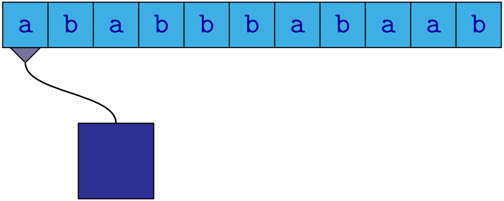
Finite Automata

Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

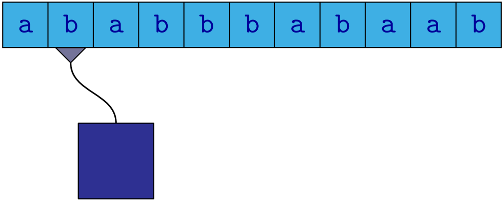


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

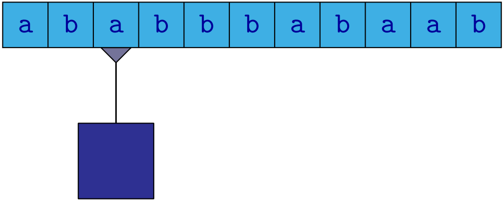


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

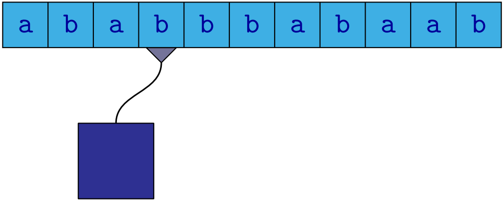


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

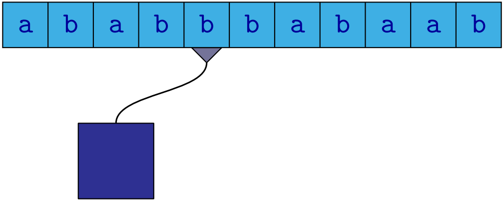


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

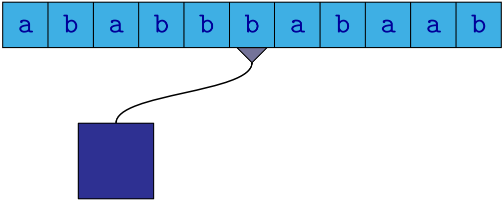


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

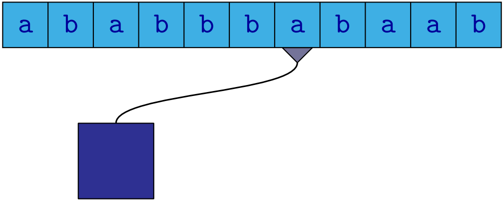


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

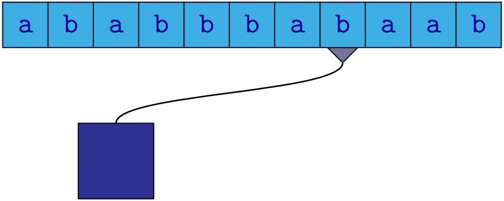


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

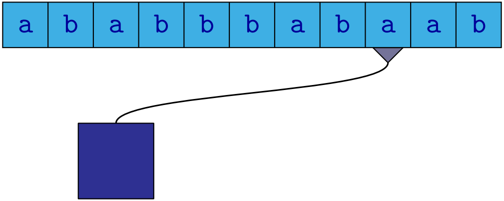


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

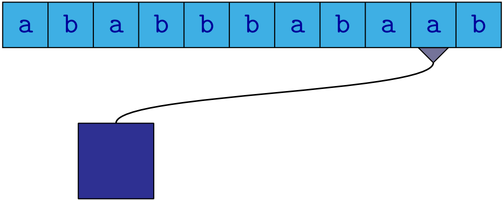


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

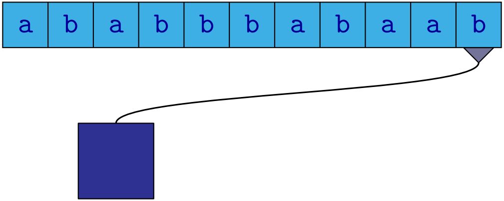


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.

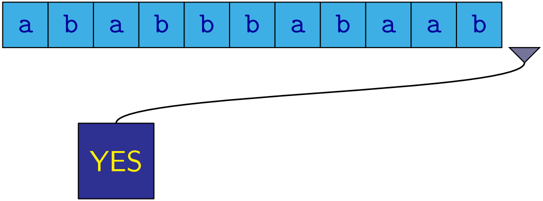


Recognition of a Language

Example: Consider words over alphabet $\{\mathtt{a}, \mathtt{b}\}$.

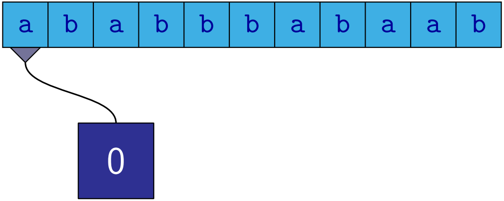
We would like to recognize a language L consisting of words with even number of symbols $\mathtt{b}$.

We want to design a device that reads a word and then tells us if the word belongs to the language L or not.



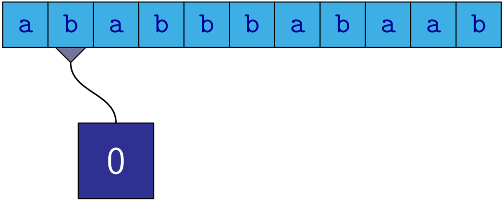
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.

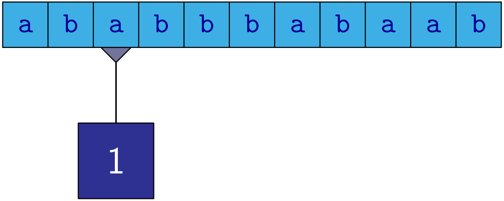


Recognition of a Language

The first idea: To count the number of occurrences of symbol `b`.

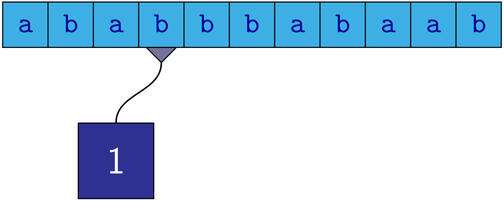


The first idea: To count the number of occurrences of symbol b.



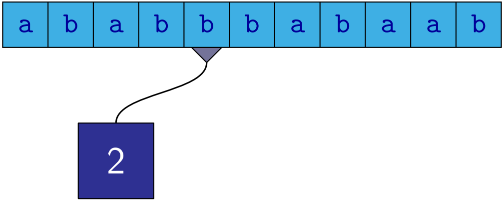
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



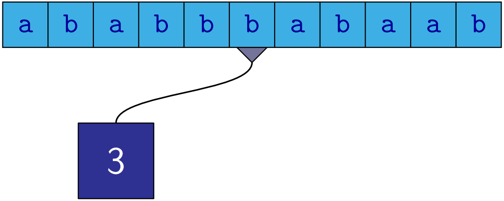
Recognition of a Language

The first idea: To count the number of occurrences of symbol `b`.



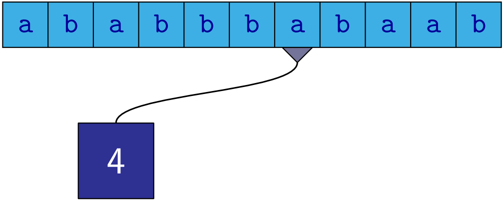
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



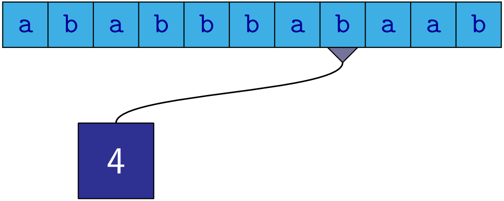
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



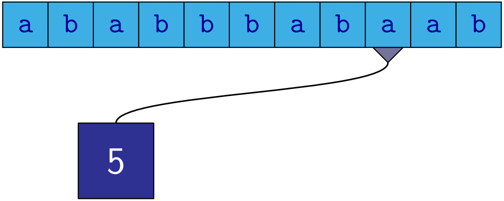
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



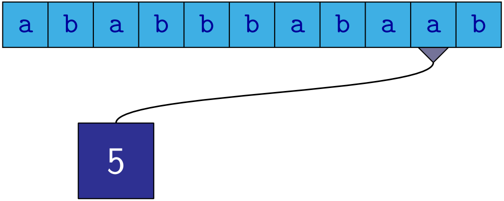
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



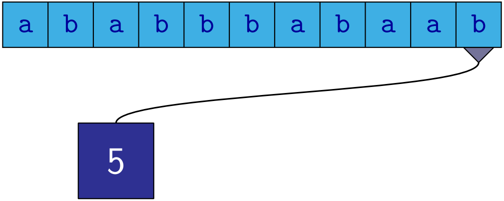
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



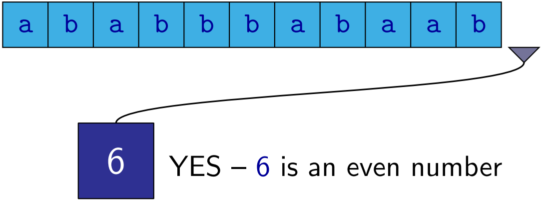
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



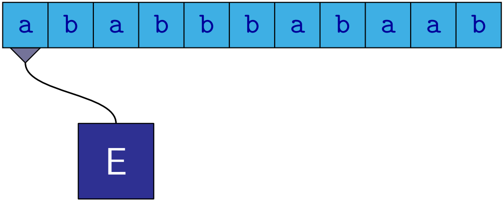
Recognition of a Language

The first idea: To count the number of occurrences of symbol b.



