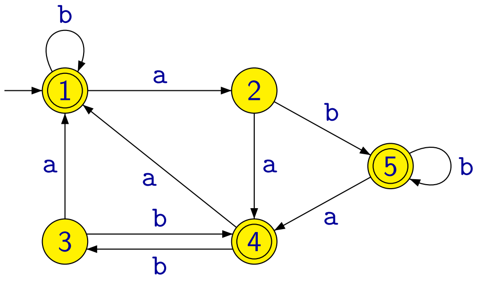
Finite Automata

Deterministic Finite Automaton



A **deterministic finite automaton** consists of **states** and **transitions**. One of the states is denoted as an **initial state** and some of states are denoted as **accepting**.

Deterministic Finite Automaton

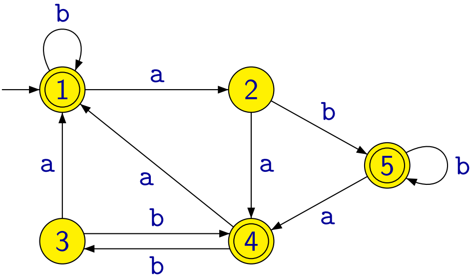
Formally, a **deterministic finite automaton** (**DFA**) is defined as a tuple

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

where:

- Q is a nonempty finite set of **states**
- Σ is an **alphabet** (a nonempty finite set of symbols)
- $\delta : Q \times \Sigma \to Q$ is a **transition function**
- $q_0 \in Q$ is an **initial state**
- $F \subseteq Q$ is a set of **accepting states**

Deterministic Finite Automaton



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

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

Deterministic Finite Automaton

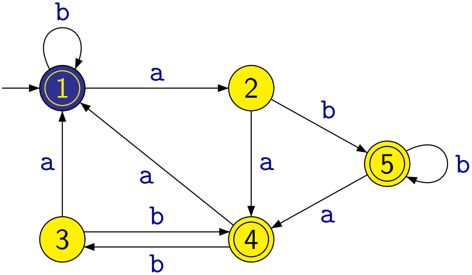
Instead of

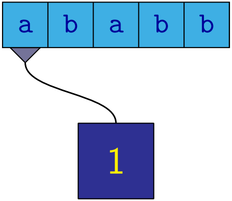
$$\begin{array}{ll} \delta(1,\mathtt{a}) = 2 & \delta(1,\mathtt{b}) = 1 \\ \delta(2,\mathtt{a}) = 4 & \delta(2,\mathtt{b}) = 5 \\ \delta(3,\mathtt{a}) = 1 & \delta(3,\mathtt{b}) = 4 \\ \delta(4,\mathtt{a}) = 1 & \delta(4,\mathtt{b}) = 3 \\ \delta(5,\mathtt{a}) = 4 & \delta(5,\mathtt{b}) = 5 \end{array}$$

we rather use a more succinct representation as a table or a depicted graph:

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

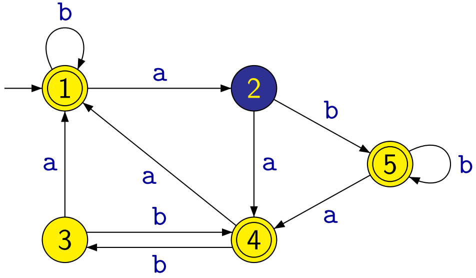
Deterministic Finite Automaton

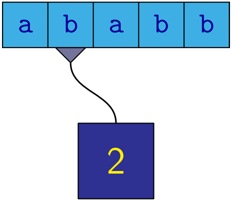




1

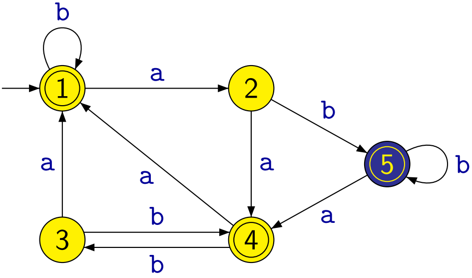
Deterministic Finite Automaton

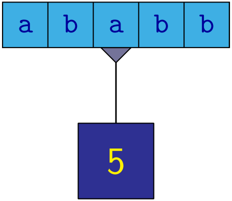




$$1 \xrightarrow{a} 2$$

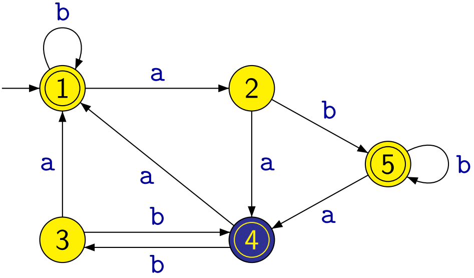
Deterministic Finite Automaton

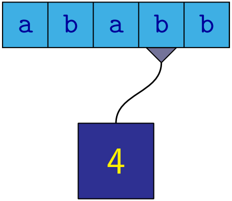




$$1 \xrightarrow{a} 2 \xrightarrow{b} 5$$

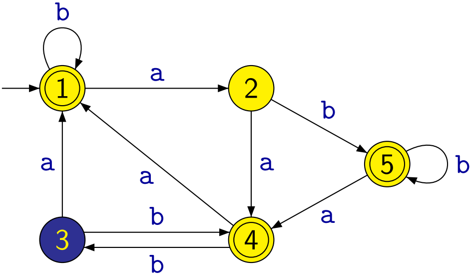
Deterministic Finite Automaton

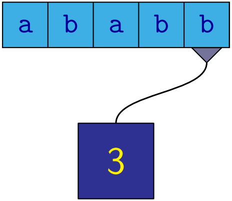




$$1 \xrightarrow{\texttt{a}} 2 \xrightarrow{\texttt{b}} 5 \xrightarrow{\texttt{a}} 4$$

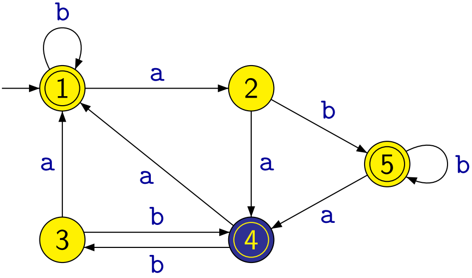
Deterministic Finite Automaton

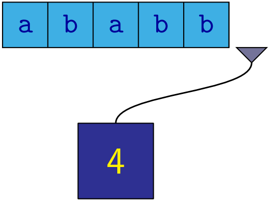




$$1 \xrightarrow{\texttt{a}} 2 \xrightarrow{\texttt{b}} 5 \xrightarrow{\texttt{a}} 4 \xrightarrow{\texttt{b}} 3$$

Deterministic Finite Automaton





$1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{a} 4 \xrightarrow{b} 3 \xrightarrow{b} 4$

Deterministic Finite Automaton

Definition

Let us have a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$.

By $q \xrightarrow{w} q'$, where $q, q' \in Q$ and $w \in \Sigma^*$, we denote the fact that the automaton, starting in state q goes to state q' by reading word w.

Remark: $\longrightarrow \subseteq Q \times \Sigma^* \times Q$ is a ternary relation.

Instead of $(q, w, q') \in \longrightarrow$ we write $q \xrightarrow{w} q'$.

It holds for a DFA that for each state q and each word w there is exactly one state q' such that $q \xrightarrow{w} q'$.

Deterministic Finite Automaton

Relation $\longrightarrow$ can be formally defined by the following inductive definition:

- $q \xrightarrow{\varepsilon} q$ for each $q \in Q$
- For $w \in \Sigma^*$ and $a \in \Sigma$:

  $q \xrightarrow{wa} q'$ iff there is $q'' \in Q$ such that

  $$q \xrightarrow{w} q'' \text{ and } \delta(q'', a) = q'$$

Deterministic Finite Automaton

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

$1 \xrightarrow{\varepsilon} 1$ $\quad$ $\delta(1, \mathtt{a}) = 2$

$1 \xrightarrow{\mathtt{a}} 2$ $\quad$ $\delta(2, \mathtt{b}) = 5$

$1 \xrightarrow{\mathtt{ab}} 5$ $\quad$ $\delta(5, \mathtt{a}) = 4$

$1 \xrightarrow{\mathtt{aba}} 4$ $\quad$ $\delta(4, \mathtt{b}) = 3$

$1 \xrightarrow{\mathtt{abab}} 3$ $\quad$ $\delta(3, \mathtt{b}) = 4$

$1 \xrightarrow{\mathtt{ababb}} 4$

Deterministic Finite Automaton

A word $w \in \Sigma^*$ is **accepted** by a deterministic finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ iff there exists a state $q \in F$ such that $q_0 \xrightarrow{w} q$.

Definition

A **language** accepted by a given deterministic finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, denoted $\mathcal{L}(\mathcal{A})$, is the set of all words accepted by the automaton, i.e.,

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

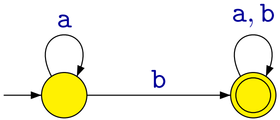
Regular languages

Definition

A language L is **regular** iff there exists some deterministic finite automaton accepting L, i.e., DFA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$.

Example: An automaton recognizing the language L over alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of those words that contain at least one occurrence of symbol $\mathtt{b}$, i.e.,

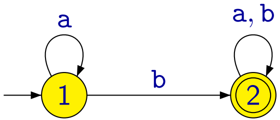
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} \geq 1\}$$



Examples of Deterministic Finite Automata

Example: An automaton recognizing the language L over alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of those words that contain at least one occurrence of symbol $\mathtt{b}$, i.e.,

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} \geq 1\}$$

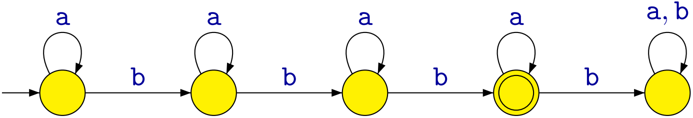


	a	b
$\rightarrow 1$	1	2
$\leftarrow 2$	2	2

Examples of Deterministic Finite Automata

Example: An automaton recognizing the language L over alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of those words that contain exactly three occurrences of symbol b, i.e.,

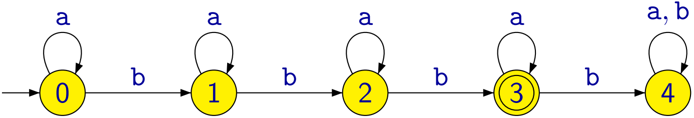
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} = 3\}$$



Examples of Deterministic Finite Automata

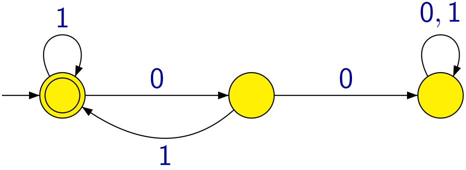
Example: An automaton recognizing the language L over alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of those words that contain exactly three occurrences of symbol b, i.e.,

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} = 3\}$$



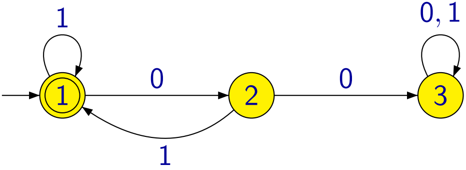
	a	b
$\rightarrow$ 0	0	1
1	1	2
2	2	3
$\leftarrow$ 3	3	4
4	4	4

Example: An automaton recognizing the language over alphabet $\{0, 1\}$ consisting of those words where every occurrence of symbol 0 is immediately followed with symbol 1.



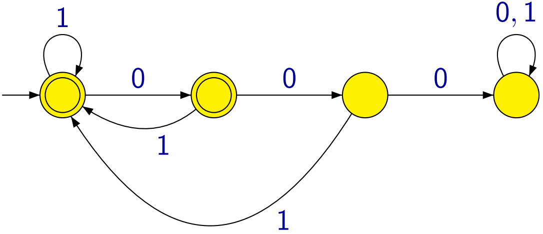
Examples of Deterministic Finite Automata

Example: An automaton recognizing the language over alphabet $\{0, 1\}$ consisting of those words where every occurrence of symbol 0 is immediately followed with symbol 1.



	0	1
$\leftrightarrow$ 1	2	1
2	3	1
3	3	3

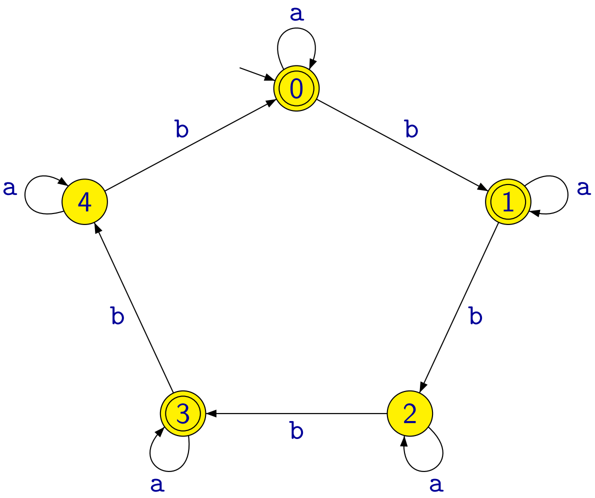
Example: An automaton recognizing the language over alphabet $\{0, 1\}$ consisting of those words where every pair of consecutive symbols 0 is immediately followed with symbol 1.



