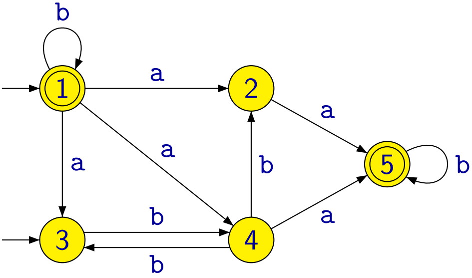
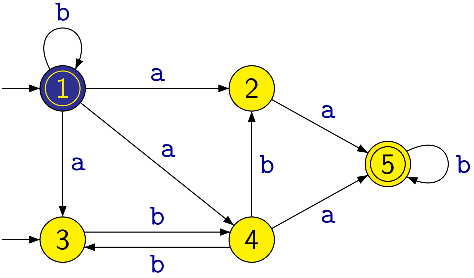
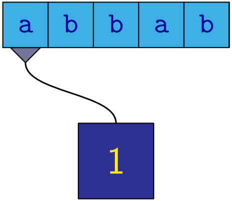
Nondeterministic Finite Automaton



- 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.

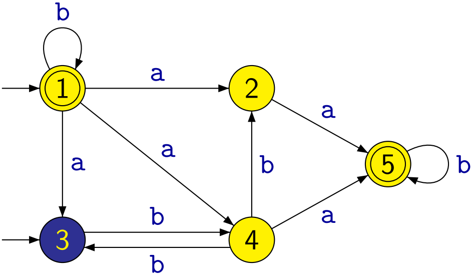
Nondeterministic Finite Automaton

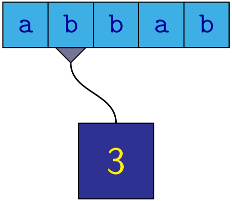




1

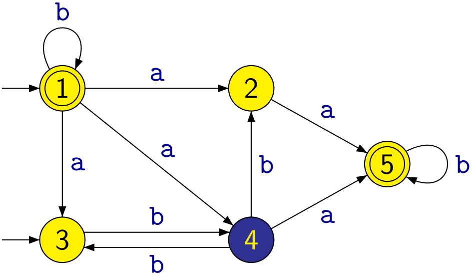
Nondeterministic Finite Automaton

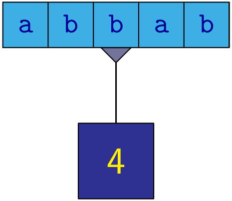




$$1 \xrightarrow{a} 3$$

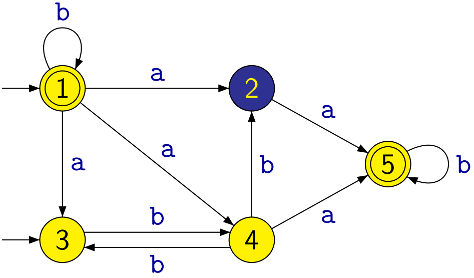
Nondeterministic Finite Automaton

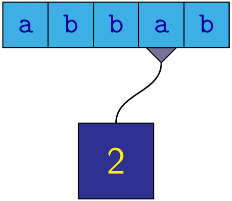




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4$$

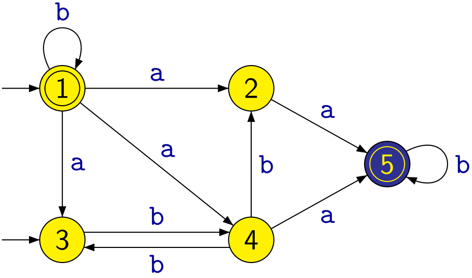
Nondeterministic Finite Automaton

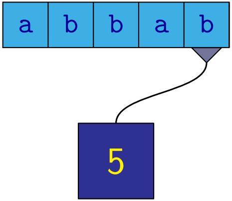




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\texttt{b}} 2$$

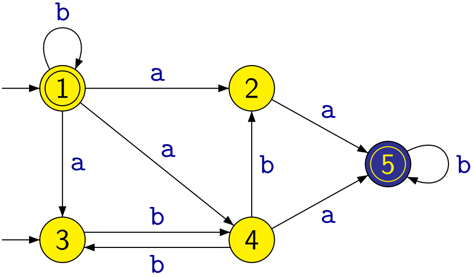
Nondeterministic Finite Automaton

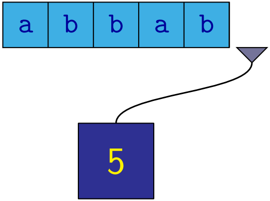




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\texttt{b}} 2 \xrightarrow{\texttt{a}} 5$$

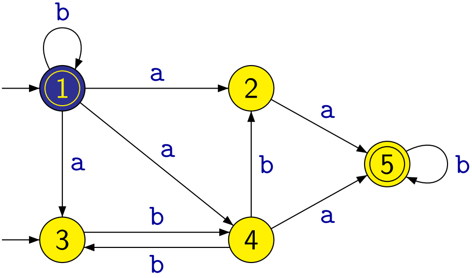
Nondeterministic Finite Automaton

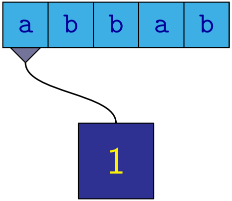




$$1 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{b} 2 \xrightarrow{a} 5 \xrightarrow{b} 5$$

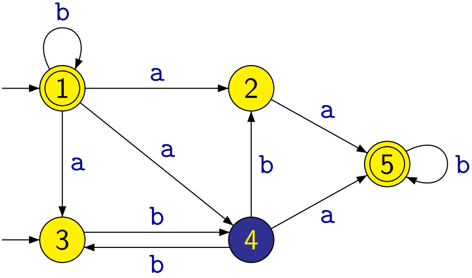
Nondeterministic Finite Automaton

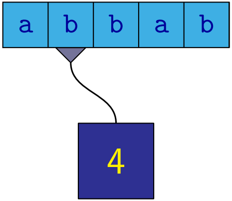




1

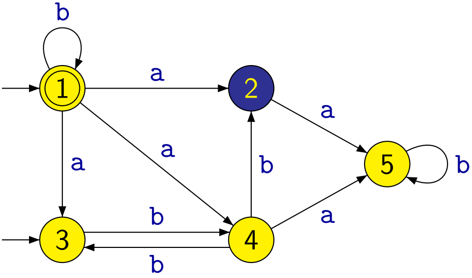
Nondeterministic Finite Automaton

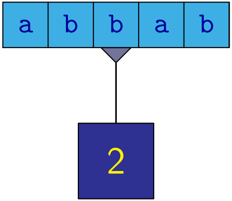




$1 \xrightarrow{a} 4$

Nondeterministic Finite Automaton

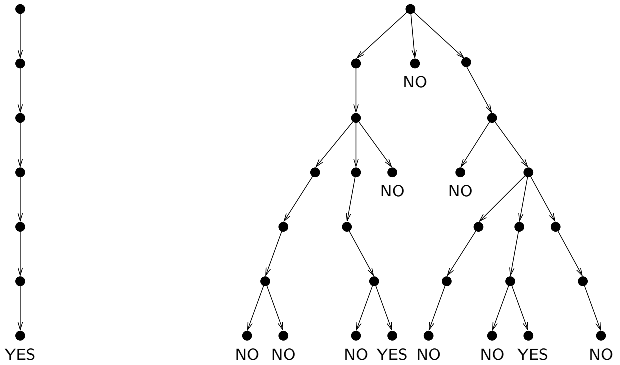




$$1 \xrightarrow{\texttt{a}} 4 \xrightarrow{\texttt{b}} 2$$

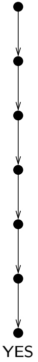
Nondeterministic Finite Automaton

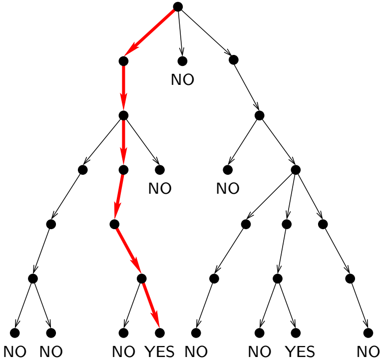
A nondeterministic finite automaton accepts a given word if there **exists** at least one computation of the automaton that accepts the word.