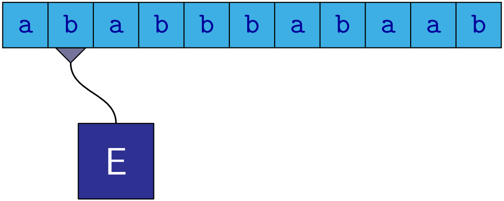
Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols b read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



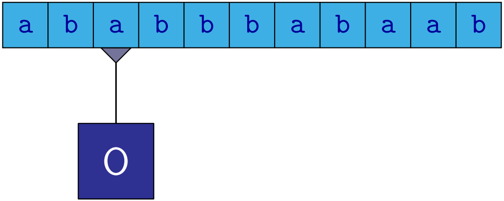
Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



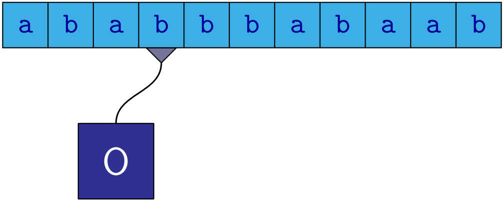
Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).

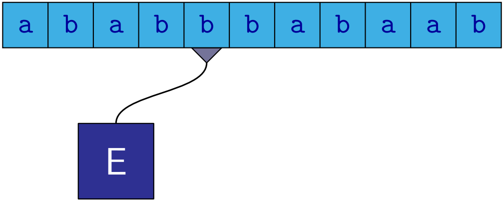


Recognition of a Language

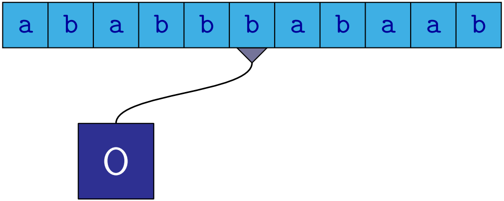
The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



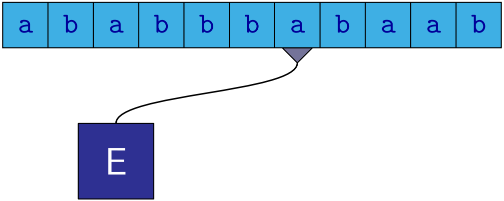
The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).

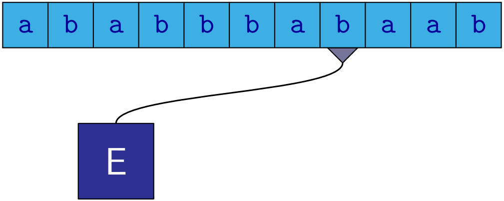


The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



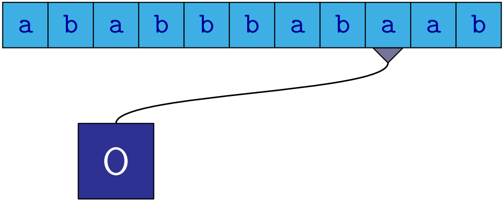
Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



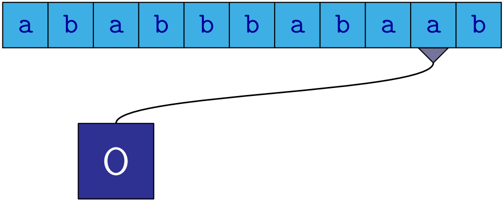
Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols b read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



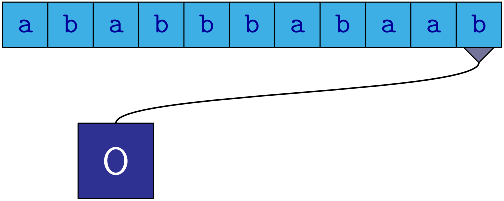
Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols b read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



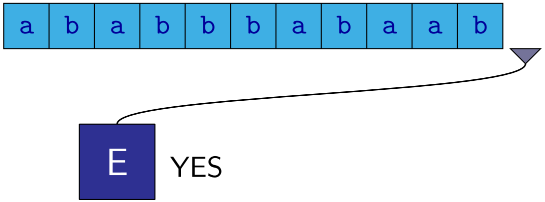
Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



Recognition of a Language

The second idea: In fact, we just need to remember if the number of symbols `b` read so far is even or odd (i.e., it is sufficient to remember only the last bit of the number).



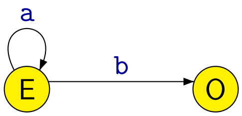
Recognition of a Language

The behaviour of the device can be described by the following graph:

Recognition of a Language

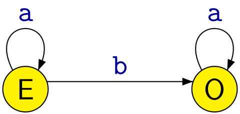
The behaviour of the device can be described by the following graph:

The behaviour of the device can be described by the following graph:



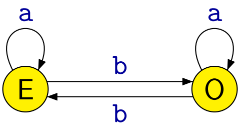
Recognition of a Language

The behaviour of the device can be described by the following graph:



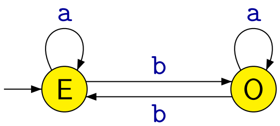
Recognition of a Language

The behaviour of the device can be described by the following graph:



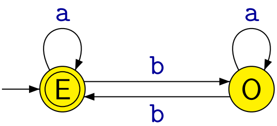
Recognition of a Language

The behaviour of the device can be described by the following graph:



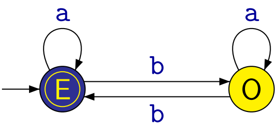
Recognition of a Language

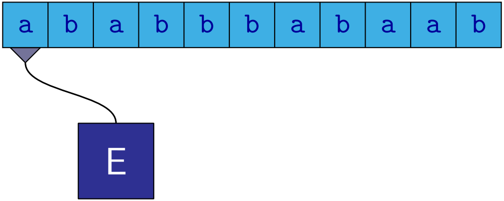
The behaviour of the device can be described by the following graph:



Recognition of a Language

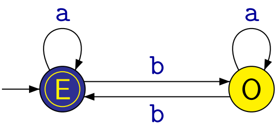
The behaviour of the device can be described by the following graph:

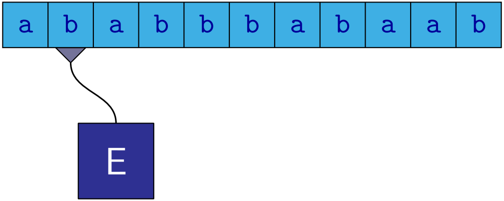




Recognition of a Language

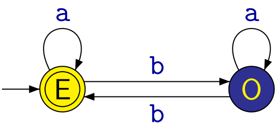
The behaviour of the device can be described by the following graph:

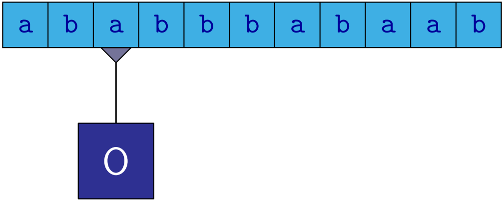




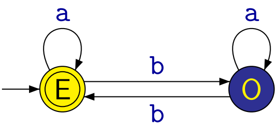
Recognition of a Language

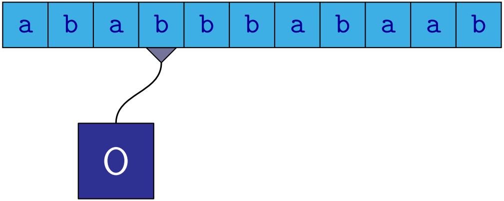
The behaviour of the device can be described by the following graph:





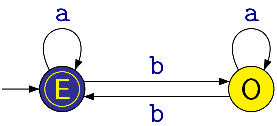
The behaviour of the device can be described by the following graph:

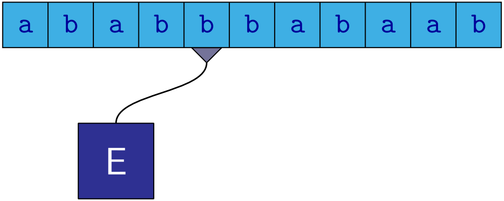




Recognition of a Language

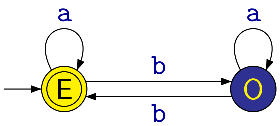
The behaviour of the device can be described by the following graph:

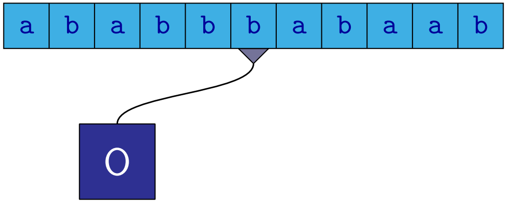




Recognition of a Language

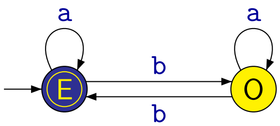
The behaviour of the device can be described by the following graph:

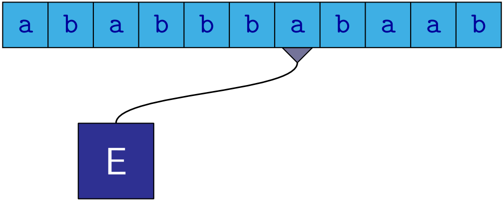




Recognition of a Language

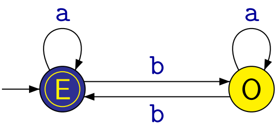
The behaviour of the device can be described by the following graph:

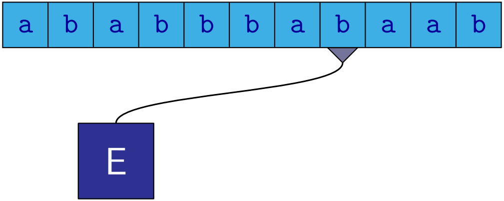




Recognition of a Language

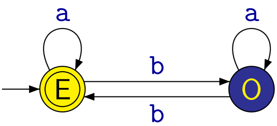
The behaviour of the device can be described by the following graph:

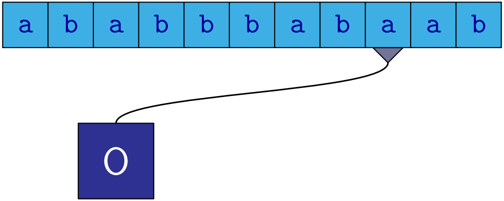




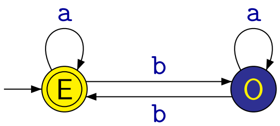
Recognition of a Language

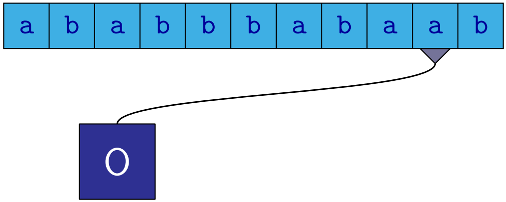
The behaviour of the device can be described by the following graph:





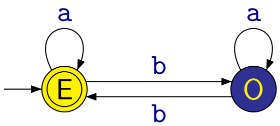
The behaviour of the device can be described by the following graph:

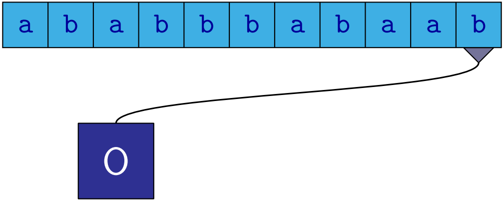




Recognition of a Language

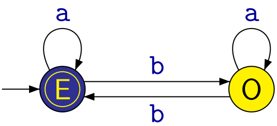
The behaviour of the device can be described by the following graph:

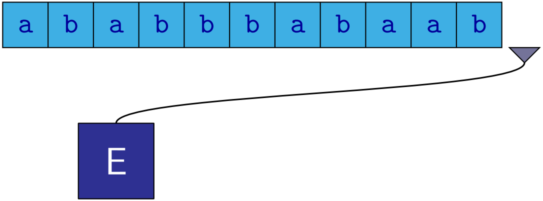




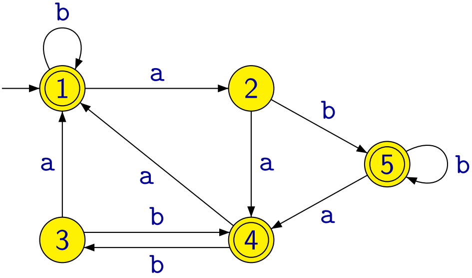
Recognition of a Language

The behaviour of the device can be described by the following graph:





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

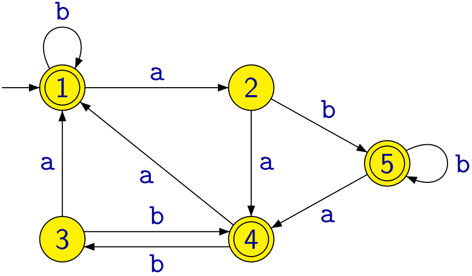
Formally, a **deterministic finite automaton** (**DFA**) is defined as a tuple

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

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

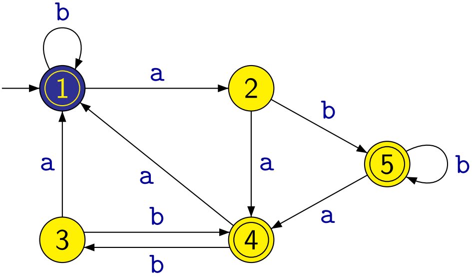
Instead of

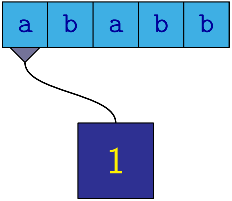
$$\begin{array}{ll} \delta(1,\mathtt{a}) = 2 & \delta(1,\mathtt{b}) = 1 \\ \delta(2,\mathtt{a}) = 4 & \delta(2,\mathtt{b}) = 5 \\ \delta(3,\mathtt{a}) = 1 & \delta(3,\mathtt{b}) = 4 \\ \delta(4,\mathtt{a}) = 1 & \delta(4,\mathtt{b}) = 3 \\ \delta(5,\mathtt{a}) = 4 & \delta(5,\mathtt{b}) = 5 \end{array}$$

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

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

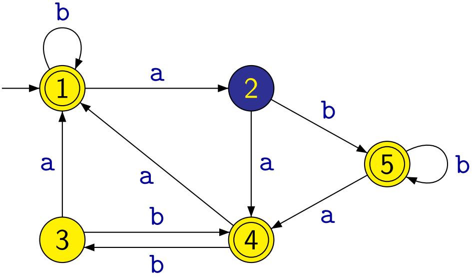
Deterministic Finite Automaton

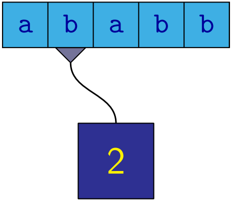




1

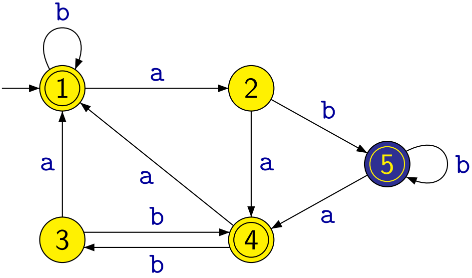
Deterministic Finite Automaton

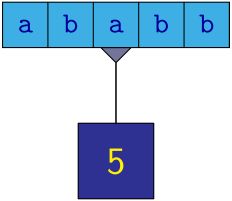




$1 \xrightarrow{a} 2$

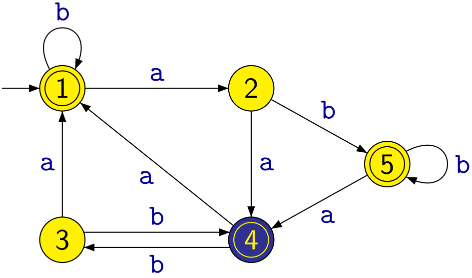
Deterministic Finite Automaton

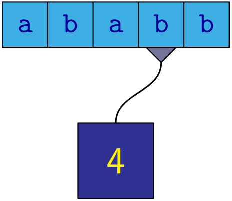




$$1 \xrightarrow{\texttt{a}} 2 \xrightarrow{\texttt{b}} 5$$

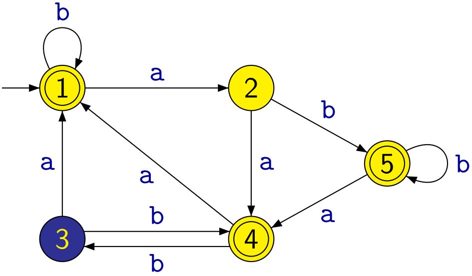
Deterministic Finite Automaton

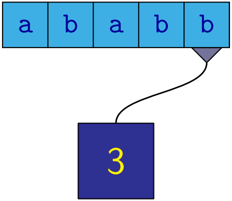




$$1 \xrightarrow{\texttt{a}} 2 \xrightarrow{\texttt{b}} 5 \xrightarrow{\texttt{a}} 4$$