Examples of Deterministic Finite Automata

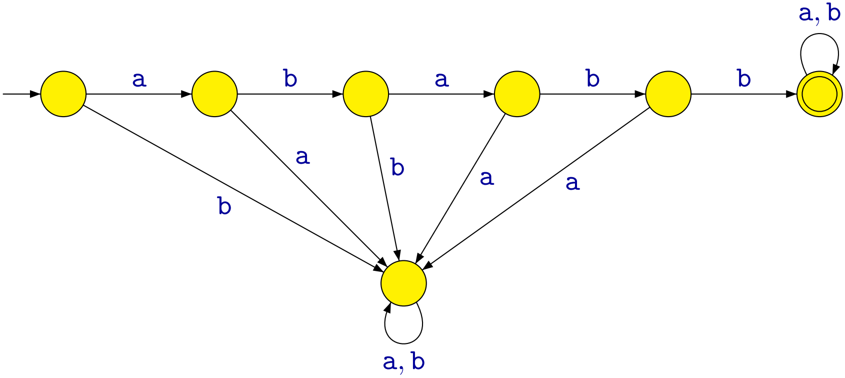
Example: An automaton recognizing the language

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid (|w|_{\mathtt{b}} \bmod 5) \in \{0, 1, 3\}\}$$

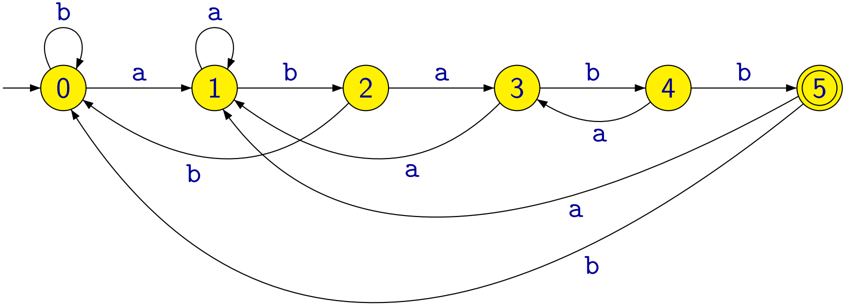


Examples of Deterministic Finite Automata

Example: An automaton recognizing the language over alphabet $\{a, b\}$ consisting of those words that start with the **prefix** ababb.



Example: An automaton recognizing the language over alphabet $\{a, b\}$ of those words that end with **suffix** ababb.



Examples of Deterministic Finite Automata

The construction of this automaton is based on the following idea:

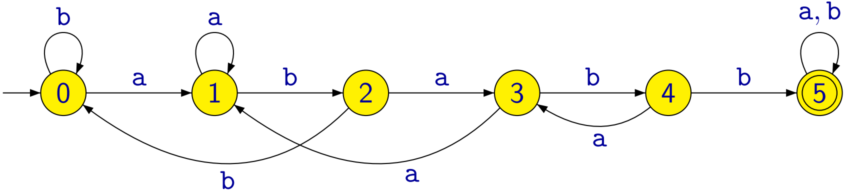
- Let us assume that we want to search for a word u of length n (i.e., $|u| = n$).
  The states of the automaton are denoted with numbers $0, 1, \ldots, n$.
- A state with number i corresponds to the situation when i is the length of the longest word that is at the same time:
  - a prefix of the pattern u we are searching for
  - a suffix of the part of the input word that the automaton has read so far

For example, for the searched pattern `ababb` the states of the automaton correspond to the following words:

- State 0 ... ε
- State 1 ... `a`
- State 2 ... `ab`
- State 3 ... `aba`
- State 4 ... `abab`
- State 5 ... `ababb`

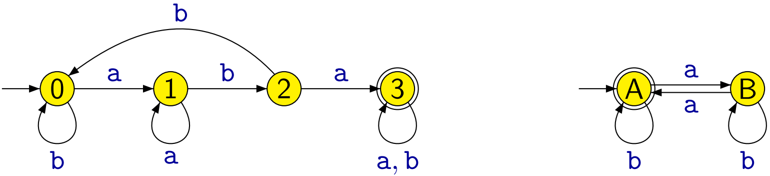
Examples of Deterministic Finite Automata

Example: An automaton recognizing the language over alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of those words that contain **subword** `ababb`.



An Automaton for Intersection of Languages

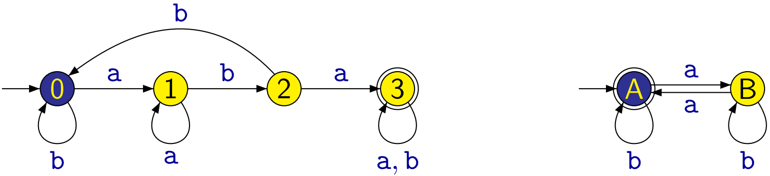
Let us have the following two automata:



Do both of them accept the word ababb?

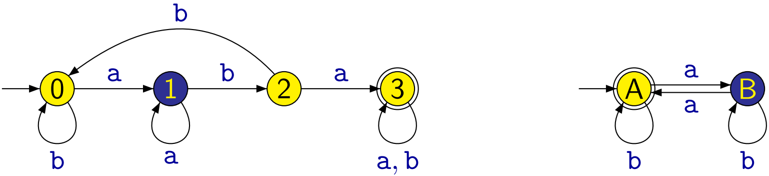
An Automaton for Intersection of Languages

Let us have the following two automata:



Do both of them accept the word ababb?

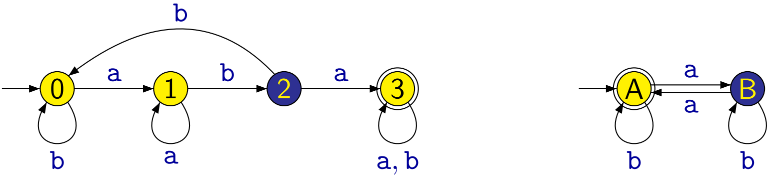
Let us have the following two automata:



Do both of them accept the word ababb?

An Automaton for Intersection of Languages

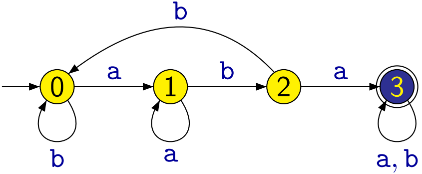
Let us have the following two automata:

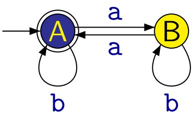


Do both of them accept the word ababb?

An Automaton for Intersection of Languages

Let us have the following two automata:

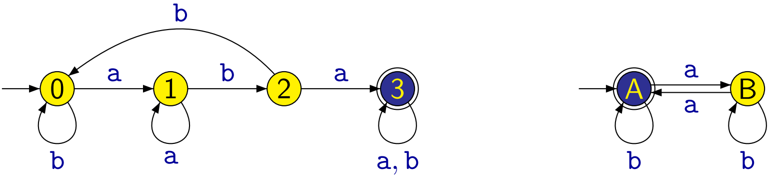




Do both of them accept the word ababb?

An Automaton for Intersection of Languages

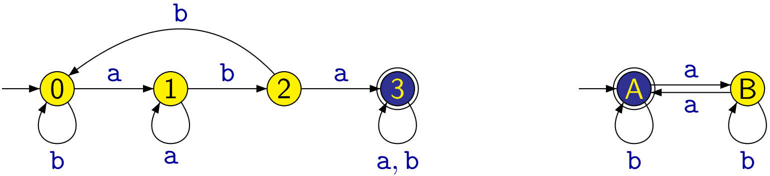
Let us have the following two automata:



Do both of them accept the word ababb?

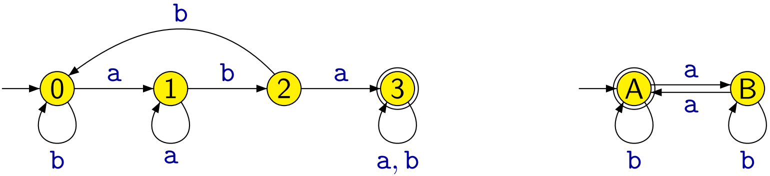
An Automaton for Intersection of Languages

Let us have the following two automata:

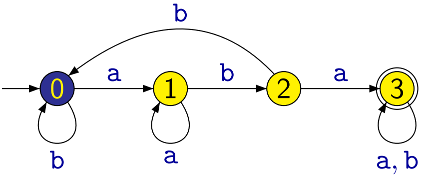


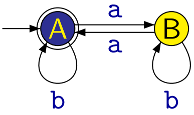
Do both of them accept the word ababb?

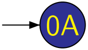
An Automaton for Intersection of Languages



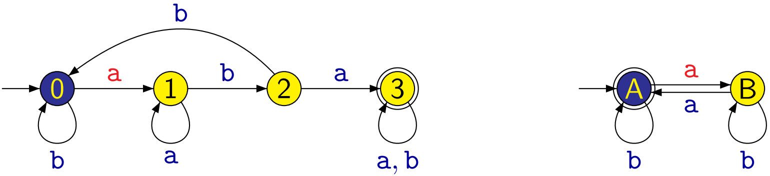
An Automaton for Intersection of Languages

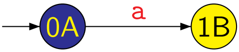




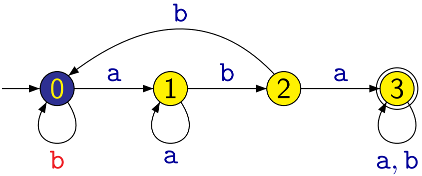


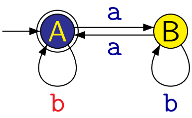
An Automaton for Intersection of Languages

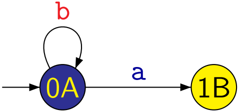




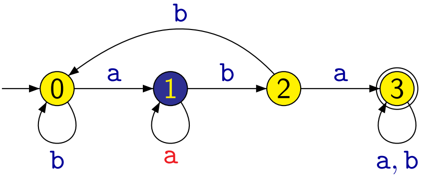
An Automaton for Intersection of Languages

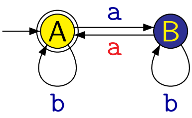


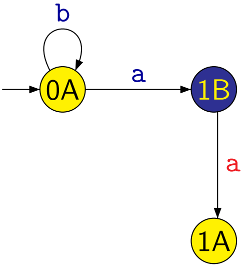




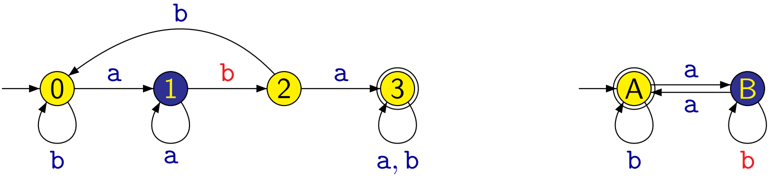
An Automaton for Intersection of Languages

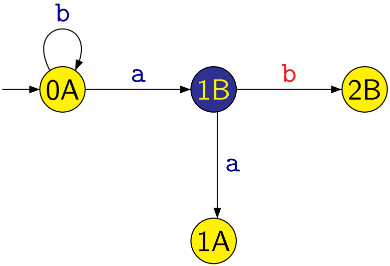




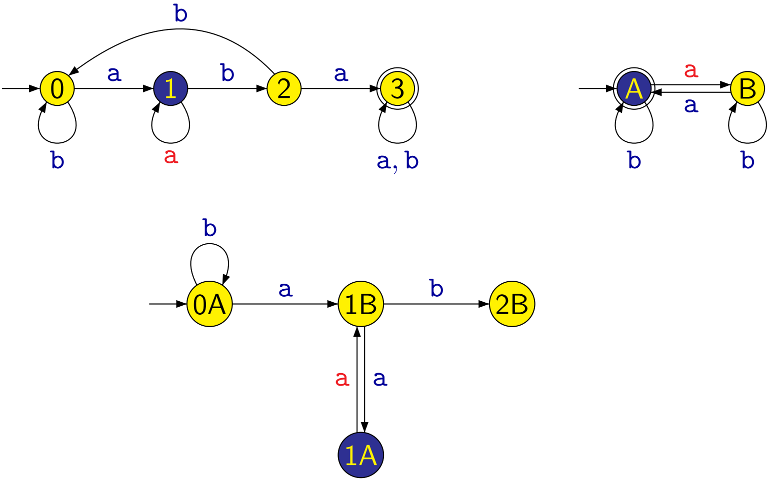


An Automaton for Intersection of Languages

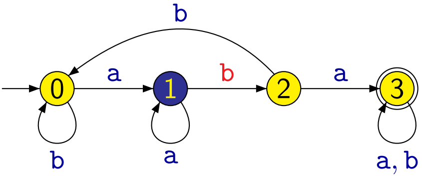


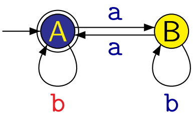


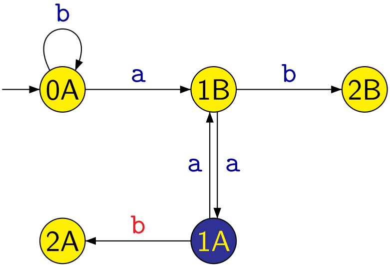
An Automaton for Intersection of Languages