Nondeterministic Finite Automaton

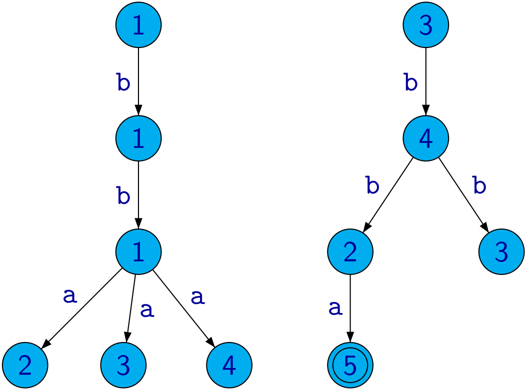
A nondeterministic finite automaton accepts a given word if there **exists** at least one computation of the automaton that accepts the word.





Nondeterministic Finite Automaton

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



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

Nondeterministic Finite Automaton

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

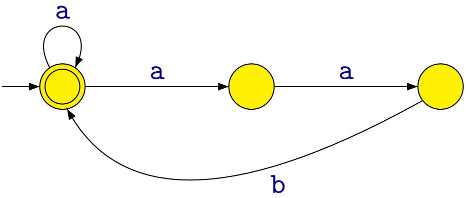
$$(Q, \Sigma, \delta, I, F)$$

where:

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

Examples of Nondeterministic Finite Automata

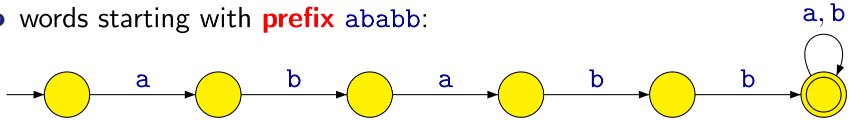
Example: An automaton recognizing the language over alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of those words where every occurrence of symbol $\mathtt{b}$ is immediately preceded with two symbols $\mathtt{a}$.



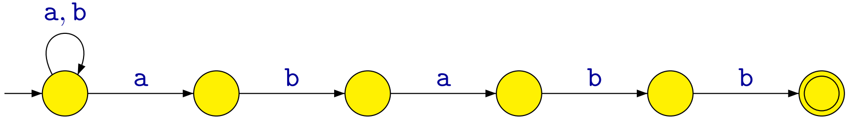
Examples of Nondeterministic Finite Automata

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

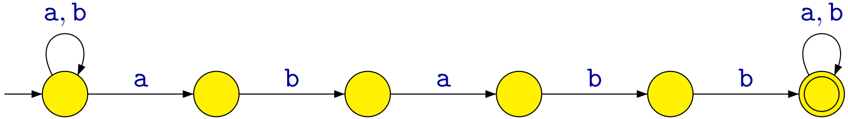
- words starting with **prefix** ababb:



- words ending with **suffix** ababb:

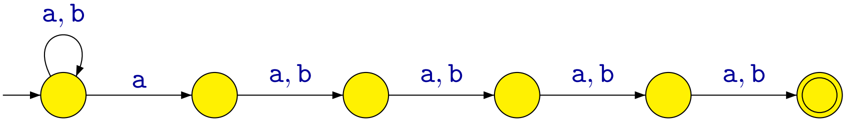


- words containing **subword** ababb:

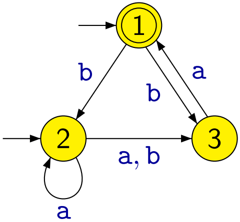


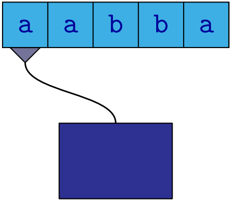
Examples of Nondeterministic Finite Automata