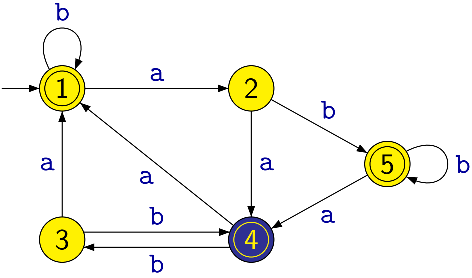
Deterministic Finite Automaton

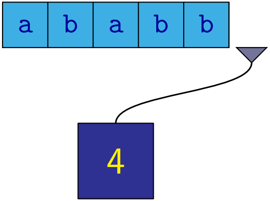




$$1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{a} 4 \xrightarrow{b} 3$$

Deterministic Finite Automaton





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

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

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

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

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

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

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

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

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

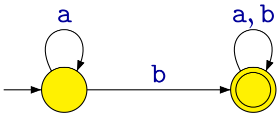
Definition

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

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.,

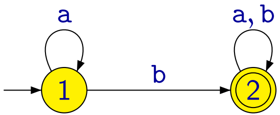
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} \geq 1\}$$



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.,

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

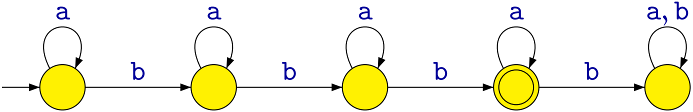


	a	b
$\rightarrow 1$	1	2
$\leftarrow 2$	2	2

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 $\mathtt{b}$, i.e.,

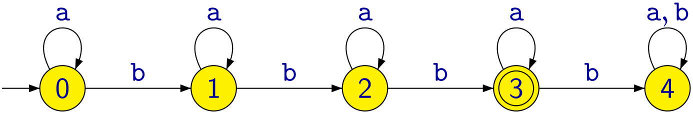
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} = 3\}$$



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 $\mathtt{b}$, i.e.,

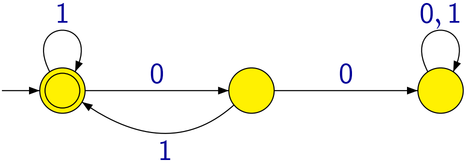
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{b}} = 3\}$$



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

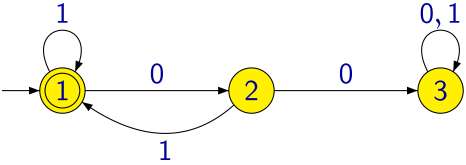
Examples of Deterministic Finite Automata

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.



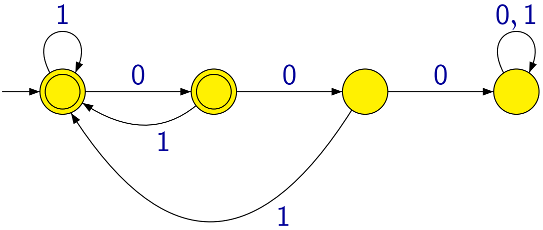
Examples of Deterministic Finite Automata

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.



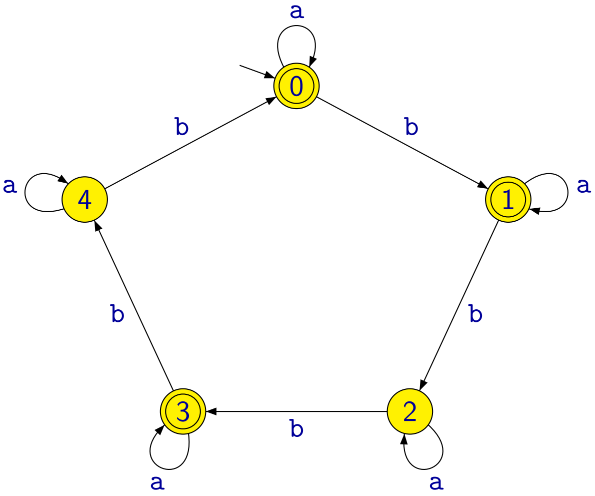
	0	1
$\leftrightarrow 1$	2	1
2	3	1
3	3	3

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.

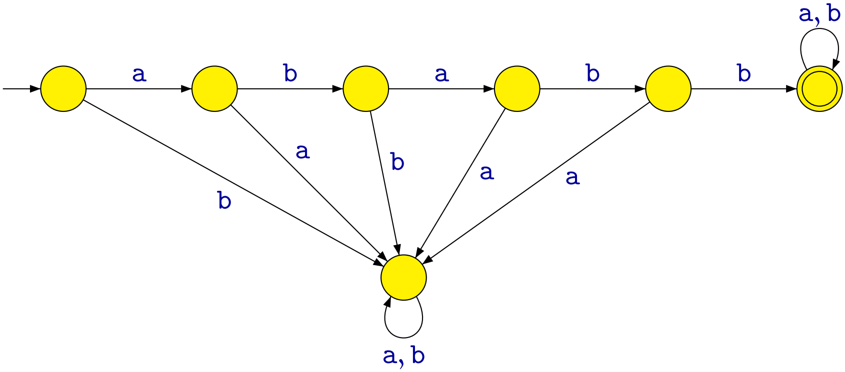


Example: An automaton recognizing the language

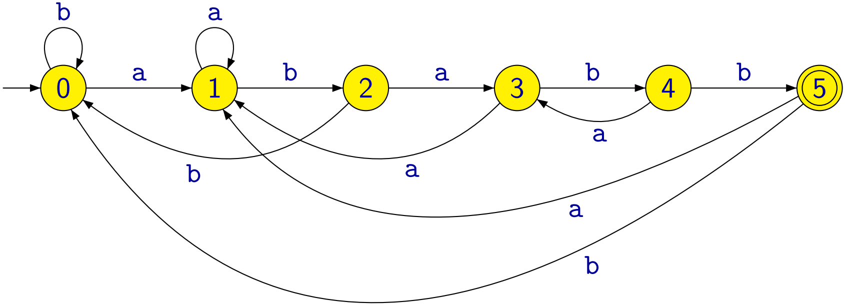
$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid (|w|_{\mathtt{b}} \bmod 5) \in \{0, 1, 3\}\}$$



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

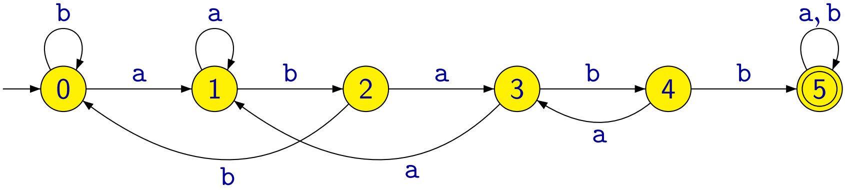
The construction of this automaton is based on the following idea:

- 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`

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



Other Examples of Finite-state Systems

- The automata for searching a given suffix or a subword can be used for efficient searching in a text.

  Several efficient algorithms for text searching are based on this idea, e.g.,

  - Knuth-Morris-Pratt
  - Aho-Corasick
  - ...

- Modelling of behaviour of an object in object-oriented programming (OOP):

  - States — an internal state of an object is determined by values of its attributes
  - Alphabet — names of methods

Other Examples of Finite-state Systems

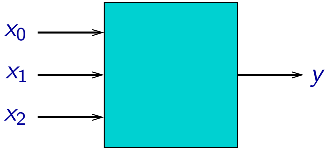
- Description of behaviour of communication network protocols:
  - Alphabet — different kinds of messages (e.g., a request for establishing a connection, a packet with data, an acknoledgement, a request for closing connection, . . .)

  Concrete example: The specification of Transmission Control Protocol (TCP) in the family of TCP/IP protocols contains a finite-state diagram describing the behaviour of this protocol.

- In the HTML5 specification, the process of so called tokenization of an html file (i.e., recognizing tags, attributes, etc.) is described in a form of a finite-state machine reading individual characters from an input file.