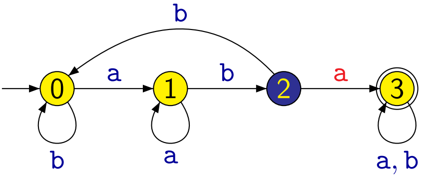
An Automaton for Intersection of Languages

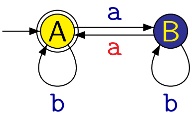


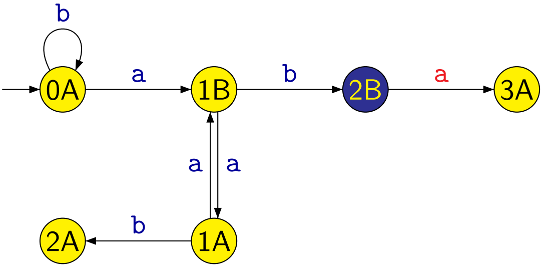




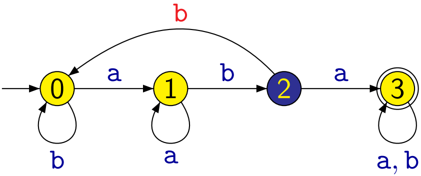
An Automaton for Intersection of Languages

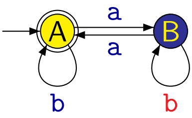


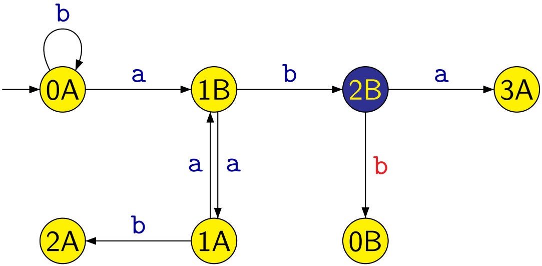




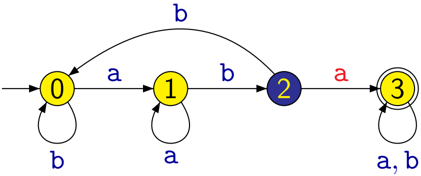
An Automaton for Intersection of Languages

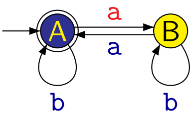


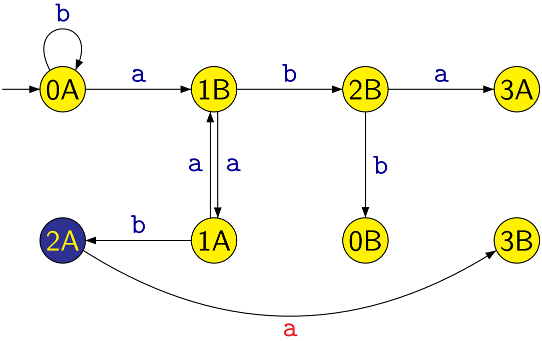




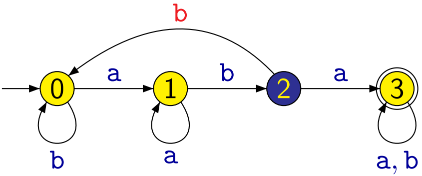
An Automaton for Intersection of Languages

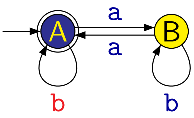


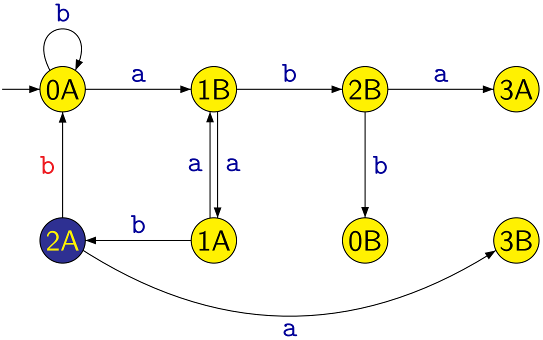




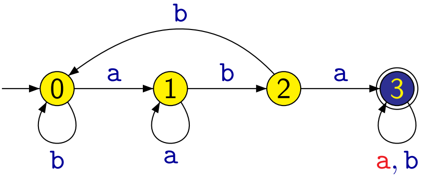
An Automaton for Intersection of Languages

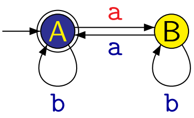


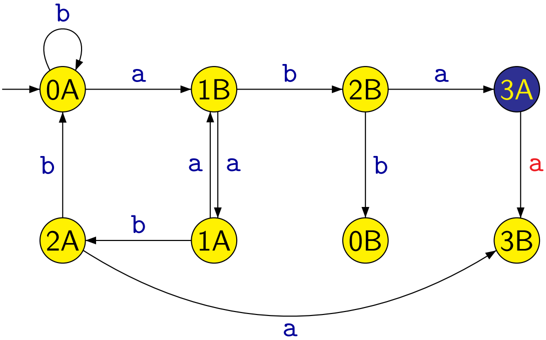




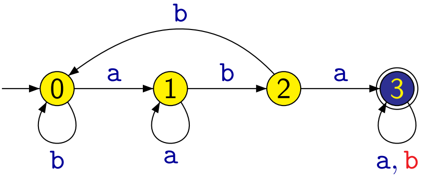
An Automaton for Intersection of Languages

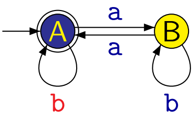


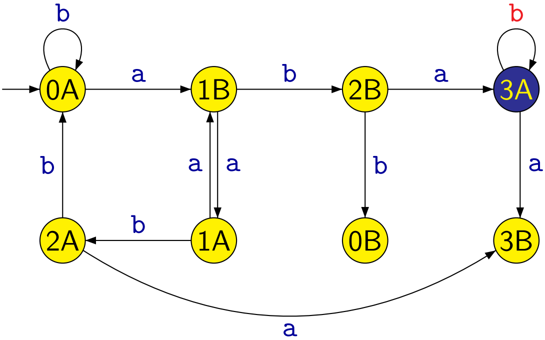




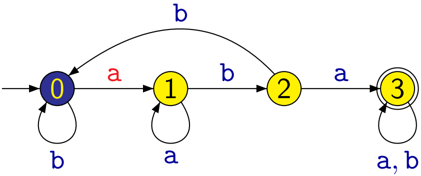
An Automaton for Intersection of Languages

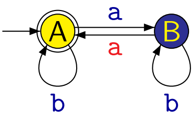


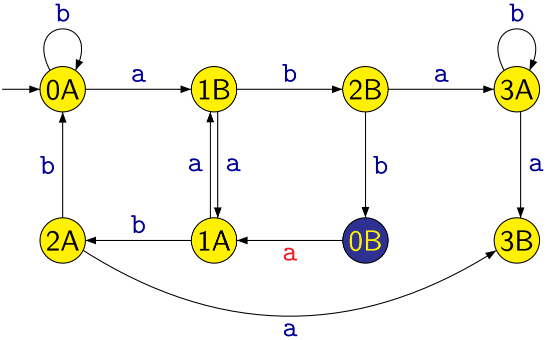




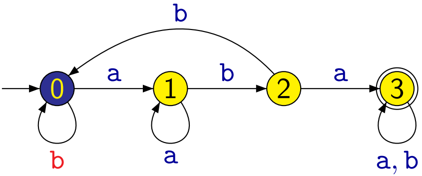
An Automaton for Intersection of Languages

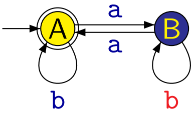


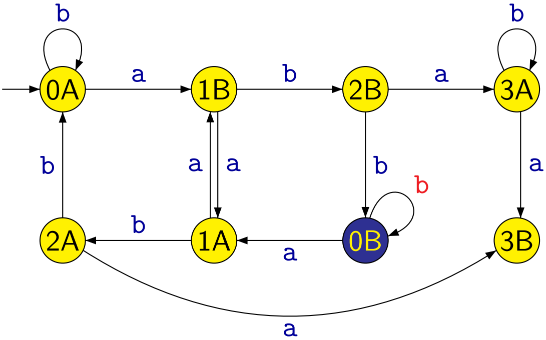




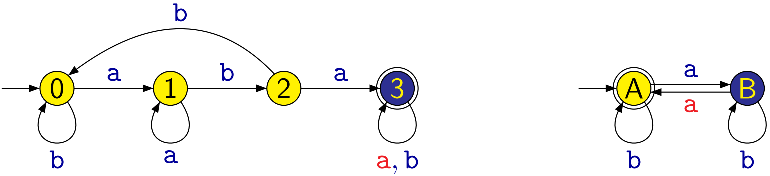
An Automaton for Intersection of Languages

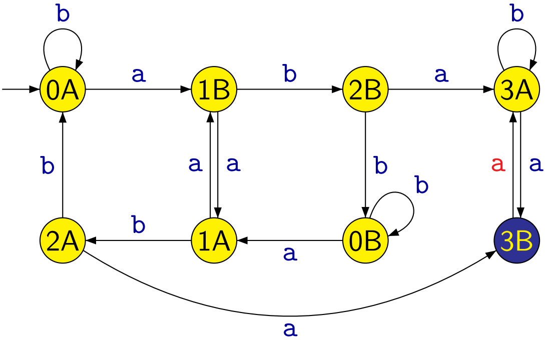




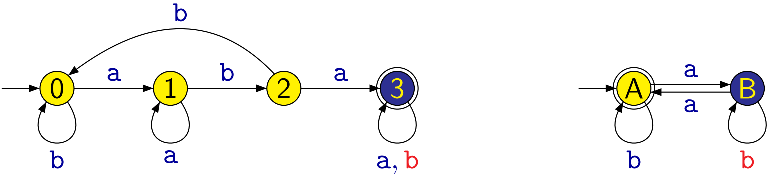


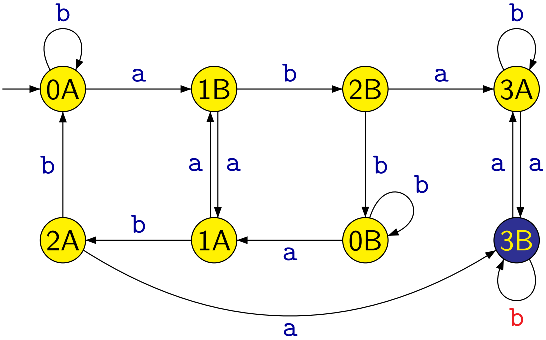
An Automaton for Intersection of Languages



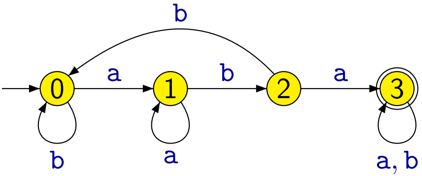


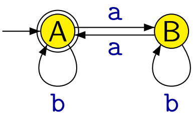
An Automaton for Intersection of Languages

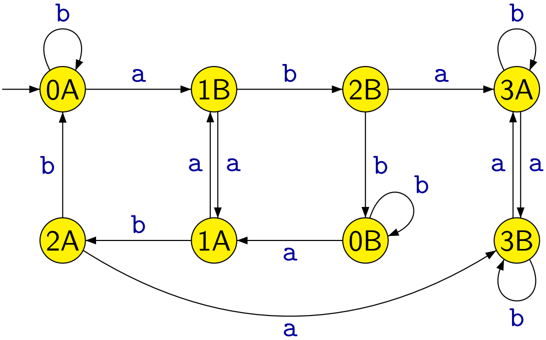




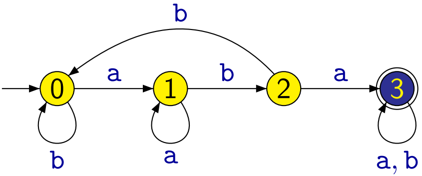
An Automaton for Intersection of Languages