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

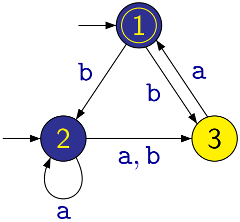


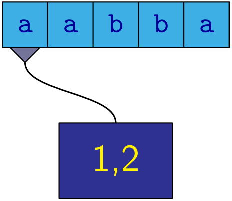
Transformation of NFA to DFA



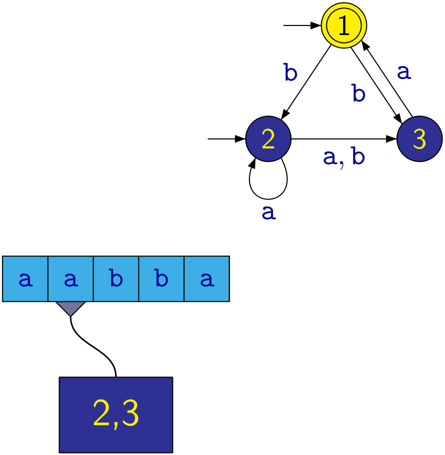


Transformation of NFA to DFA

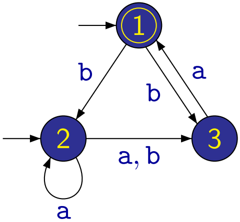


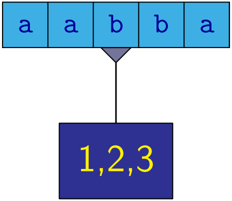


Transformation of NFA to DFA

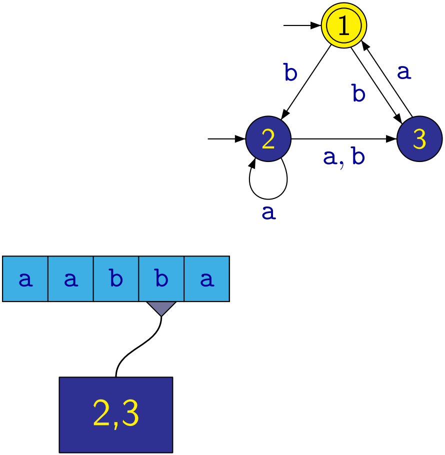


Transformation of NFA to DFA

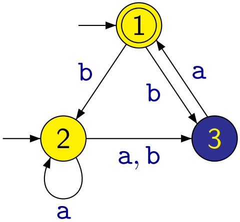


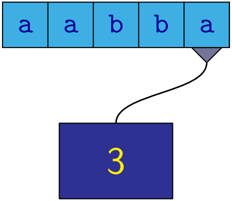


Transformation of NFA to DFA

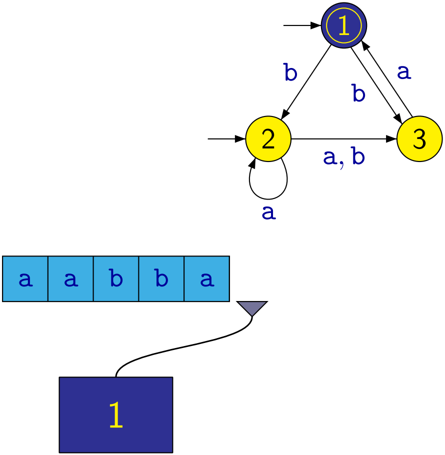


Transformation of NFA to DFA

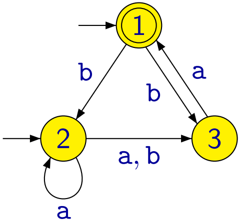




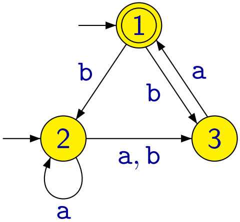
Transformation of NFA to DFA



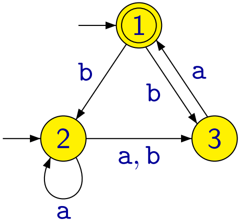
Transformation of NFA to DFA

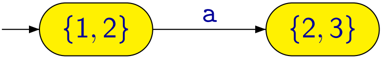


Transformation of NFA to DFA

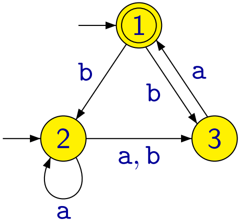


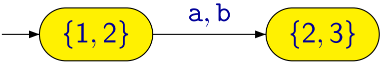
Transformation of NFA to DFA



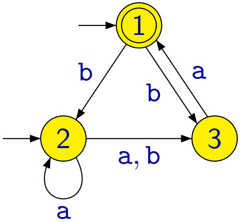


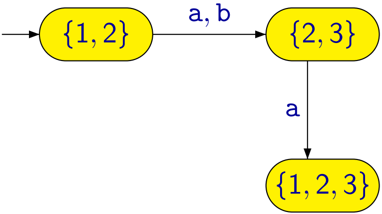
Transformation of NFA to DFA



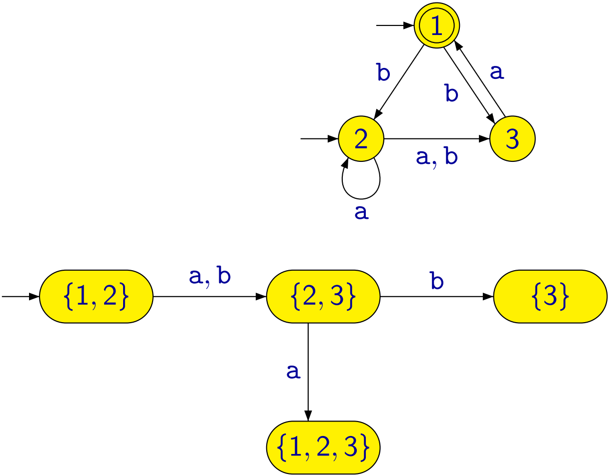


Transformation of NFA to DFA

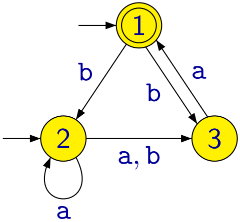


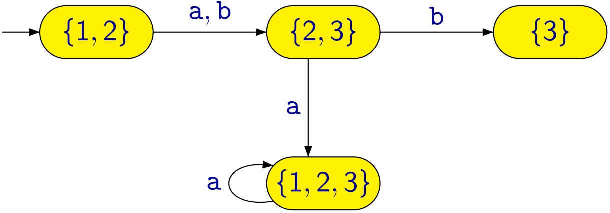


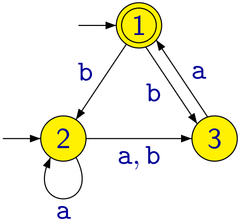
Transformation of NFA to DFA

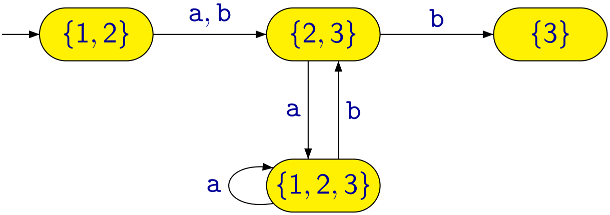


Transformation of NFA to DFA

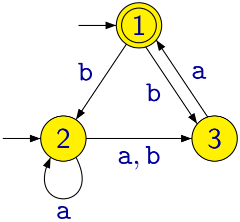


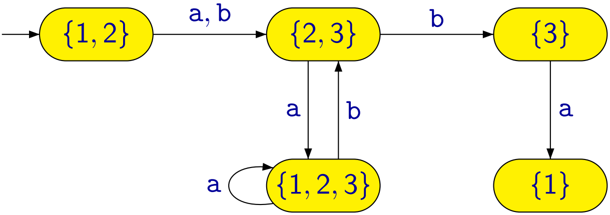



1
2
3
b
b
a
a, b
a

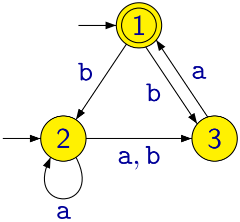

{1, 2}
{2, 3}
{3}
{1, 2, 3}
a, b
b
a
b
a

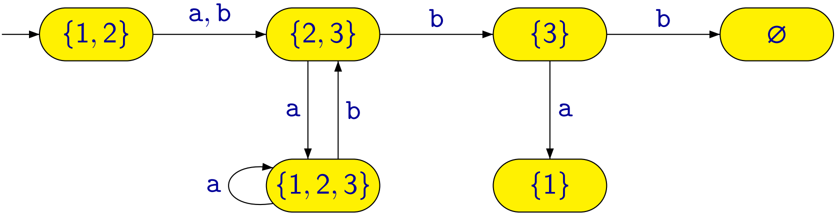
Transformation of NFA to DFA



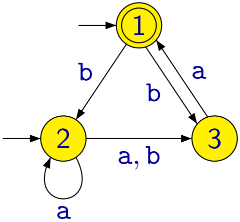


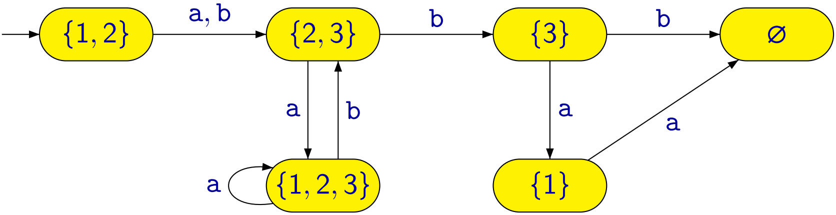
Transformation of NFA to DFA



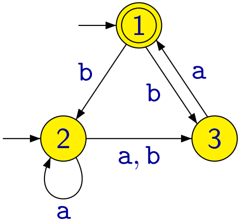


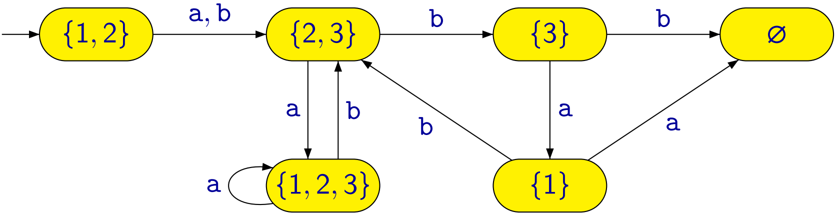
Transformation of NFA to DFA



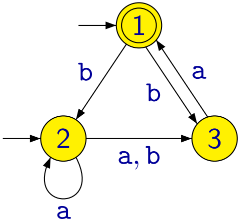


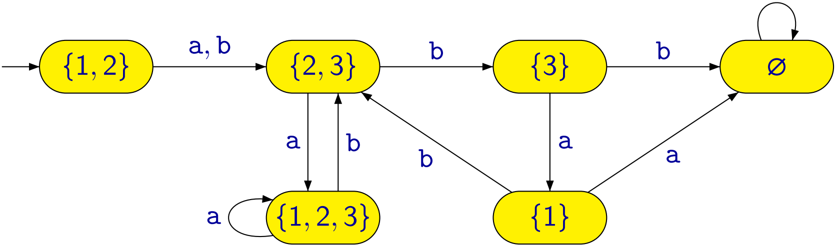
Transformation of NFA to DFA



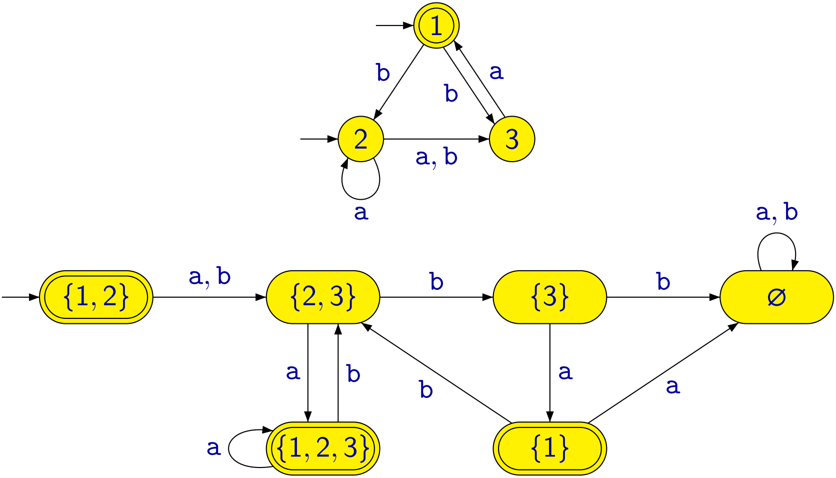


Transformation of NFA to DFA





Transformation of NFA to DFA



Transformation of NFA to DFA