Other Examples of Finite-state Systems

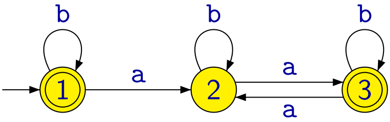
- A sequential hardware circuit

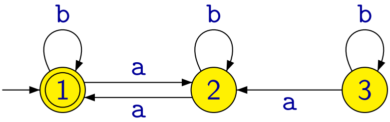


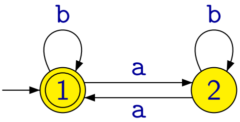
A circuit with n input wires can be viewed as an automaton working with an alphabet whose symbols are n-tuples of zeros and ones.

$$\left\{ \begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}, \begin{pmatrix}1\\1\\1\end{pmatrix} \right\}$$

Equivalence of Automata







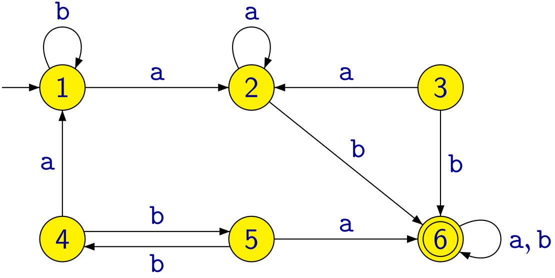
All three automata accept the language of all words with an even number of a's.

Equivalence of Automata

Definition

We say automata $\mathcal{A}_1$, $\mathcal{A}_2$ are **equivalent** if $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

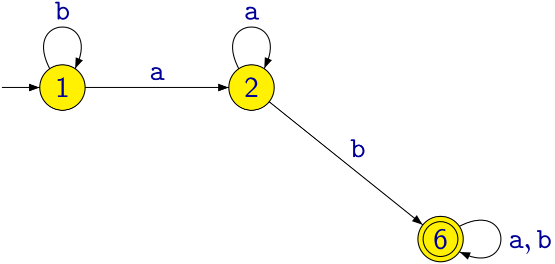
Unreachable States of an Automaton



- The automaton accepts the language
  $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ contains subword } \mathtt{ab}\}$
- There is no input sequence such that after reading it, the automaton gets to states 3, 4, or 5.

.

Unreachable States of an Automaton



- The automaton accepts the language
  $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ contains subword } \mathtt{ab}\}$
- There is no input sequence such that after reading it, the automaton gets to states 3, 4, or 5.
- If we remove these states, the automaton still accepts the same language L.

Unreachable States of an Automaton

Definition

A state q of a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is **reacheable** if there exists a word w such that $q_0 \xrightarrow{w} q$.

Otherwise the state is **unreachable**.

- There is no path in a graph of an automaton going from the initial state to some unreachable state.
- Unreachable states can be removed from an automaton (together with all transitions going to them and from them). The language accepted by the automaton is not affected.

Automaton and Operations on Languages

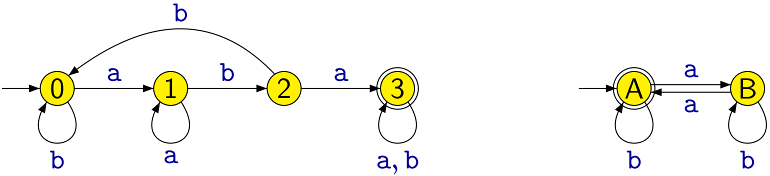
When we construct automata, it can be difficult to construct an automaton for a given language L directly.

If it is possible to describe the language L as a result of some language operations (intersection, union, concatenation, iteration, ...) applied to some simpler languages L_1 and L_2, then it can be easier to proceed in a modular manner:

- To construct automata for languages L_1 and L_2.
- Then to use some of general constructions that allow to algorithmically construct an automaton for language L, which is a result of applying a given language operation on languages L_1 and L_2, from automata for languages L_1 and L_2.

An Automaton for Intersection of Languages

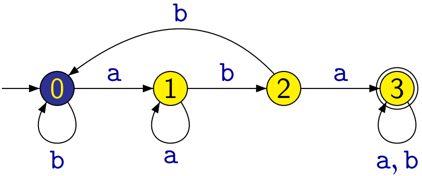
Let us have the following two automata:

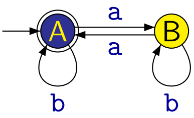


Do both of them accept the word ababb?

An Automaton for Intersection of Languages

Let us have the following two automata:

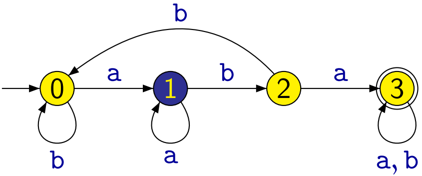


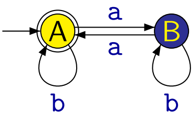


Do both of them accept the word ababb?

An Automaton for Intersection of Languages

Let us have the following two automata:

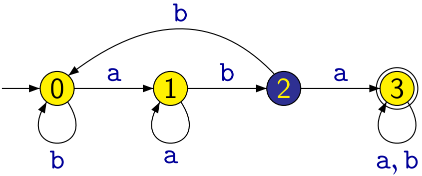


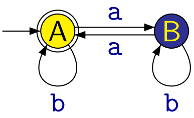


Do both of them accept the word ababb?

An Automaton for Intersection of Languages

Let us have the following two automata:

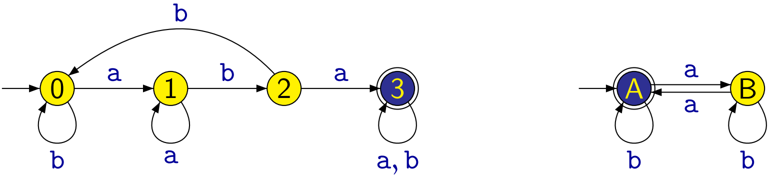




Do both of them accept the word ababb?

An Automaton for Intersection of Languages

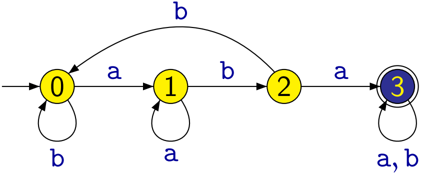
Let us have the following two automata:

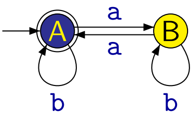


Do both of them accept the word $\texttt{ababb}$?

An Automaton for Intersection of Languages

Let us have the following two automata:

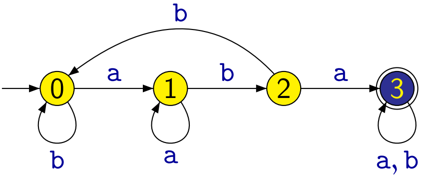


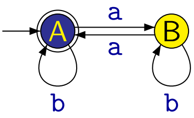


Do both of them accept the word ababb?

An Automaton for Intersection of Languages

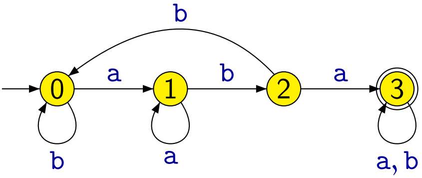
Let us have the following two automata:

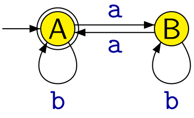




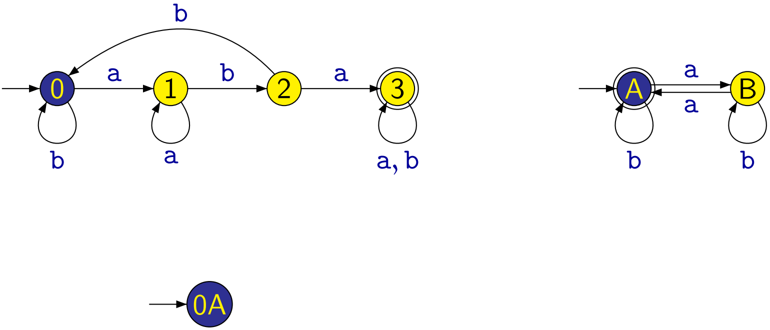
Do both of them accept the word ababb?