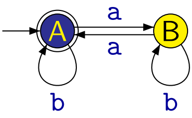


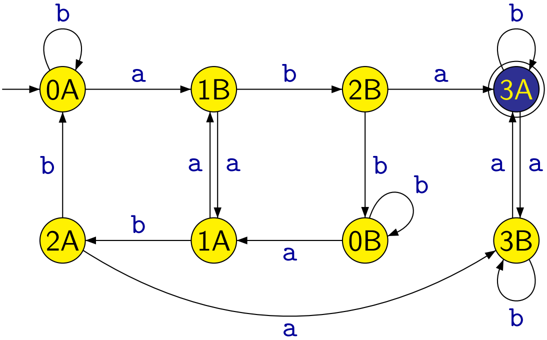




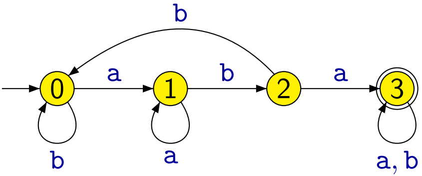
An Automaton for Intersection of Languages

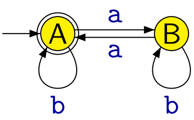


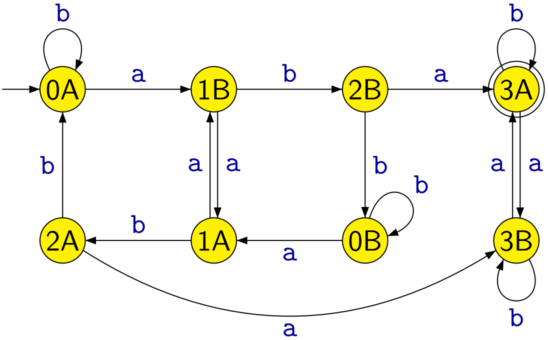




An Automaton for Intersection of Languages







An Automaton for Intersection of Languages

Formally, the construction can be described as follows:

We assume we have two deterministic finite automata $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$.

We construct DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where:

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

It is not difficult to check that for each word $w \in \Sigma^*$ we have $w \in \mathcal{L}(\mathcal{A})$ iff $w \in \mathcal{L}(\mathcal{A}_1)$ and $w \in \mathcal{L}(\mathcal{A}_2)$, i.e.,

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$$

Intersection of Regular Languages

Theorem

If languages $L_1, L_2 \subseteq \Sigma^*$ are regular then also the language $L_1 \cap L_2$ is regular.

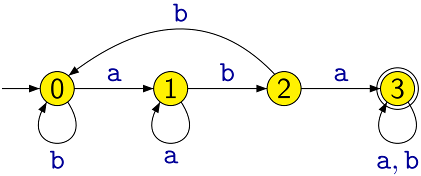
Proof: Let us assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic finite automata such that

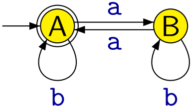
$$L_1 = \mathcal{L}(\mathcal{A}_1) \qquad L_2 = \mathcal{L}(\mathcal{A}_2)$$

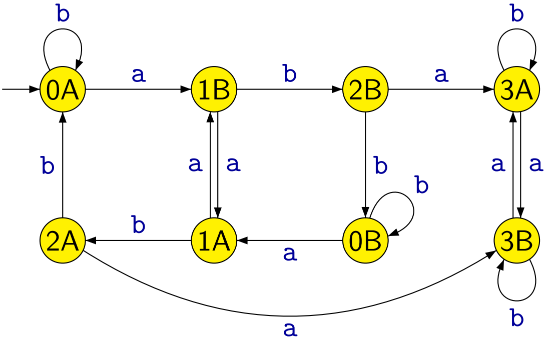
Using the described construction, we can construct a deterministic finite automaton $\mathcal{A}$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = L_1 \cap L_2$$

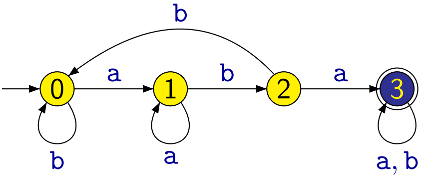
An Automaton for the Union of Languages

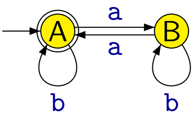


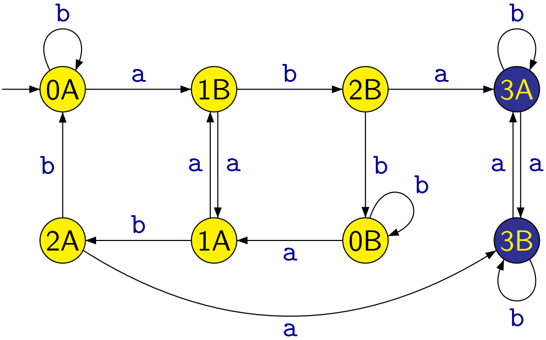




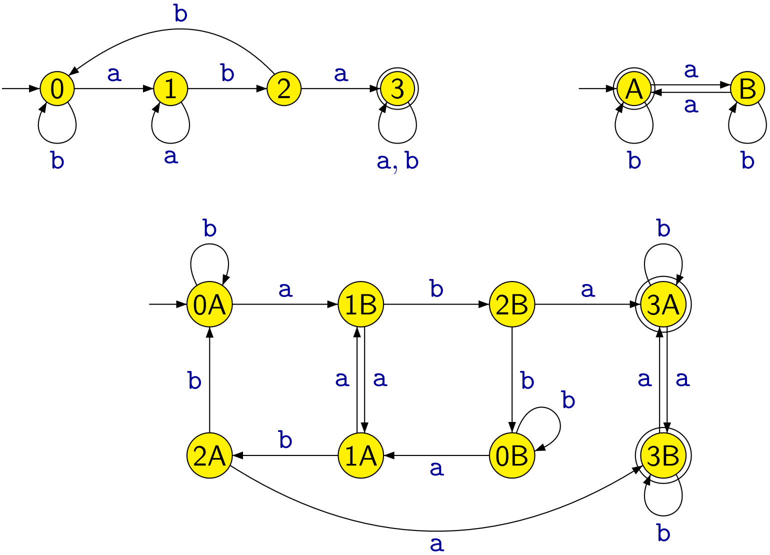
An Automaton for the Union of Languages



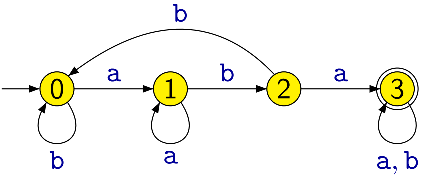


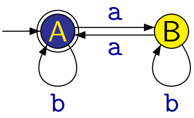


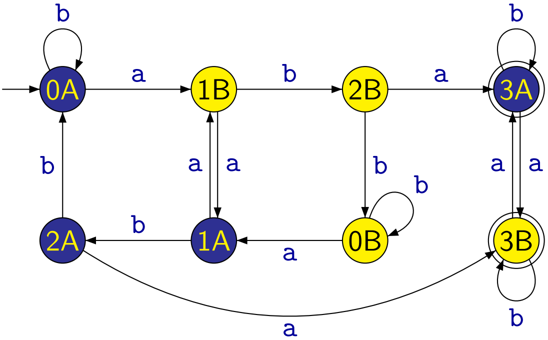
An Automaton for the Union of Languages



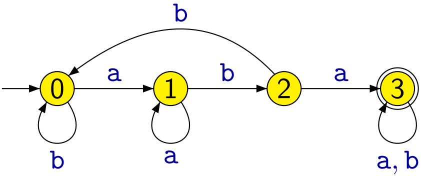
An Automaton for the Union of Languages

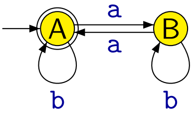


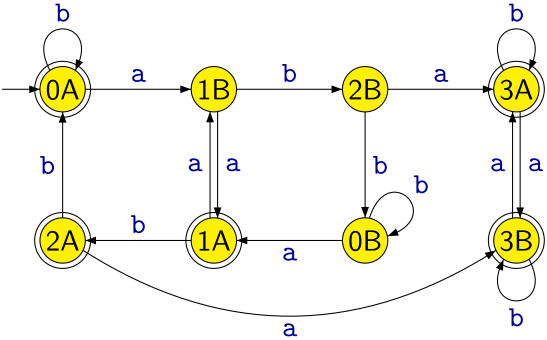




An Automaton for the Union of Languages







Union of Regular Languages

The construction of an automaton $\mathcal{A}$ that accepts the **union** of languages accepted by automata $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e., the language

$$\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_1)$$

is almost identical as in the case of the automaton accepting $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

The only difference is the set of accepting states:

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

Union of Regular Languages

The construction of an automaton $\mathcal{A}$ that accepts the **union** of languages accepted by automata $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e., the language

$$\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_1)$$

is almost identical as in the case of the automaton accepting $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

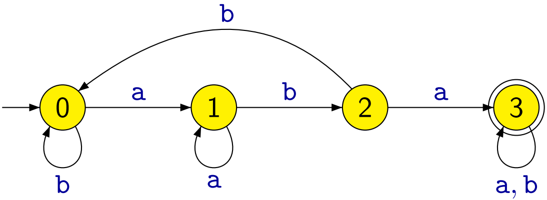
The only difference is the set of accepting states:

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

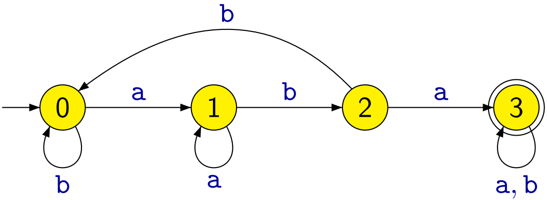
Theorem

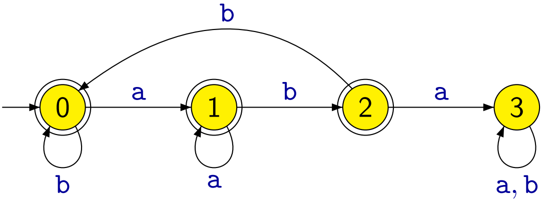
If languages $L_1, L_2 \subseteq \Sigma^*$ are regular then also the language $L_1 \cup L_2$ is regular.

An Automaton for the Complement of a Language



An Automaton for the Complement of a Language





Complement of a Regular Language

Given a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ we construct DFA $\mathcal{A}' = (Q, \Sigma, \delta, q_0, Q - F)$.

It is obvious that for each word $w \in \Sigma^*$ we have $w \in \mathcal{L}(\mathcal{A}')$ iff $w \notin \mathcal{L}(\mathcal{A})$, i.e.,

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

Complement of a Regular Language

Given a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ we construct DFA $\mathcal{A}' = (Q, \Sigma, \delta, q_0, Q - F)$.

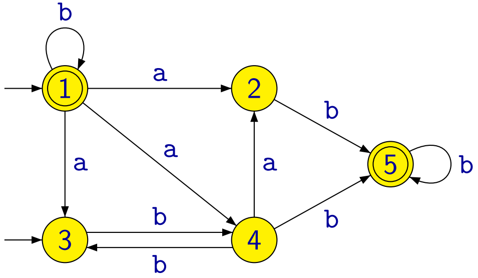
It is obvious that for each word $w \in \Sigma^*$ we have $w \in \mathcal{L}(\mathcal{A}')$ iff $w \notin \mathcal{L}(\mathcal{A})$, i.e.,

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

Theorem

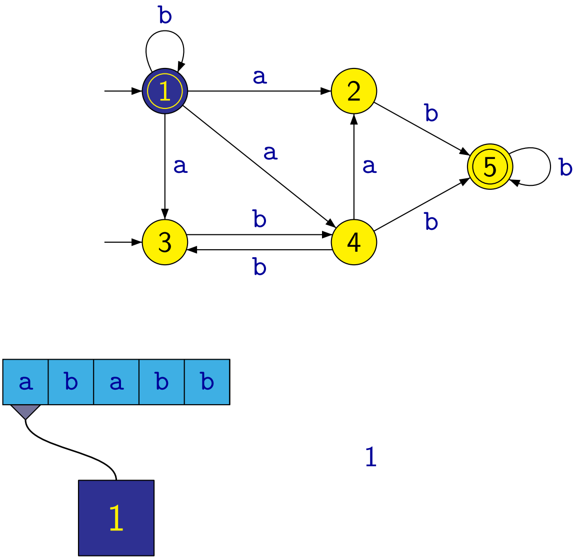
If a language L is regular then also its complement $\overline{L}$ is regular.