	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

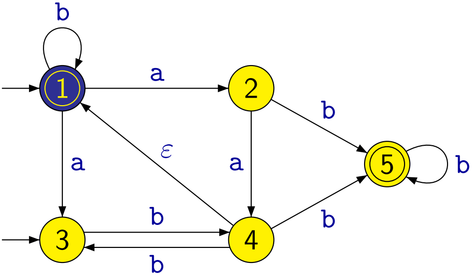
Transformation of NFA to DFA

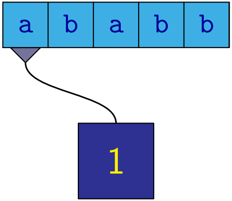
Remark: When a nondeterministic automaton with n states is transformed into a deterministic one, the resulting automaton can have 2^n states.

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.

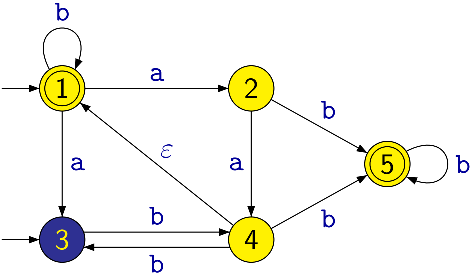
Generalized Nondeterministic Finite Automaton

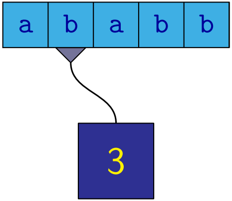




1

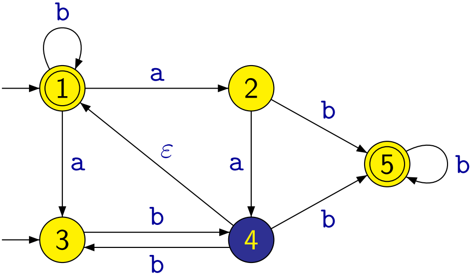
Generalized Nondeterministic Finite Automaton

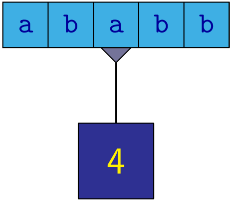




$1 \xrightarrow{\texttt{a}} 3$

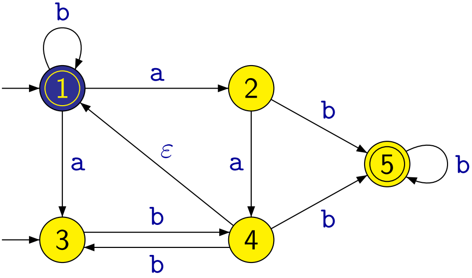
Generalized Nondeterministic Finite Automaton

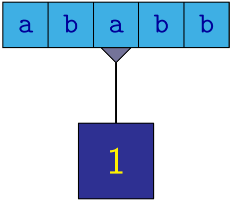




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4$$

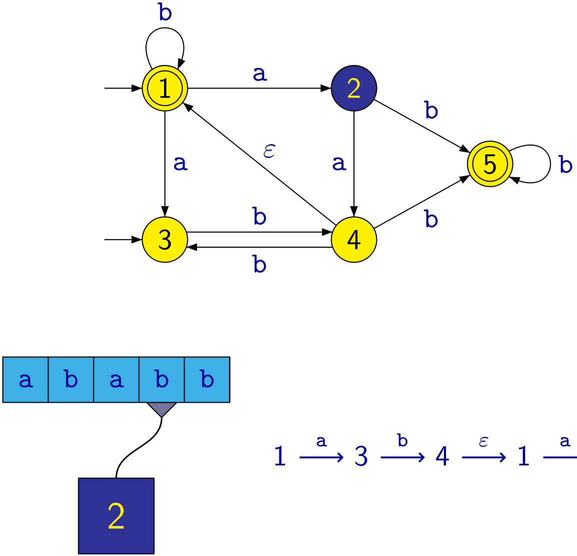
Generalized Nondeterministic Finite Automaton





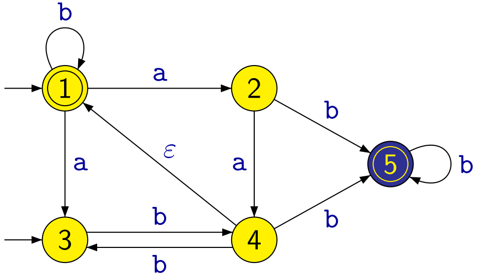
$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\varepsilon} 1$$

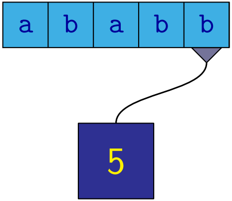
Generalized Nondeterministic Finite Automaton



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

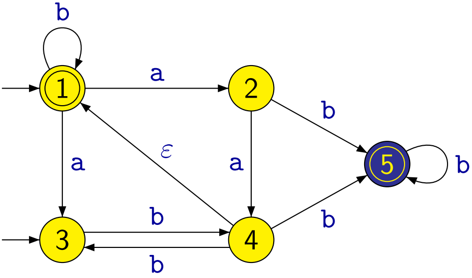
Generalized Nondeterministic Finite Automaton

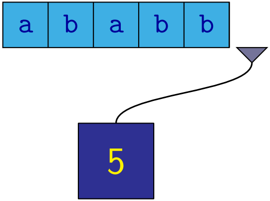




$$1 \xrightarrow{\texttt{a}} 3 \xrightarrow{\texttt{b}} 4 \xrightarrow{\varepsilon} 1 \xrightarrow{\texttt{a}} 2 \xrightarrow{\texttt{b}} 5$$

Generalized Nondeterministic Finite Automaton





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

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

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

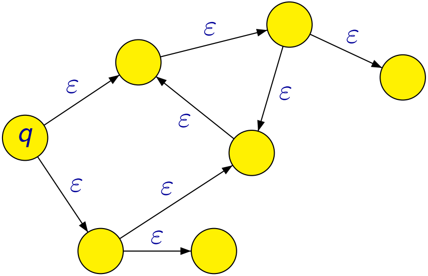
where:

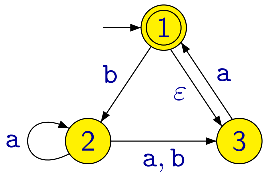
- Q is a finite set of **states**
- Σ is a finite **alphabet**
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ is a **transition function**
- $I \subseteq Q$ is a set of **initial states**
- $F \subseteq Q$ is a set of **accepting states**

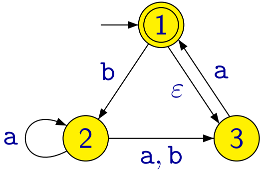
Remark: NFA can be viewed as a special case of GNFA, where $\delta(q, \varepsilon) = \emptyset$ for all $q \in Q$.