An Automaton for Intersection of Languages

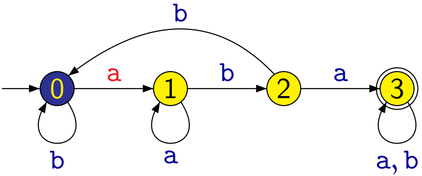


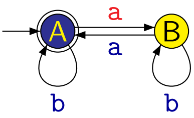


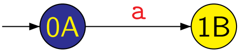
An Automaton for Intersection of Languages



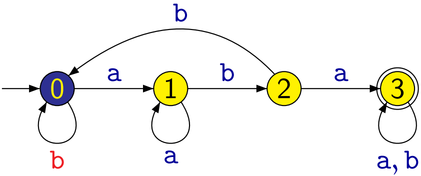
An Automaton for Intersection of Languages

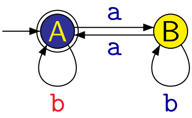


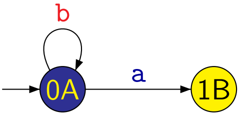




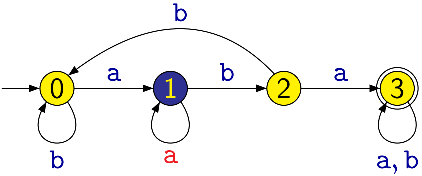
An Automaton for Intersection of Languages

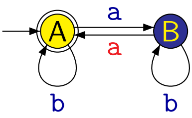


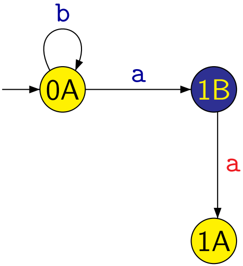




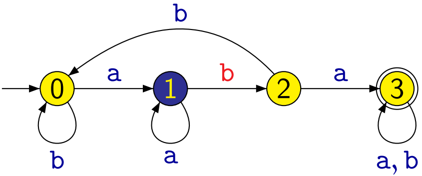
An Automaton for Intersection of Languages

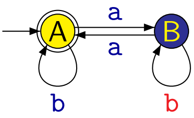


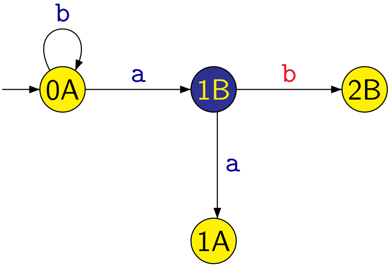




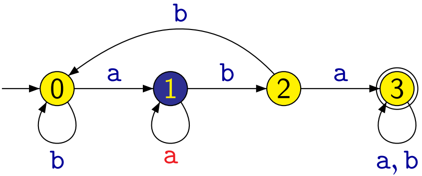
An Automaton for Intersection of Languages

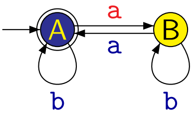


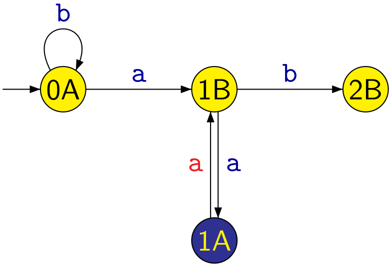




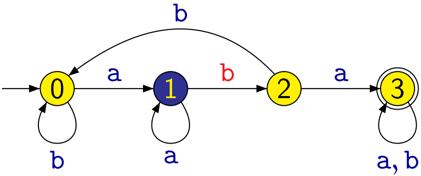
An Automaton for Intersection of Languages

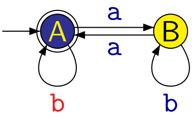


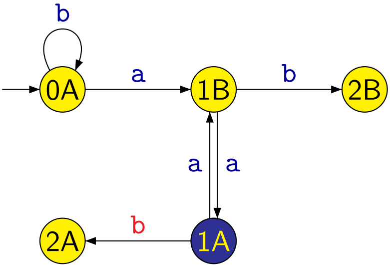




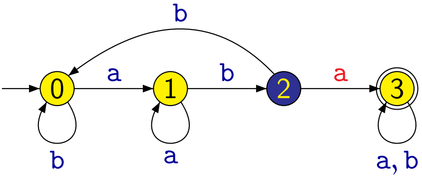
An Automaton for Intersection of Languages

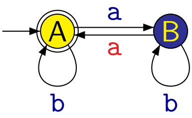


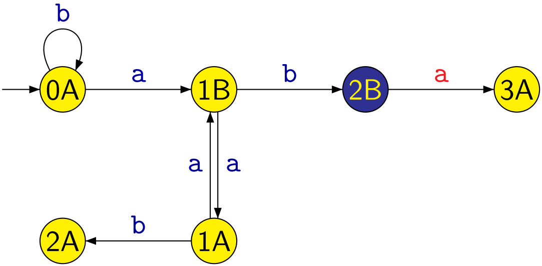




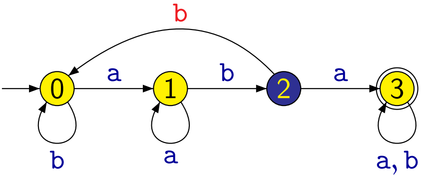
An Automaton for Intersection of Languages

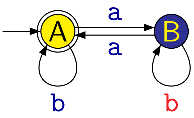


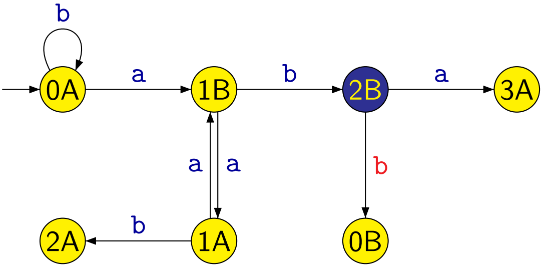




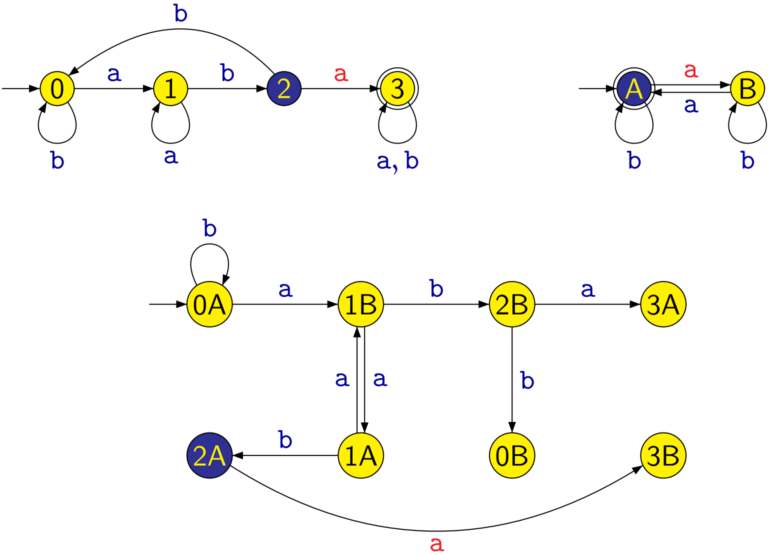
An Automaton for Intersection of Languages



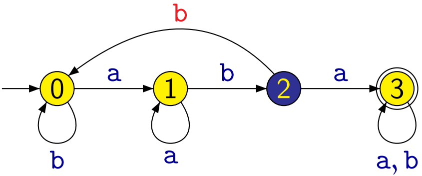


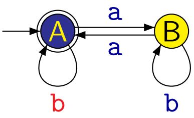


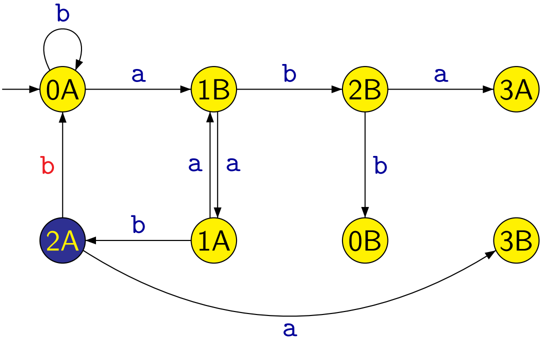
An Automaton for Intersection of Languages



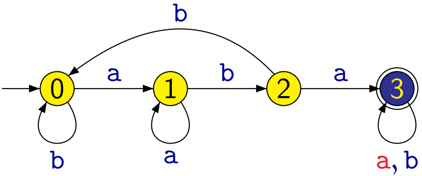
An Automaton for Intersection of Languages