Nondeterministic Finite Automaton



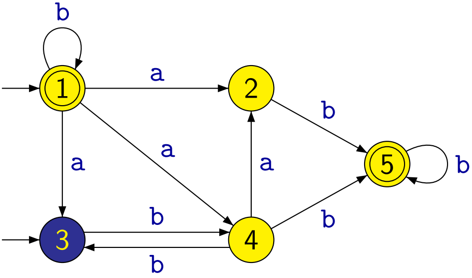
- The number of transitions going from one state and labelled with the same symbol can be arbitrary (including zero).
- There can be more than one initial state in the automaton.

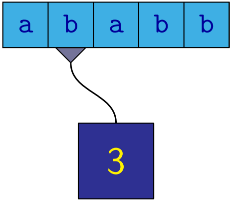
Nondeterministic Finite Automaton



1

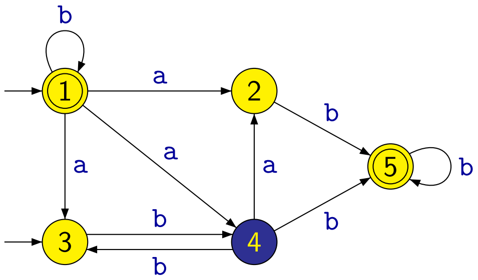
Nondeterministic Finite Automaton

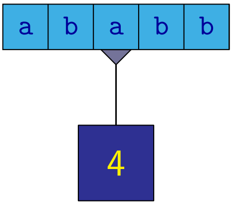




$1 \xrightarrow{a} 3$

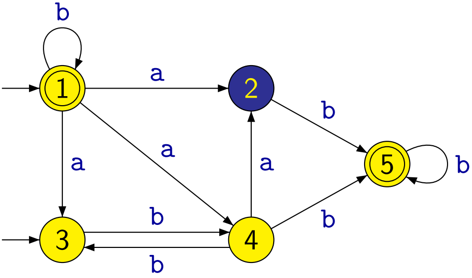
Nondeterministic Finite Automaton

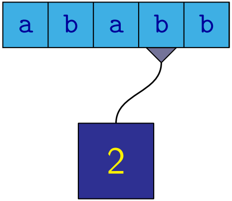




$1 \xrightarrow{a} 3 \xrightarrow{b} 4$

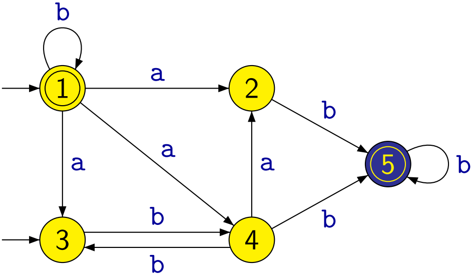
Nondeterministic Finite Automaton

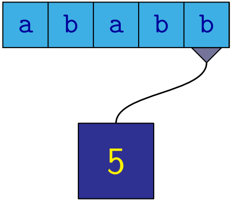




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\texttt{a}} 2$$

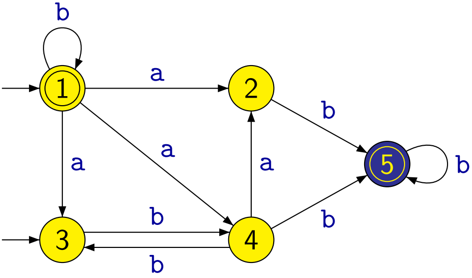
Nondeterministic Finite Automaton

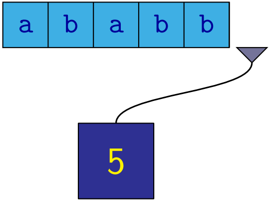




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{a} 2 \xrightarrow{b} 5$$

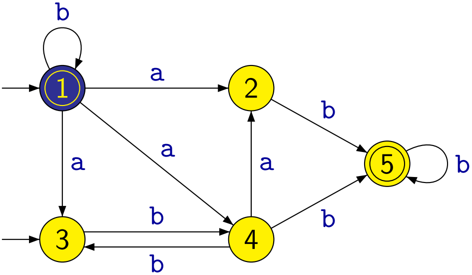
Nondeterministic Finite Automaton

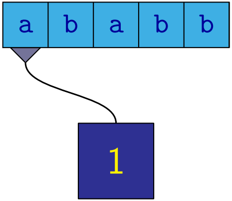




$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{b} 5$

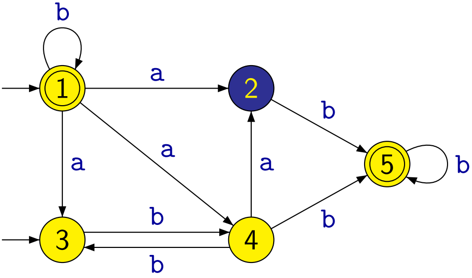
Nondeterministic Finite Automaton

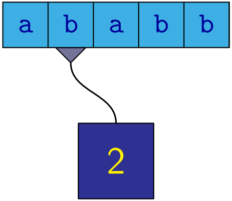




1

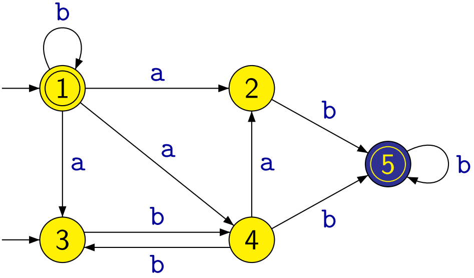
Nondeterministic Finite Automaton

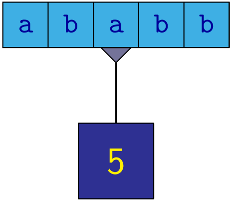




$1 \xrightarrow{a} 2$

Nondeterministic Finite Automaton

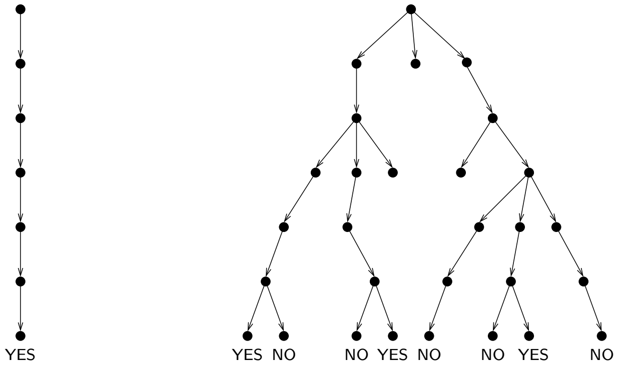




$$1 \xrightarrow{\mathtt{a}} 2 \xrightarrow{\mathtt{b}} 5$$

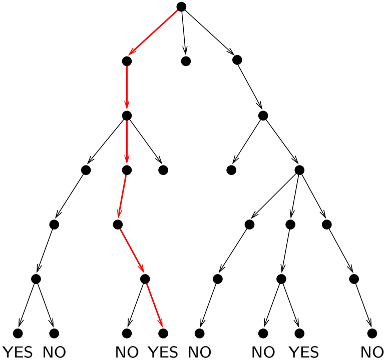
Nondeterministic Finite Automaton

A nondeterministic finite automaton accepts a given word if there **exists** at least one computation of the automaton that accepts the word.



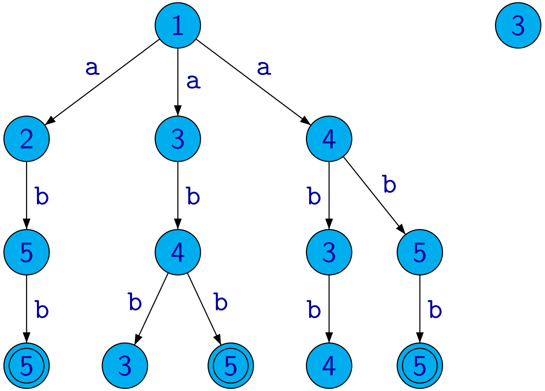
Nondeterministic Finite Automaton

A nondeterministic finite automaton accepts a given word if there **exists** at least one computation of the automaton that accepts the word.



Nondeterministic Finite Automaton

	a	b
$\leftrightarrow 1$	$2, 3, 4$	1
2	$-$	5
$\rightarrow 3$	$-$	4
4	2	$3, 5$
$\leftarrow 5$	$-$	5



Example: A forest representing all possible computations over the word abb.

Nondeterministic Finite Automaton

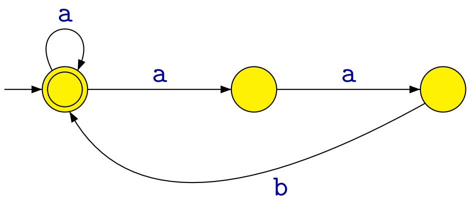
Formally, a **nondeterministic finite automaton** (**NFA**) is defined as a tuple

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

where:

- Q is a finite set of **states**
- Σ is a finite **alphabet**
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is a **transition fuction**
- $I \subseteq Q$ is a set of **initial states**
- $F \subseteq Q$ is a set of **accepting states**

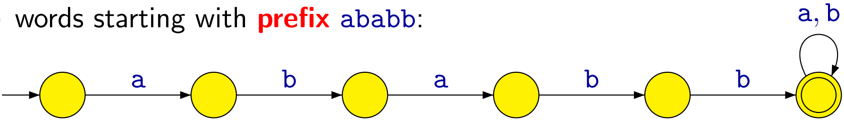
Example: An automaton recognizing the language over alphabet $\{a, b\}$ consisting of those words where every occurrence of symbol b is immediately preceded with two symbols a.



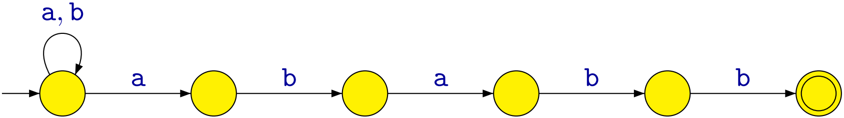
Examples of Nondeterministic Finite Automata

Example: An automaton recognizing the language over alphabet $\{a, b\}$:

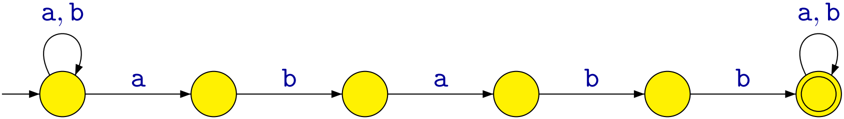
- words starting with **prefix** ababb:



- words ending with **suffix** ababb:

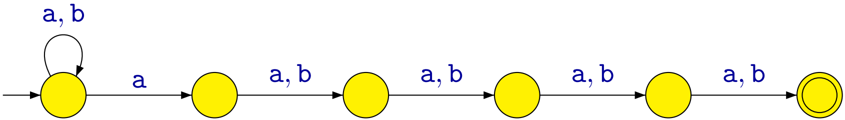


- words containing **subword** ababb:

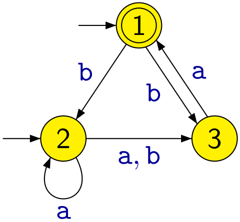


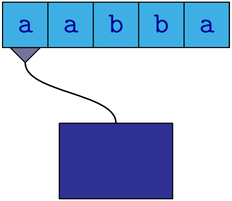
Examples of Nondeterministic Finite Automata

Example: An automaton recognizing the language over alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of those words where the fifth symbol from the end is $\mathtt{a}$.