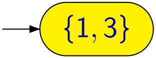
Transformation to a Deterministic Finite Automaton

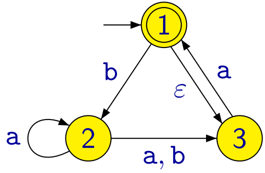
A generalized nondeterministic finite automaton can be transformed into a deterministic one using a similar construction as a nondeterministic finite automaton with the difference that we add to sets of states also all states that are reachable from already added states by some sequence of ε-transitions.

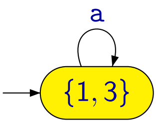


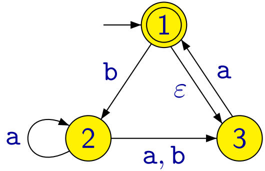

1
b
ε
a
a
2
a, b
3

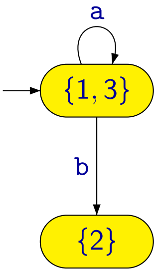

1
2
3
a
b
a
ε
a, b

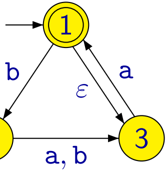

$\{1, 3\}$

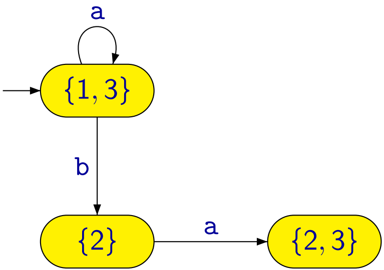

1
b
a
ε
a
2
3
a, b

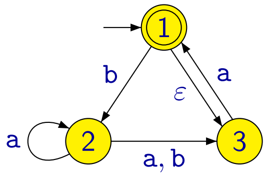

a
{1, 3}

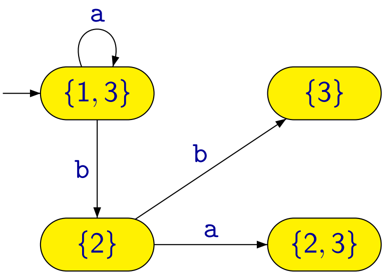

1
b
a
ε
a
2
3
a, b

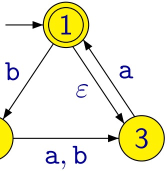

a
{1, 3}
b
{2}

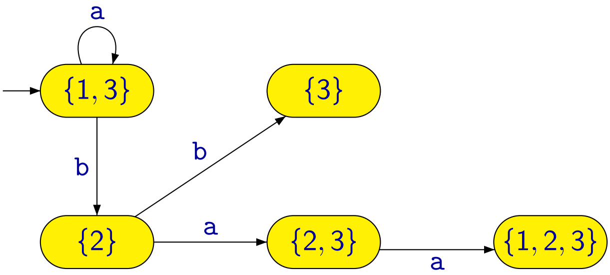

1
b
a
ε
a
2
3
a, b

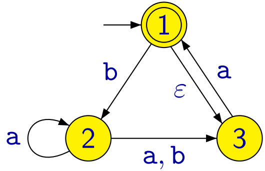

a
{1, 3}
b
{2}
a
{2, 3}

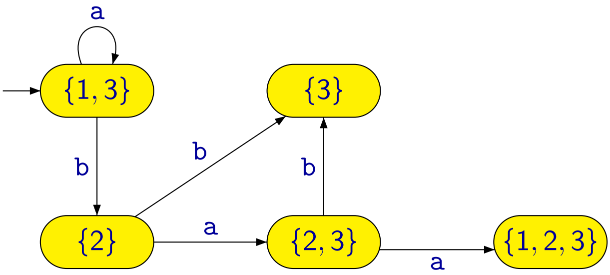

1
b
a
ε
a
2
3
a, b

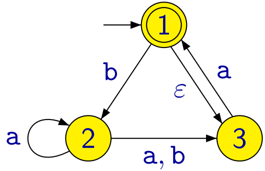

a
{1, 3}
{3}
b
b
{2}
a
{2, 3}

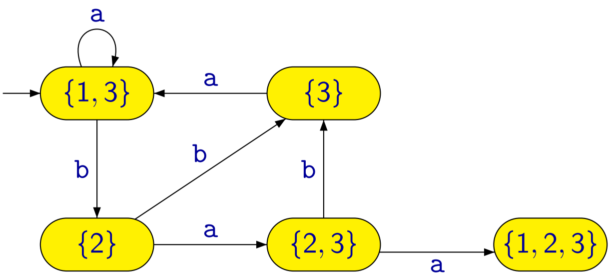

1
2
3
a
b
ε
a
a, b

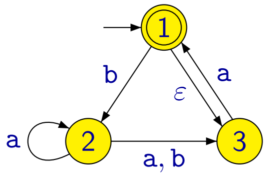

{1, 3}
{3}
{2}
{2, 3}
{1, 2, 3}
a
b
b
a
a

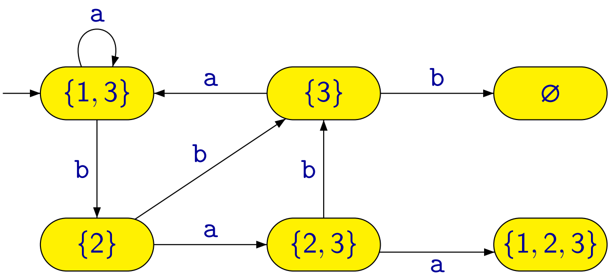

1
2
3
a
b
ε
a
a, b

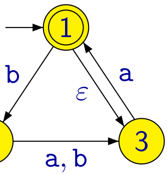

{1, 3}
{3}
{2}
{2, 3}
{1, 2, 3}
a
b
b
b
a
a


1
b
a
ε
a
2
3
a, b

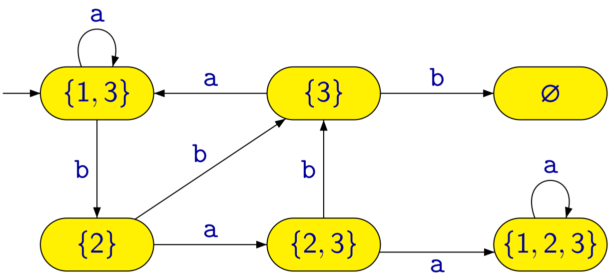

a
a
{1, 3}
{3}
b
b
b
a
{2}
{2, 3}
{1, 2, 3}
a


1
2
3
a
b
ε
a
a, b

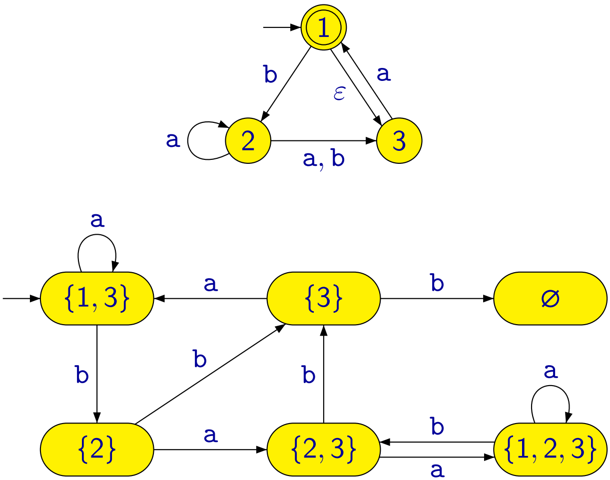

{1, 3}
{3}
∅
{2}
{2, 3}
{1, 2, 3}
a
a
b
b
b
b
a
a

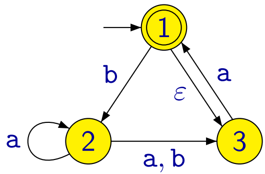


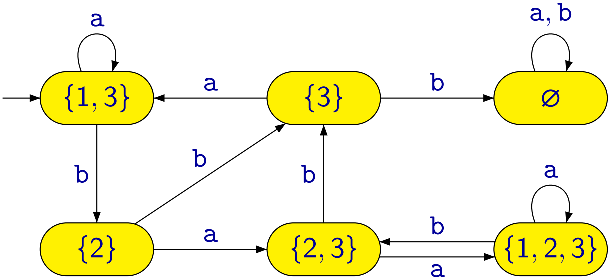
1
b
ε
a
a
2
3
a, b

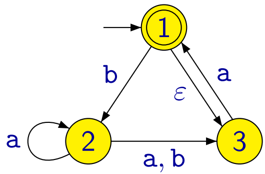


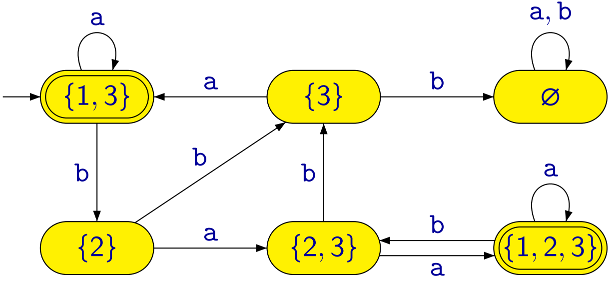
a
{1, 3}
a
{3}
b
∅
b
b
b
a
{2}
a
{2, 3}
a
{1, 2, 3}
a


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


1
2
3
a
b
ε
a
a, b


{1, 3}
{3}
∅
{2}
{2, 3}
{1, 2, 3}
a
a
b
a, b
b
b
b
a
b
a
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 $a \in \Sigma$:

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

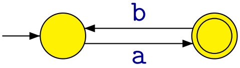
- $q'_0 = Cl_\varepsilon(I)$
- $F' = \{K \in Q' \mid Cl_\varepsilon(K) \cap F \neq \emptyset\}$