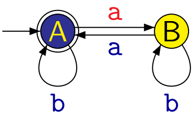


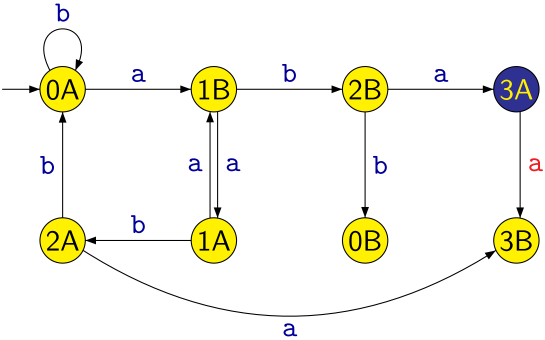




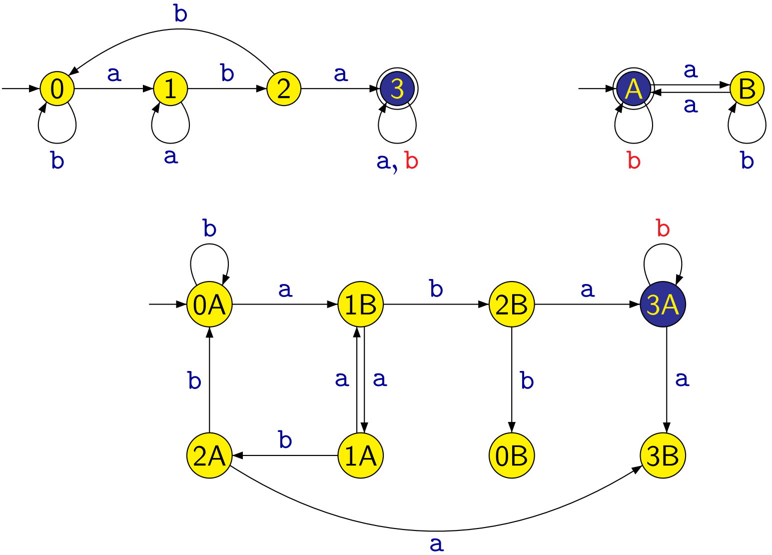
An Automaton for Intersection of Languages



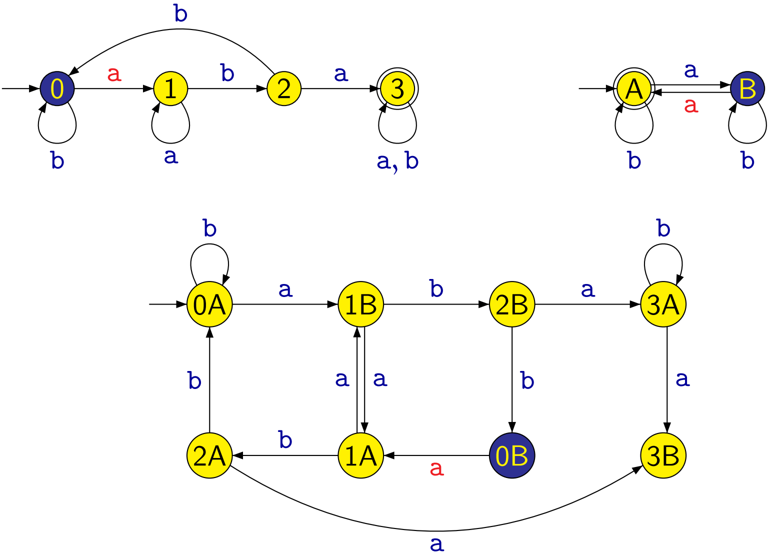




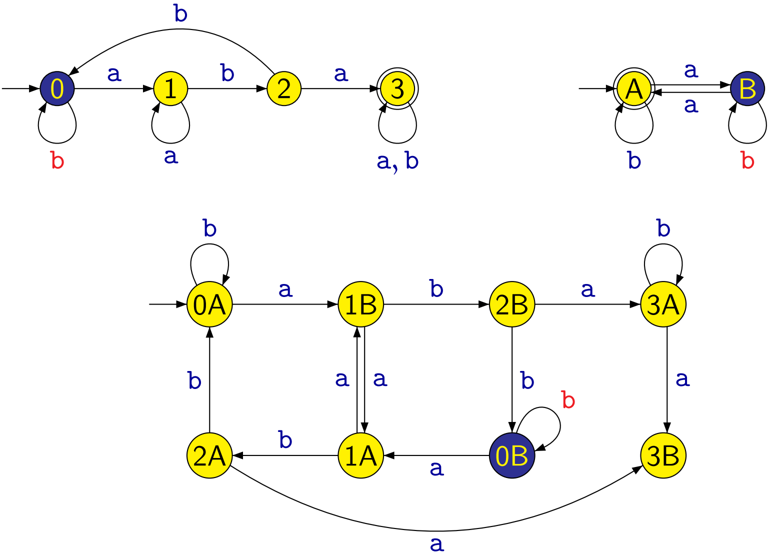
An Automaton for Intersection of Languages



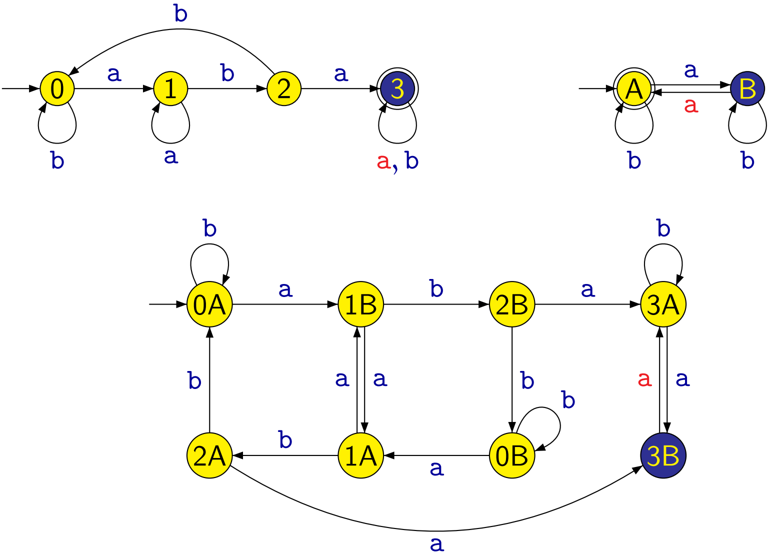
An Automaton for Intersection of Languages



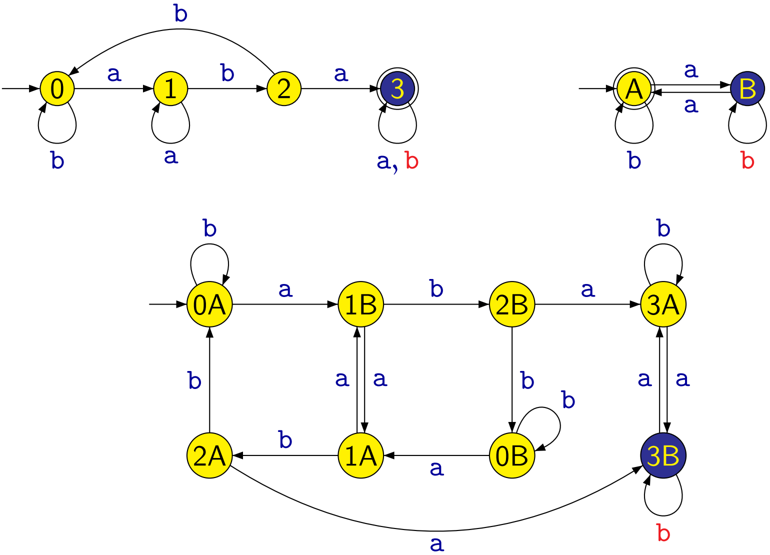
An Automaton for Intersection of Languages



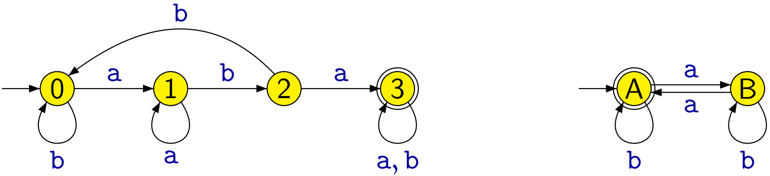
An Automaton for Intersection of Languages