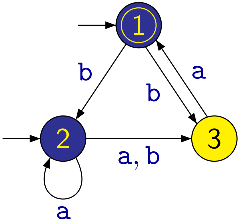


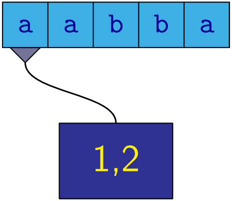
Transformation of NFA to DFA



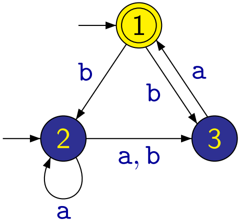


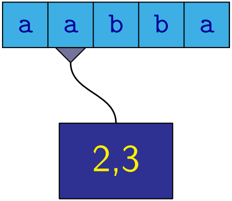
Transformation of NFA to DFA



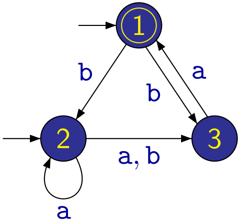


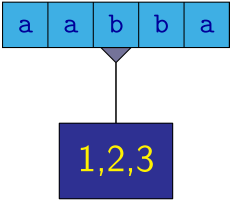
Transformation of NFA to DFA



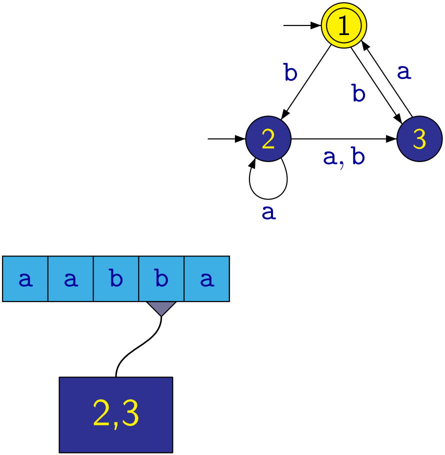


Transformation of NFA to DFA

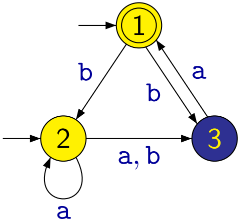


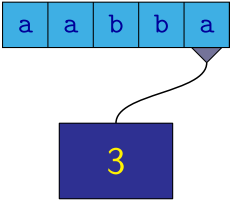


Transformation of NFA to DFA

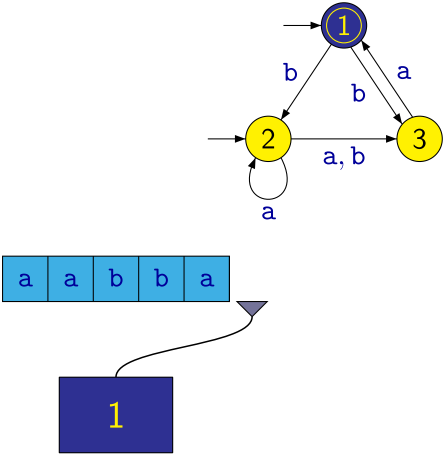


Transformation of NFA to DFA

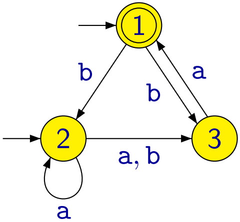




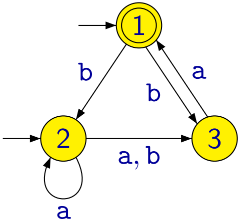
Transformation of NFA to DFA



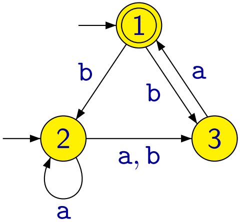
Transformation of NFA to DFA

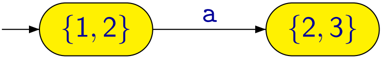


Transformation of NFA to DFA

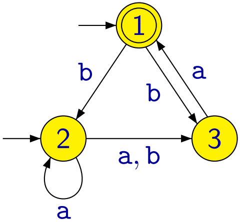


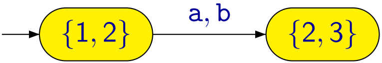
Transformation of NFA to DFA



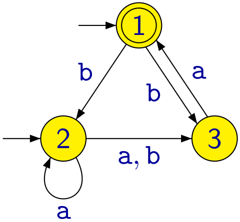


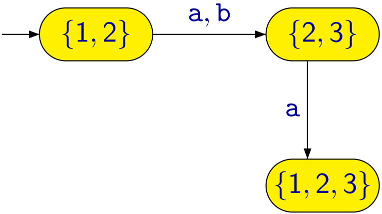
Transformation of NFA to DFA



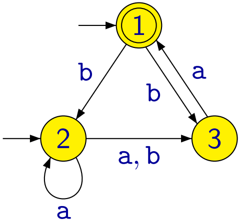


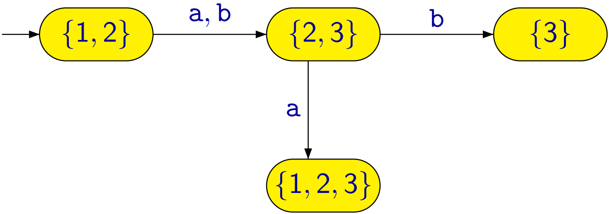
Transformation of NFA to DFA



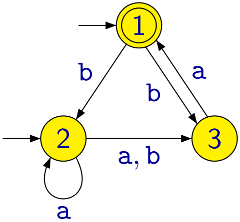


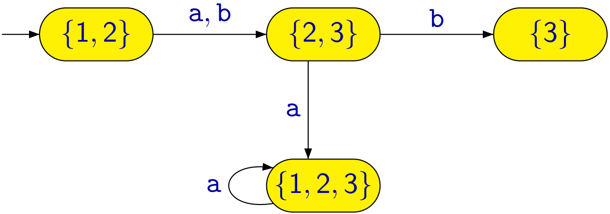
Transformation of NFA to DFA



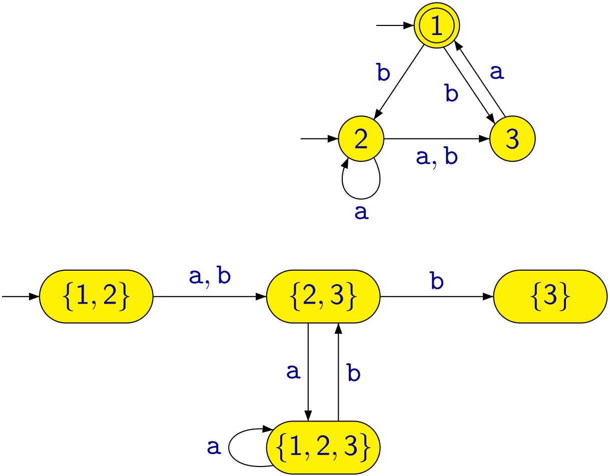


Transformation of NFA to DFA

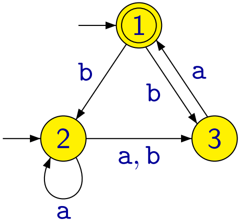


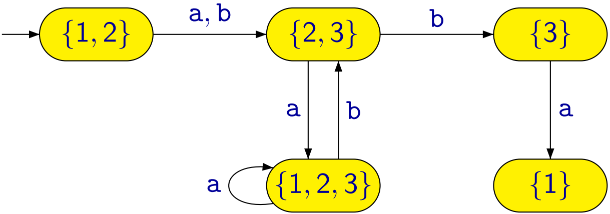


Transformation of NFA to DFA

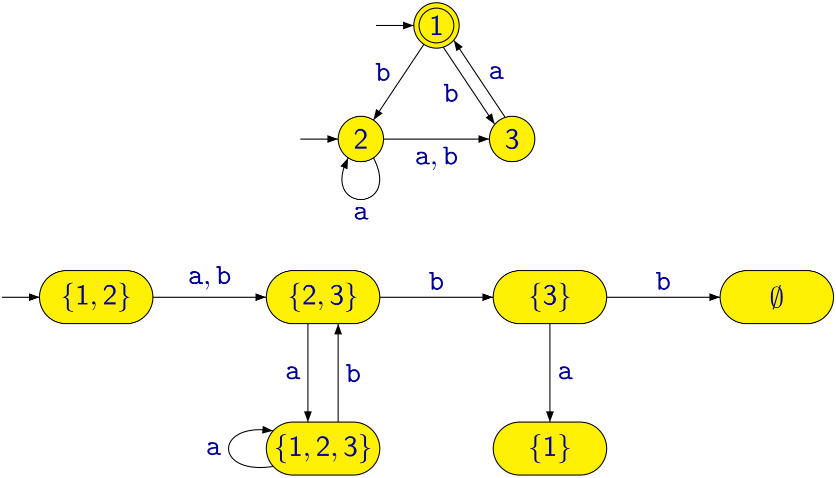


Transformation of NFA to DFA

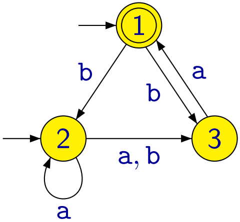


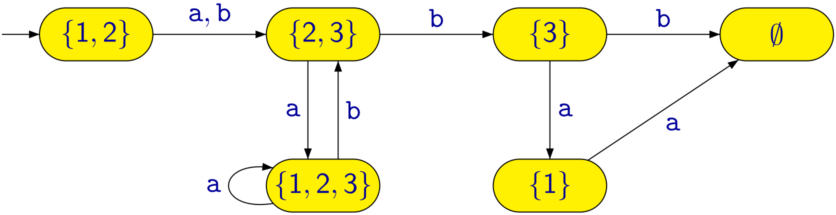


Transformation of NFA to DFA

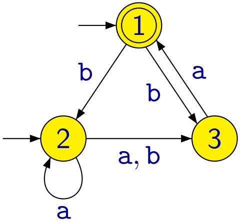


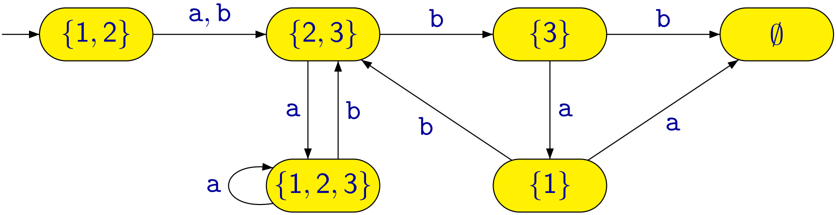
Transformation of NFA to DFA



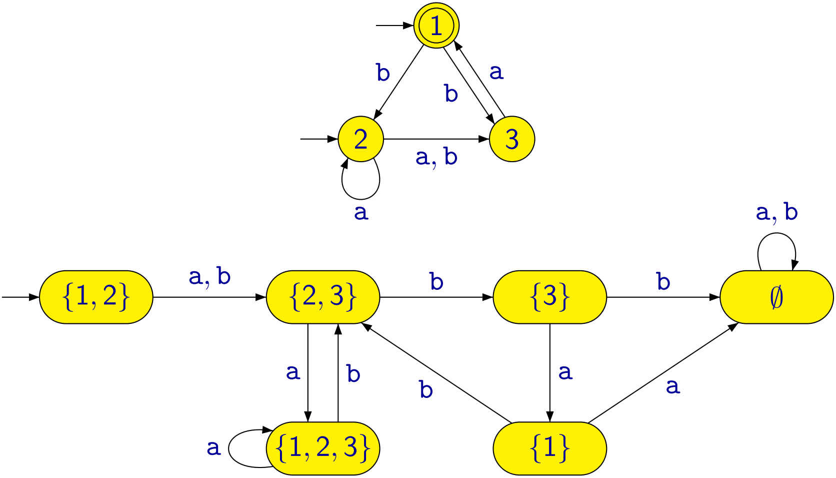


Transformation of NFA to DFA

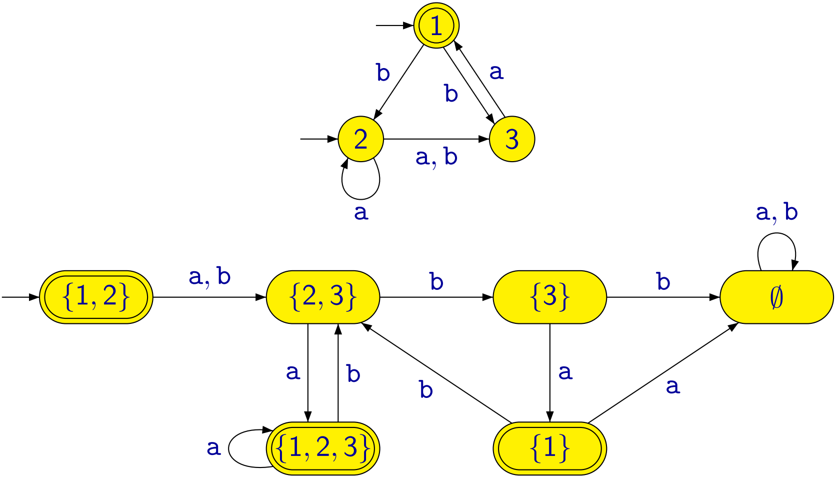




Transformation of NFA to DFA



Transformation of NFA to DFA



Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	$-$

	a	b

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	
$\{2,3\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	
$\leftarrow \{1, 2, 3\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$\leftarrow \{1, 2, 3\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$\leftarrow \{1, 2, 3\}$		
$\{3\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$\leftarrow \{1, 2, 3\}$	$\{1, 2, 3\}$	
$\{3\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$\leftarrow \{1, 2, 3\}$	$\{1, 2, 3\}$	$\{2, 3\}$
$\{3\}$	$\{1\}$	
$\leftarrow \{1\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\emptyset$
$\leftarrow \{1\}$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\emptyset$
$\leftarrow \{1\}$		
$\emptyset$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2,3$
$\rightarrow 2$	$2,3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\leftarrow \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{3\}$	$\{1\}$	$\emptyset$
$\leftarrow \{1\}$	$\emptyset$	
$\emptyset$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	–	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	–