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

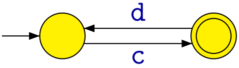
Concatenation of Languages

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

$\mathcal{A}_1$:



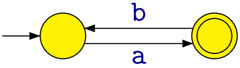
$\mathcal{A}_2$:



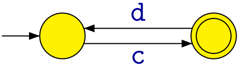
Concatenation of Languages

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

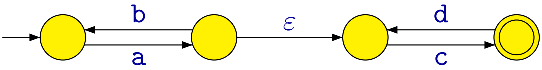
$\mathcal{A}_1$:



$\mathcal{A}_2$:



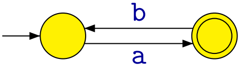
$\mathcal{A}$:



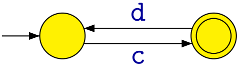
$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$

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

$\mathcal{A}_1$:

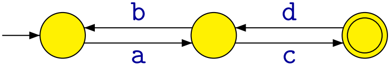


$\mathcal{A}_2$:



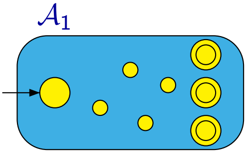
An incorrect construction:

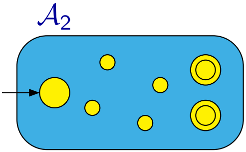
$\mathcal{A}$:



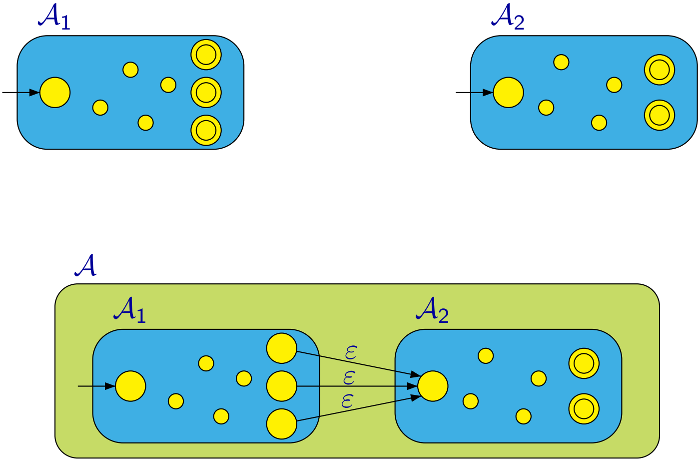
$\mathtt{acdbac} \in \mathcal{L}(\mathcal{A})$ but $\mathtt{acdbac} \notin \mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$

Concatenation of Languages

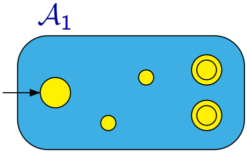




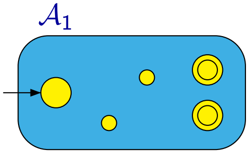
Concatenation of Languages

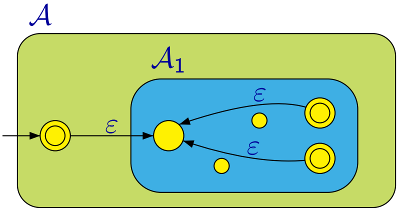


Iteration of a Language

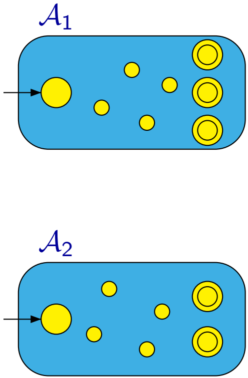


Iteration of a Language



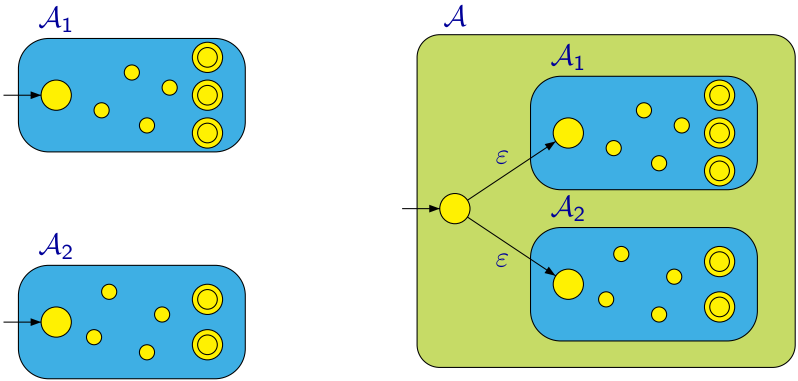


An alternative construction for the union of languages:



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
- ...

Transformation of a Regular Expression to a Finite Automaton

Proposition

Every language that can be represented by a regular expression is regular (i.e., it is accepted by some finite automaton).

Proof: It is sufficient to show how to construct for a given regular expression α a finite automaton accepting the language $\mathcal{L}(\alpha)$.

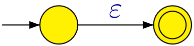
The construction is recursive and proceeds by the structure of the expression α:

- If α is a elementary expression (i.e., $\emptyset$, ε or a):
  - We construct the corresponding automaton directly.
- If α is of the form $(\beta + \gamma)$, $(\beta \cdot \gamma)$ or (β^*):
  - We construct automata accepting languages $\mathcal{L}(\beta)$ and $\mathcal{L}(\gamma)$ recursively.
  - Using these two automata, we construct the automaton accepting the language $\mathcal{L}(\alpha)$.

