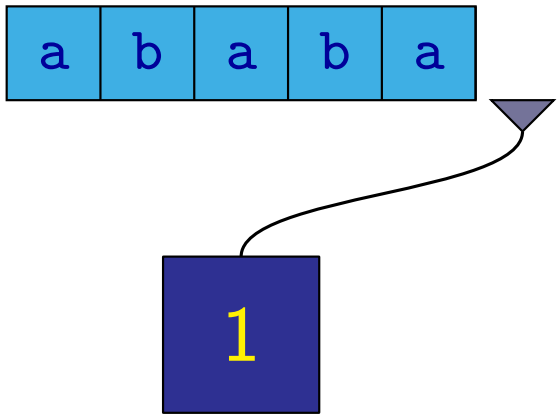




$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (A, \mathtt{ba}) \vdash$
$(B, \mathtt{a}) \vdash (0, \mathtt{a})$

Zobecněný nedeterministický konečný automat





$(A, \mathtt{ababa}) \vdash (A, \mathtt{baba}) \vdash$
$(B, \mathtt{aba}) \vdash (A, \mathtt{ba}) \vdash$
$(B, \mathtt{a}) \vdash (0, \mathtt{a}) \vdash (3, \varepsilon)$

Úplný přijímající výpočet