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$\leftarrow \{1, 2, 3\}$	$\{1, 2, 3\}$	$\{2, 3\}$
$\{3\}$	$\{1\}$	$\emptyset$
$\leftarrow \{1\}$	$\emptyset$	$\{2, 3\}$
$\emptyset$		

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	–	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	–

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$\leftarrow \{1, 2, 3\}$	$\{1, 2, 3\}$	$\{2, 3\}$
$\{3\}$	$\{1\}$	$\emptyset$
$\leftarrow \{1\}$	$\emptyset$	$\{2, 3\}$
$\emptyset$	$\emptyset$	$\emptyset$

Transformation of NFA to DFA

	a	b
$\leftrightarrow 1$	$-$	$2, 3$
$\rightarrow 2$	$2, 3$	3
3	1	$-$

	a	b
$\leftrightarrow \{1, 2\}$	$\{2, 3\}$	$\{2, 3\}$
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$\leftarrow \{1, 2, 3\}$	$\{1, 2, 3\}$	$\{2, 3\}$
$\{3\}$	$\{1\}$	$\emptyset$
$\leftarrow \{1\}$	$\emptyset$	$\{2, 3\}$
$\emptyset$	$\emptyset$	$\emptyset$

	a	b
$\leftrightarrow 1$	2	2
2	3	4
$\leftarrow 3$	3	2
4	5	6
$\leftarrow 5$	6	2
6	6	6

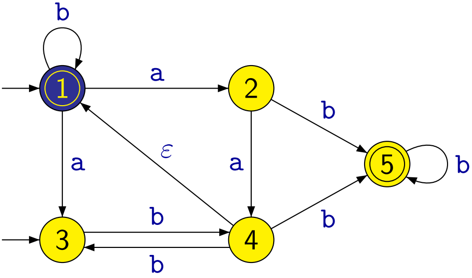
Transformation of NFA to DFA

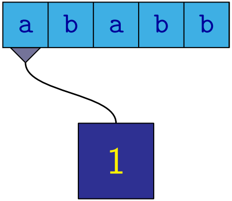
Remark: When a nondeterministic automaton with n states is transformed into a deterministic one, the resulting automaton can have 2^n states.

For example when we transform an automaton with 20 states, the resulting automaton can have $2^{20} = 1048576$ states.

It is often the case that the resulting automaton has far less than 2^n states. However, the worst cases are possible.

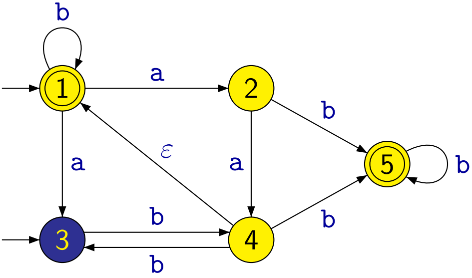
Generalized Nondeterministic Finite Automaton

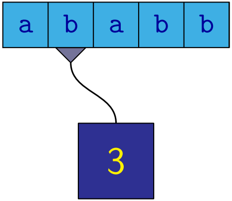




1

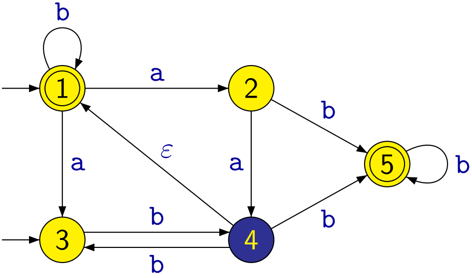
Generalized Nondeterministic Finite Automaton

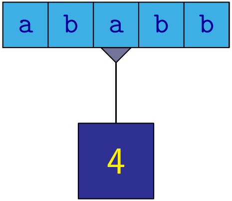




$1 \xrightarrow{\texttt{a}} 3$

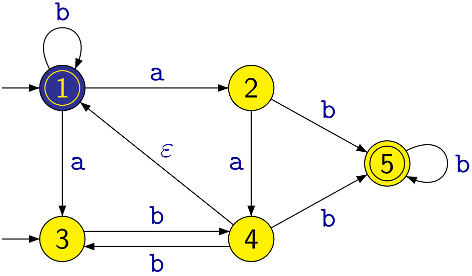
Generalized Nondeterministic Finite Automaton

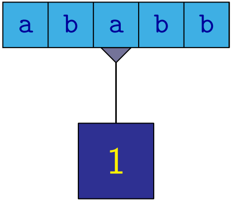




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4$$

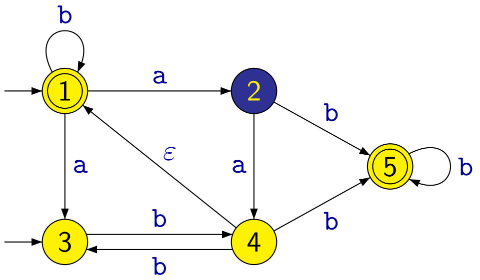
Generalized Nondeterministic Finite Automaton

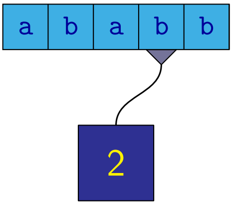




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\varepsilon} 1$$

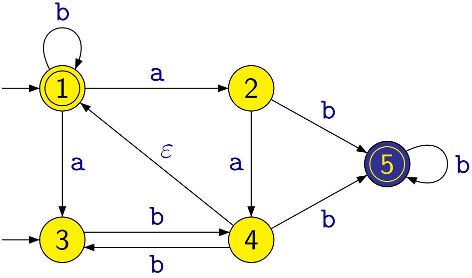
Generalized Nondeterministic Finite Automaton

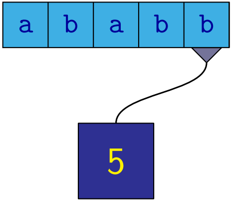




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\varepsilon} 1 \xrightarrow{\texttt{a}} 2$$

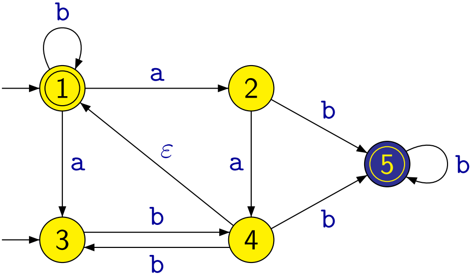
Generalized Nondeterministic Finite Automaton

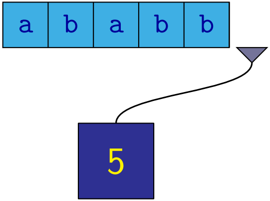




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\varepsilon} 1 \xrightarrow{\texttt{a}} 2 \xrightarrow{\texttt{b}} 5$$

Generalized Nondeterministic Finite Automaton





$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\varepsilon} 1 \xrightarrow{\texttt{a}} 2 \xrightarrow{\texttt{b}} 5 \xrightarrow{\texttt{b}} 5$

Generalized Nondeterministic Finite Automaton

Compared to a nondeterministic finite automaton, a **generalized nondeterministic finite automaton** has the so called **ε-transitions**, i.e., transitions labelled with symbol ε.

When ε-transition is performed, only the state of the control unit is changed but the head on the tape is not moved.

Remark: The computations of a generalized nondeterministic automaton can be of an arbitrary length, even infinite (if the graph of the automaton contains a cycle consisting only of ε-transitions) regardless of the length of the word on the tape.

Generalized Nondeterministic Finite Automaton

Formally, a **generalized nondeterministic finite automaton** (**GNFA**) is defined as a tuple

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

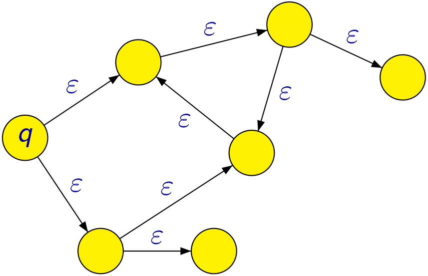
where:

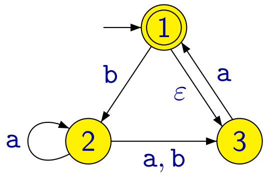
- Q is a finite set of **states**
- Σ is a finite **alphabet**
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$ is a **transition function**
- $I \subseteq Q$ is a set of **initial states**
- $F \subseteq Q$ is a set of **accepting states**

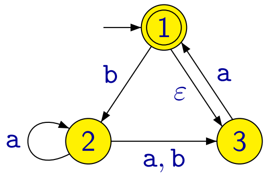
Remark: NFA can be viewed as a special case of GNFA, where $\delta(q, \varepsilon) = \emptyset$ for all $q \in Q$.

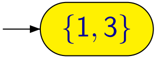
Transformation to a Deterministic Finite Automaton

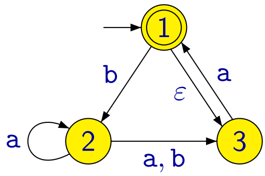
A generalized nondeterministic finite automaton can be transformed into a deterministic one using a similar construction as a nondeterministic finite automaton with the difference that we add to sets of states also all states that are reachable from already added states by some sequence of ε-transitions.