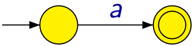
Transformation of a Regular Expression to a Finite Automaton

The automata for the elementary expressions:

$\emptyset$



ε

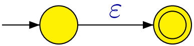


a

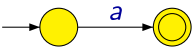
Transformation of a Regular Expression to a Finite Automaton

The automata for the elementary expressions:

$\emptyset$

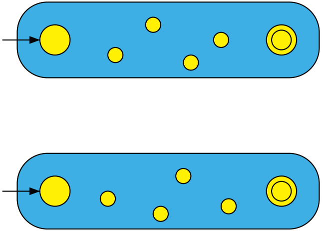


ε



a

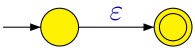
The construction for the union:



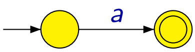
Transformation of a Regular Expression to a Finite Automaton

The automata for the elementary expressions:

$\emptyset$

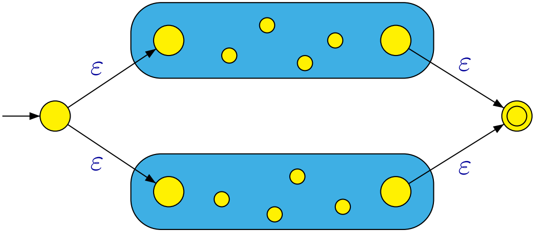


ε



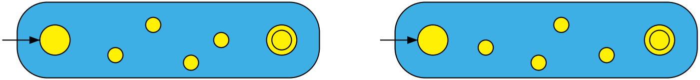
a

The construction for the union:



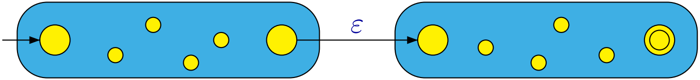
Transformation of a Regular Expression to a Finite Automaton

The construction for the concatenation:



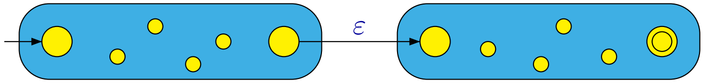
Transformation of a Regular Expression to a Finite Automaton

The construction for the concatenation:

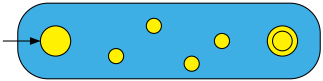


Transformation of a Regular Expression to a Finite Automaton

The construction for the concatenation:

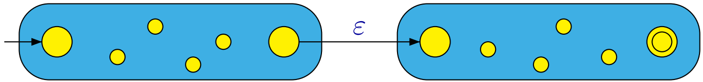


The construction for the iteration:

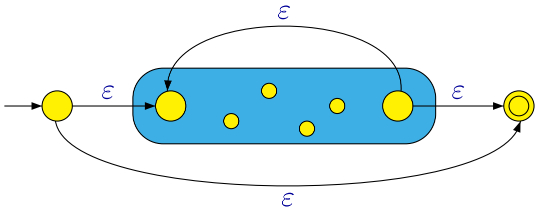


Transformation of a Regular Expression to a Finite Automaton

The construction for the concatenation:



The construction for the iteration:

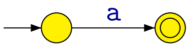


Transformation of a Regular Expression to a Finite Automaton

Example: The construction of an automaton for expression $((\mathtt{a}+\mathtt{b})\cdot\mathtt{b})^*$:

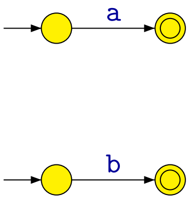
Transformation of a Regular Expression to a Finite Automaton

Example: The construction of an automaton for expression $((\mathtt{a}+\mathtt{b})\cdot\mathtt{b})^*$:



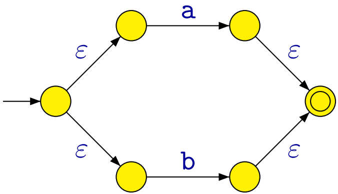
Transformation of a Regular Expression to a Finite Automaton

Example: The construction of an automaton for expression $((\mathtt{a}+\mathtt{b})\cdot\mathtt{b})^*$:



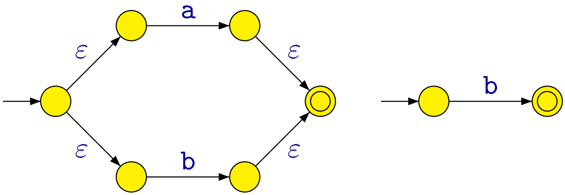
Transformation of a Regular Expression to a Finite Automaton

Example: The construction of an automaton for expression $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:



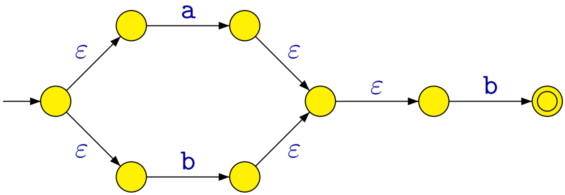
Transformation of a Regular Expression to a Finite Automaton

Example: The construction of an automaton for expression $((\mathtt{a}+\mathtt{b})\cdot\mathtt{b})^*$:



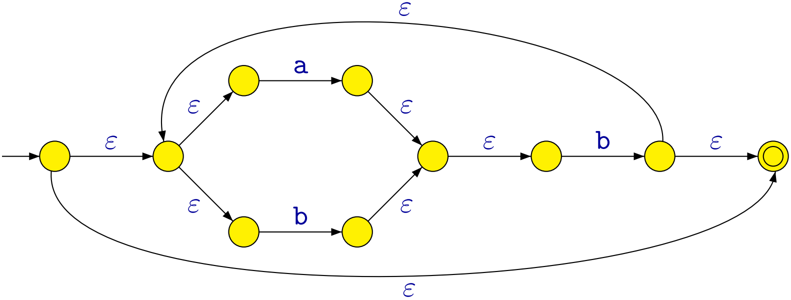
Transformation of a Regular Expression to a Finite Automaton

Example: The construction of an automaton for expression $((\mathtt{a} + \mathtt{b}) \cdot \mathtt{b})^*$:



Transformation of a Regular Expression to a Finite Automaton

Example: The construction of an automaton for expression $((\mathtt{a}+\mathtt{b})\cdot\mathtt{b})^*$:



Transformation of a Regular Expression to a Finite Automaton

If an expression α consists of n symbols (not counting parenthesis) then the resulting automaton has:

- at most $2n$ states,
- at most $4n$ transitions.

Remark: By transforming the generalized nondeterministic automaton into a deterministic one, the number of states can grow exponentially, i.e., the resulting automaton can have up to $2^{2n} = 4^n$ states.

Transformation of an Automaton to a Regular Expression

Proposition

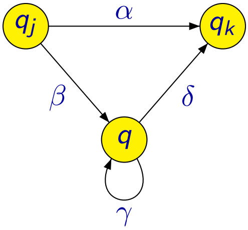
Every regular language can be represented by some regular expression.

Proof: It is sufficient to show how to construct for a given finite automaton $\mathcal{A}$ a regular expression α such that $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{A})$.

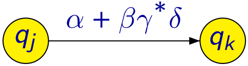
- We modify $\mathcal{A}$ in such a way that ensures it has exactly one initial and exactly one accepting state.
- Its states will be removed one by one.
- Its transitions will be labelled with regular expressions.
- The resulting automaton will have only two states – the initial and the accepting, and only one transition labelled with the resulting regular expression.

Transformation of an Automaton to a Regular Expression

The main idea: If a state q is removed, for every pair of remaining states q_j, q_k we extend the label on a transition from q_j to q_k by a regular expression representing paths from q_j to q_k going through q.

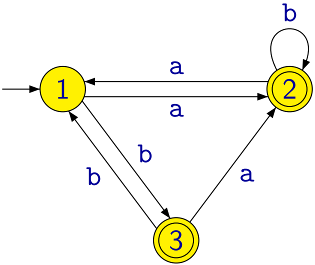


After removing of the state q:



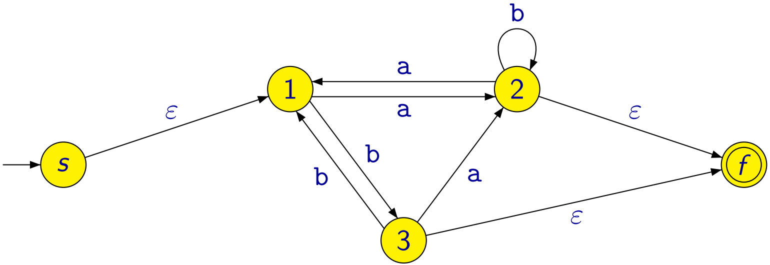
Transformation of an Automaton to a Regular Expression

Example:

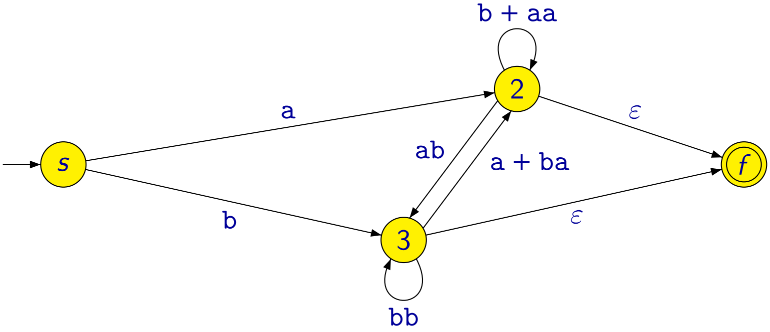


Transformation of an Automaton to a Regular Expression

Example:

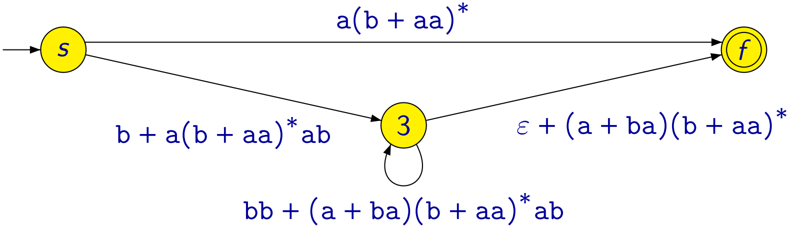


Example:



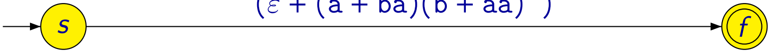
Transformation of an Automaton to a Regular Expression

Example:



Example:

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



Equivalence of Finite Automata and Regular Expressions

Theorem

A language is regular iff it can be represented by a regular expression.

Nonregular Languages

Not all languages are regular.

There are languages for which there exist no finite automata accepting them.

Examples of nonregular languages:

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

Remark: The existence of nonregular languages is already apparent from the fact that there are only countably many (nonisomorphic) automata working over some alphabet Σ but there are uncountably many languages over the alphabet Σ.

Nonregular Languages

How to prove that some language L is not regular?

A language is not regular if there is no automaton (i.e., it is not possible to construct an automaton) accepting the language.

But how to prove that something does not exist?

Nonregular Languages

How to prove that some language L is not regular?

A language is not regular if there is no automaton (i.e., it is not possible to construct an automaton) accepting the language.

But how to prove that something does not exist?

The answer: By contradiction.

E.g., we can assume there is some automaton $\mathcal{A}$ accepting the language L, and show that this assumption leads to a contradiction.

Nonregular Languages

We show that language $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

The proof by contradiction.

Let us assume there exists a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ such that $\mathcal{L}(\mathcal{A}) = L$.

We show that language $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

The proof by contradiction.

Let us assume there exists a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ such that $\mathcal{L}(\mathcal{A}) = L$.

Let $|Q| = n$.

Nonregular Languages

We show that language $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

The proof by contradiction.

Let us assume there exists a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ such that $\mathcal{L}(\mathcal{A}) = L$.

Let $|Q| = n$.

Consider word $z = a^n b^n$.

Nonregular Languages

We show that language $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

The proof by contradiction.

Let us assume there exists a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ such that $\mathcal{L}(\mathcal{A}) = L$.

Let $|Q| = n$.

Consider word $z = a^n b^n$.

Since $z \in L$, there must be an accepting computation of the automaton $\mathcal{A}$

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

where q_0 is an initial state, and $q_{2n} \in F$.

Nonregular Languages

Consider now the first $n + 1$ states of the computation

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

i.e., the sequence of states $q_0, q_1, \ldots, q_n$.

It is obvious that all states in this sequence can not be pairwise different, since $|Q| = n$ and the sequence has $n + 1$ elements.

This means that there exists a state $q \in Q$ which occurs (at least) twice in the sequence.

Nonregular Languages

Consider now the first $n+1$ states of the computation

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

i.e., the sequence of states $q_0, q_1, \ldots, q_n$.

It is obvious that all states in this sequence can not be pairwise different, since $|Q| = n$ and the sequence has $n+1$ elements.

This means that there exists a state $q \in Q$ which occurs (at least) twice in the sequence.

It is an application of so called **pigeonhole principle**.

Pigeonhole principle

If we have $n+1$ pigeons in n holes then there is at least one hole containing at least two pigeons.

Nonregular Languages

Consider now the first $n + 1$ states of the computation

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

i.e., the sequence of states $q_0, q_1, \ldots, q_n$.

It is obvious that all states in this sequence can not be pairwise different, since $|Q| = n$ and the sequence has $n + 1$ elements.

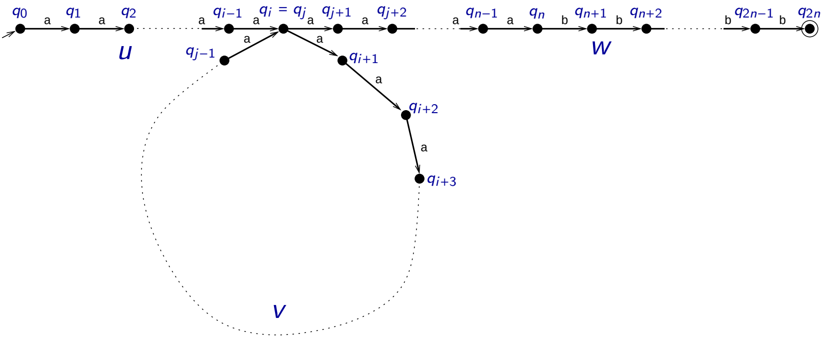
This means that there exists a state $q \in Q$ which occurs (at least) twice in the sequence.

I.e., there are indexes i, j such that $0 \leq i < j \leq n$ and

$$q_i = q_j$$

which means that the automaton $\mathcal{A}$ must go through a cycle when reading the symbols a in the word $z = a^n b^n$.

Nonregular Languages



The word $z = a^n b^n$ can be divided into three parts u, v, w such that $z = uvw$:

$$u = a^i \qquad v = a^{j-i} \qquad w = a^{n-j} b^n$$

Nonregular Languages

For the words $u = a^i$, $v = a^{j-i}$, and $w = a^{n-j} b^n$ we have

$$q_0 \xrightarrow{u} q_i \qquad q_i \xrightarrow{v} q_j \qquad q_j \xrightarrow{w} q_{2n}$$

Let r be the length of the word v, i.e., $r = j - i$ (obviously $r > 0$, due to $i < j$).

Since $q_i = q_j$, the automaton accepts word $uw = a^{n-r} b^n$ that does not belong to L:

$$q_0 \xrightarrow{u} q_i \xrightarrow{w} q_{2n}$$

The word $uvvw = a^{n+r} b^n$, that also does not belong to L, is accepted too:

$$q_0 \xrightarrow{u} q_i \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{w} q_{2n}$$

Nonregular Languages

Similarly we can show that every word of the form $uvvvv\cdots vvw$, i.e., of the form uv^kw for some $k \geq 0$, is accepted by the automaton $\mathcal{A}$:

$$q_0 \xrightarrow{u} q_i \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{v} \cdots \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{w} q_{2n}$$

A word of the form uv^kw looks as follows: $a^{n-r+rk}b^n$.

Since $r > 0$, the following equivalence holds only for $k = 1$:

$$n - r + rk = n$$

This means that if $k \neq 1$ then uv^kw does not belong to the language L.

However, the automaton $\mathcal{A}$ accepts each such word, which is a contradiction with the assumption that $\mathcal{L}(\mathcal{A}) = \{a^nb^n \mid n \geq 0\}$.