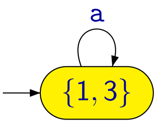


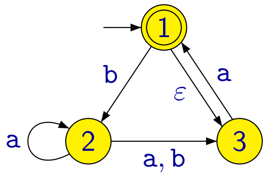

1
2
3
a
b
a
ε
a, b

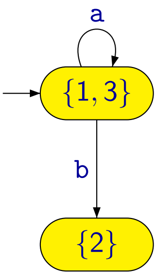

1
b
a
ε
a
2
3
a, b

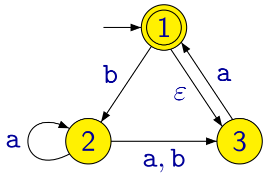

{1, 3}


1
b
a
ε
a
2
3
a, b

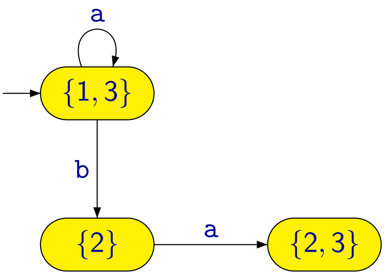

a
{1, 3}


1
b
a
ε
a
2
a, b
3

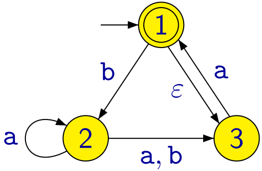

a
{1, 3}
b
{2}

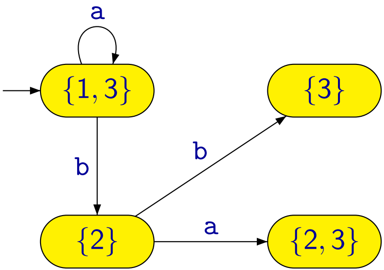


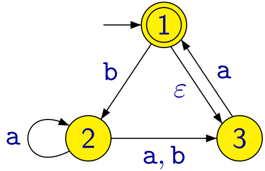
1
b
a
ε
2
a
3
a, b

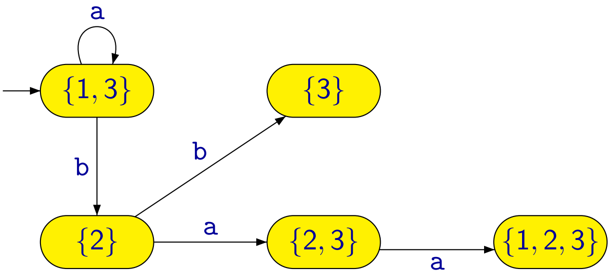


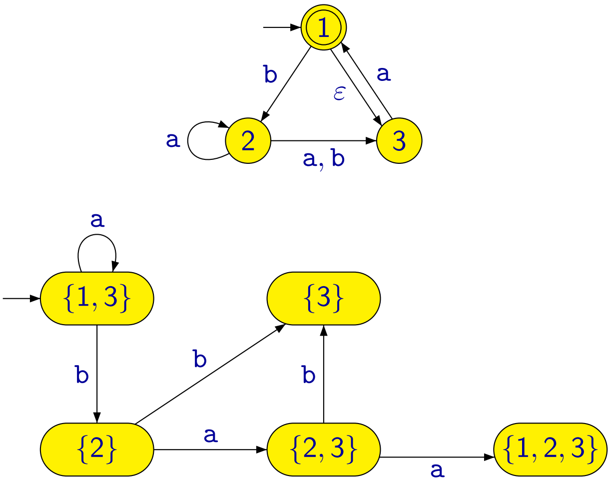
a
{1, 3}
b
{2}
a
{2, 3}

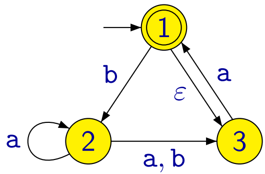

1
b
a
ε
a
2
3
a, b

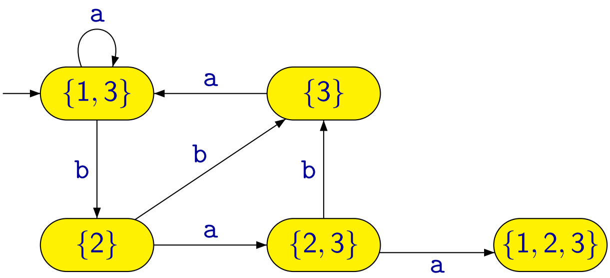

a
{1, 3}
{3}
b
b
{2}
a
{2, 3}

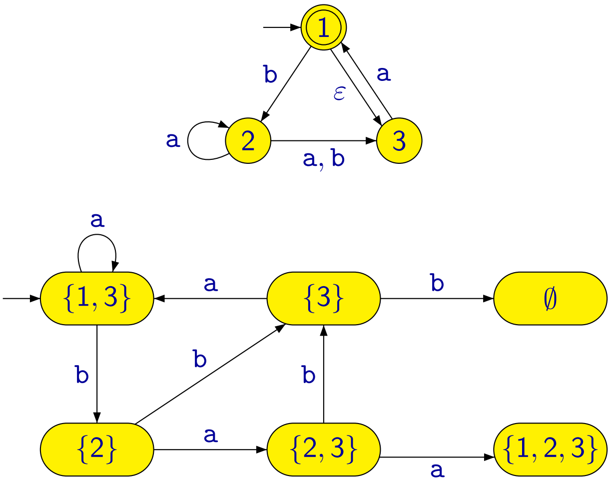

1
b
ε
a
a
2
3
a, b

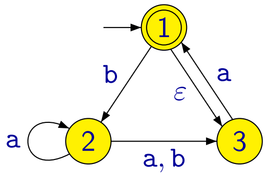

a
{1, 3}
{3}
b
b
{2}
a
{2, 3}
a
{1, 2, 3}

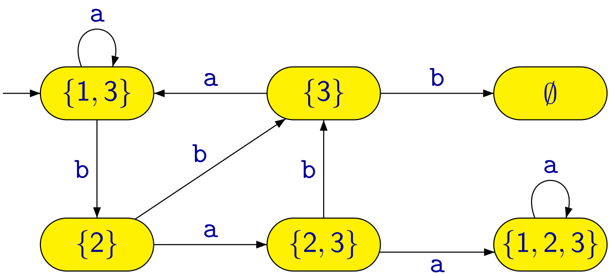

1
b
a
ε
a
2
a, b
3
a
{1, 3}
{3}
b
b
b
{2}
a
{2, 3}
a
{1, 2, 3}

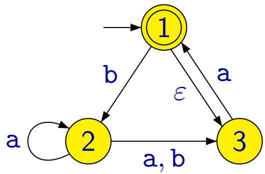

1
2
3
b
a
ε
a
a, b

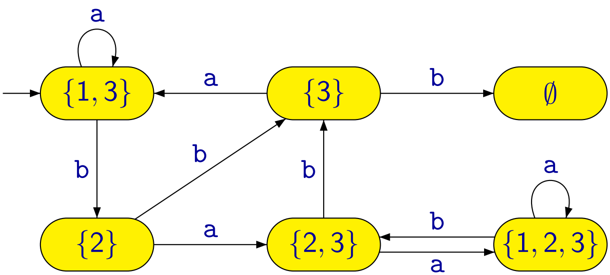

a
{1, 3}
a
{3}
b
b
b
{2}
a
{2, 3}
a
{1, 2, 3}

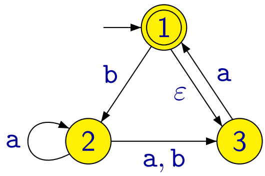

1
b
a
ε
a
2
3
a, b
a
a
b
{1, 3}
{3}
∅
b
b
b
{2}
a
{2, 3}
a
{1, 2, 3}

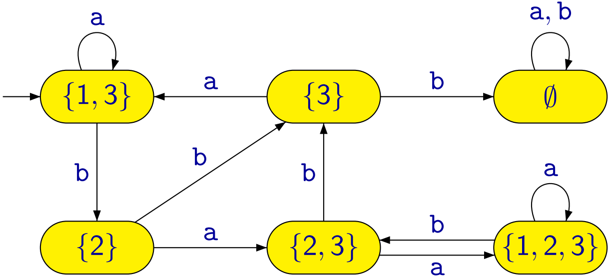

1
b
ε
a
a
2
3
a, b

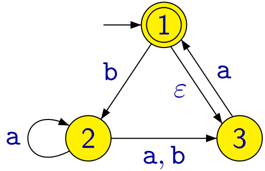

a
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
a
{1, 2, 3}

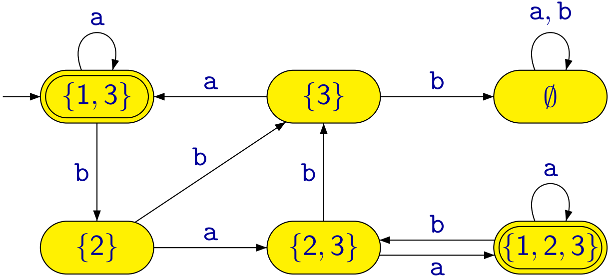

1
b
a
ε
a
2
3
a, b


a
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
b
a
{1, 2, 3}


1
b
ε
a
a
2
3
a, b


a
a, b
a
b
{1, 3}
{3}
∅
b
b
b
a
{2}
a
{2, 3}
b
{1, 2, 3}
a


1
b
ε
a
a
2
3
a, b


a
a, b
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
b
{1, 2, 3}
a

Transformation of GNFA to DFA

Before formally describing the transition of GNFA to DFA, let us introduce some auxiliary definitions.

Let us assume some given GNFA $\mathcal{A} = (Q, \Sigma, \delta, I, F)$.

Let us define the function $\hat{\delta} : \mathcal{P}(Q) \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ so that for $K \subseteq Q$ and $a \in \Sigma \cup \{\varepsilon\}$ there is

$$\hat{\delta}(K, a) = \bigcup_{q \in K} \delta(q, a)$$

Transformation of GNFA to DFA

For $K \subseteq Q$, let $Cl_\varepsilon(K)$ be all the states reachable from the states from the set K by some arbitrary sequence of ε-transitions.

This means that the function $Cl_\varepsilon : \mathcal{P}(Q) \to \mathcal{P}(Q)$ is defined so that for $K \subseteq Q$ is $Cl_\varepsilon(K)$ the smallest (with respect to inclusion) set satisfying the following two conditions:

- $K \subseteq Cl_\varepsilon(K)$
- For each $q \in Cl_\varepsilon(K)$ it holds that $\delta(q, \varepsilon) \subseteq Cl_\varepsilon(K)$.

Remark: Let us note that $Cl_\varepsilon(Cl_\varepsilon(K)) = Cl_\varepsilon(K)$ for arbitrary K.

Let us also note that in the case of NFA (where $\delta(q, \varepsilon) = \emptyset$ for each $q \in Q$) is $Cl_\varepsilon(K) = K$.

Transformation of GNFA to DFA

For a given GNFA $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ we can now construct DFA $\mathcal{A}' = (Q', \Sigma, \delta', q'_0, F')$, where:

- $Q' = \mathcal{P}(Q)$ (so $K \in Q'$ means that $K \subseteq Q$)
- $\delta' : Q' \times \Sigma \to Q'$ is defined so that for $K \in Q'$ and $a \in \Sigma$:

$$\delta'(K, a) = Cl_\varepsilon(\hat{\delta}(Cl_\varepsilon(K), a))$$

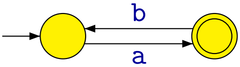
- $q'_0 = Cl_\varepsilon(I)$
- $F' = \{K \in Q' \mid Cl_\varepsilon(K) \cap F \neq \emptyset\}$

It is not difficult to verify that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

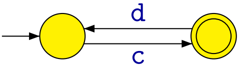
Concatenation of Languages

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

$\mathcal{A}_1$:



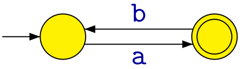
$\mathcal{A}_2$:



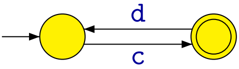
Concatenation of Languages

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

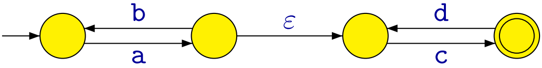
$\mathcal{A}_1$:



$\mathcal{A}_2$:



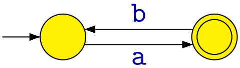
$\mathcal{A}$:



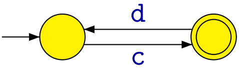
$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$

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

$\mathcal{A}_1$:

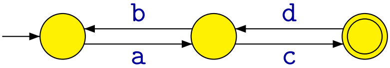


$\mathcal{A}_2$:



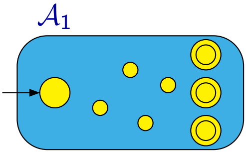
An incorrect construction:

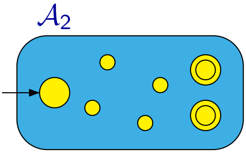
$\mathcal{A}$:



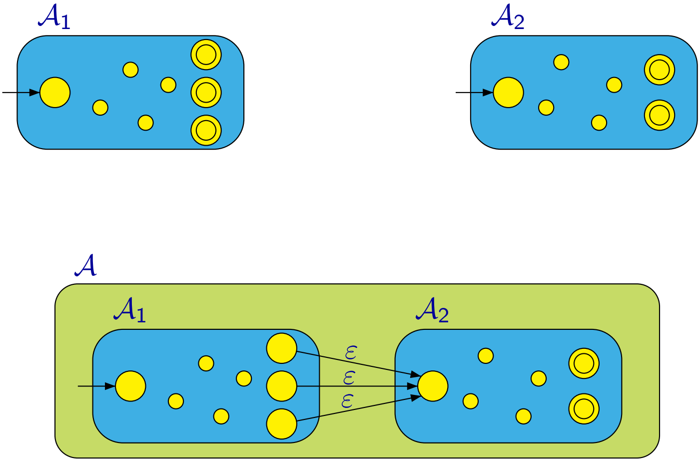
$\mathtt{acdbac} \in \mathcal{L}(\mathcal{A})$ but $\mathtt{acdbac} \notin \mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$

Concatenation of Languages

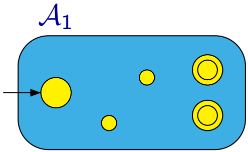




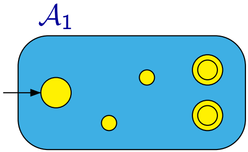
Concatenation of Languages

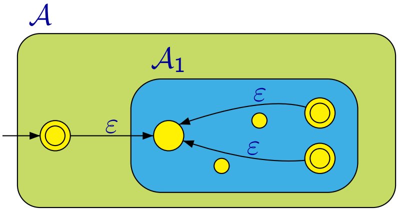


Iteration of a Language



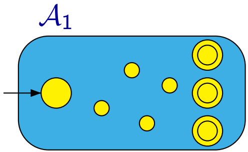
Iteration of a Language

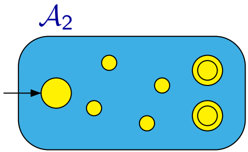




Union of Languages

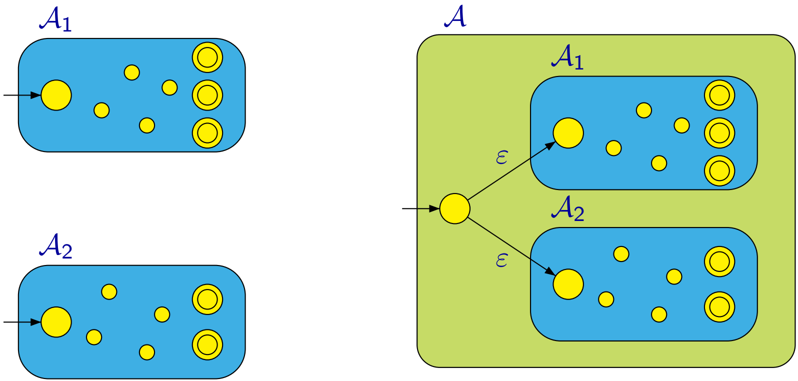
An alternative construction for the union of languages:





Union of Languages

An alternative construction for the union of languages:



Closure Properties of the Class of Regular Languages

The set of (all) regular languages is closed with respect to:

- union
- intersection
- complement
- concatenation
- iteration
- ...