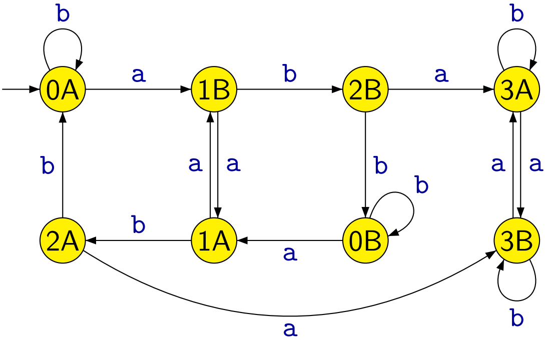


An Automaton for Intersection of Languages

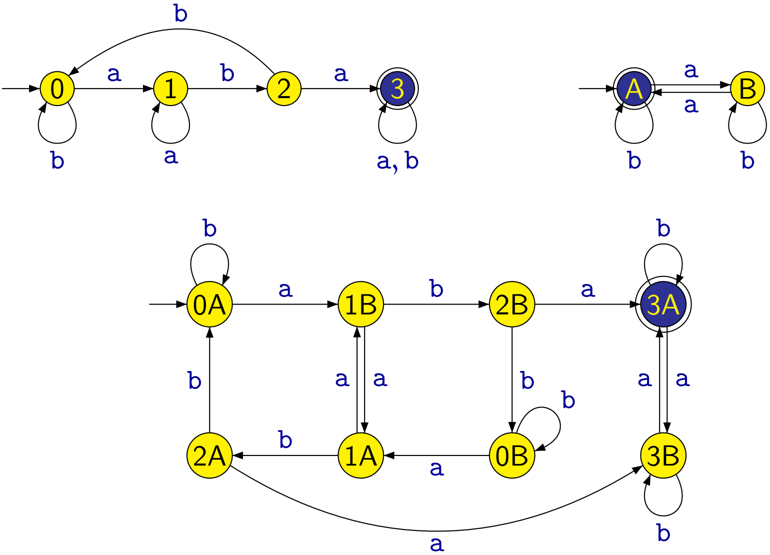


An Automaton for Intersection of Languages

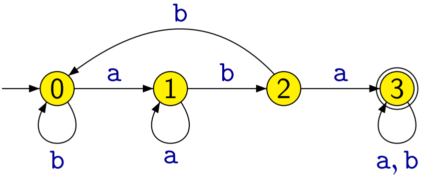


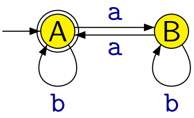


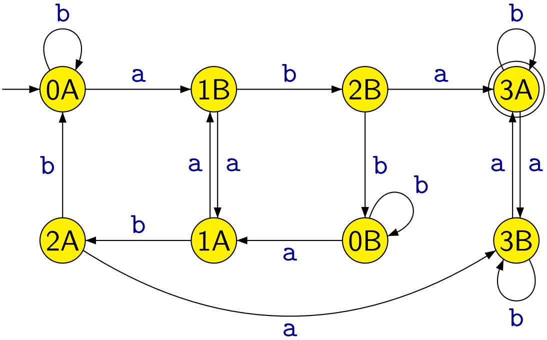
An Automaton for Intersection of Languages



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.

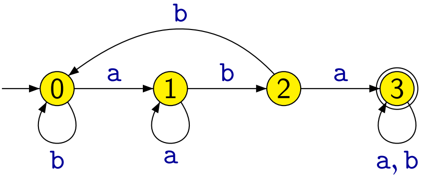
Proof: Let us assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic finite automata such that

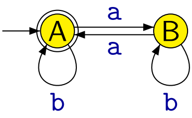
$$L_1 = \mathcal{L}(\mathcal{A}_1) \qquad\qquad L_2 = \mathcal{L}(\mathcal{A}_2)$$

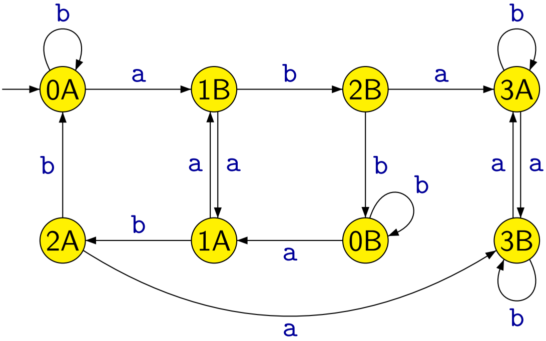
Using the described construction, we can construct a deterministic finite automaton $\mathcal{A}$ such that

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

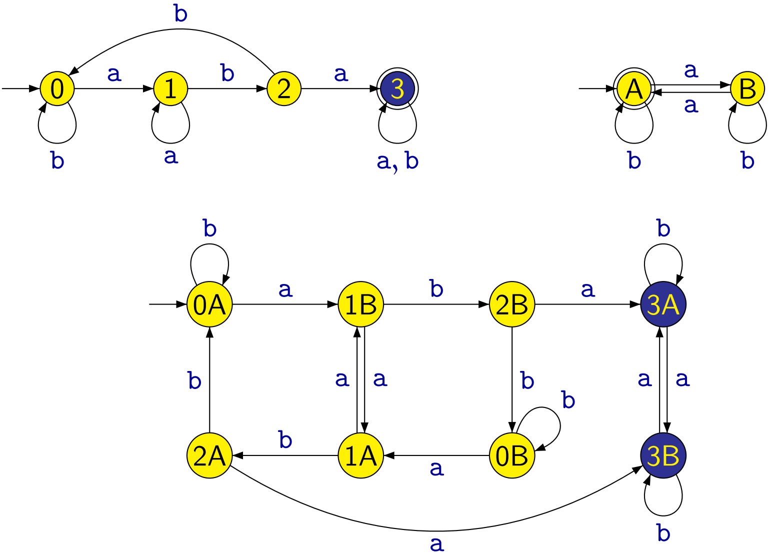
An Automaton for the Union of Languages



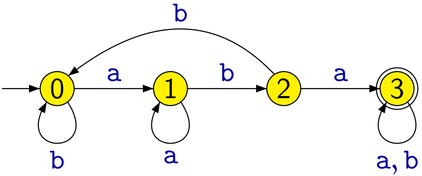


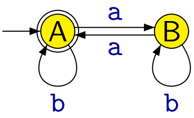


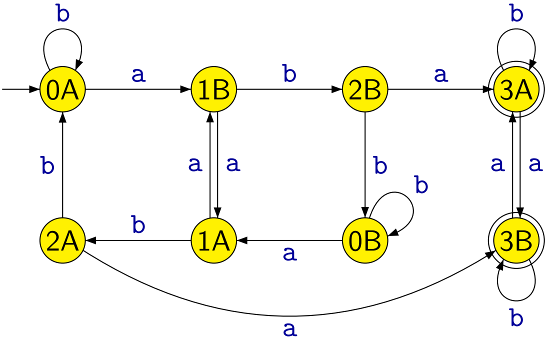
An Automaton for the Union of Languages



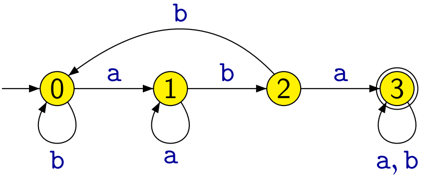
An Automaton for the Union of Languages

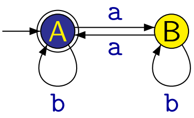


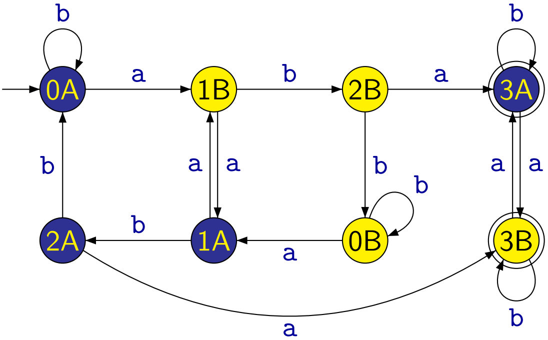




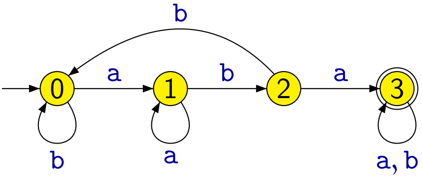
An Automaton for the Union of Languages

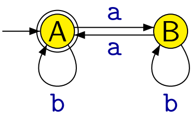


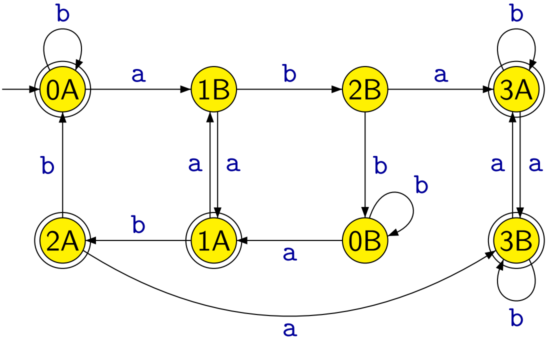




An Automaton for the Union of Languages







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

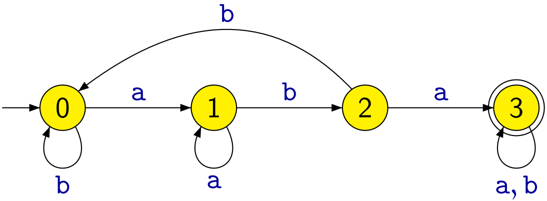
The only difference is the set of accepting states:

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

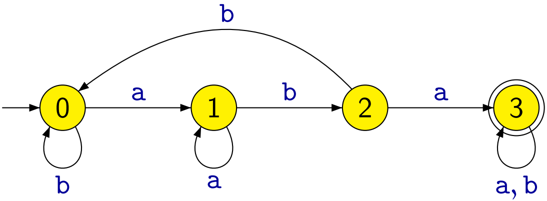
Theorem

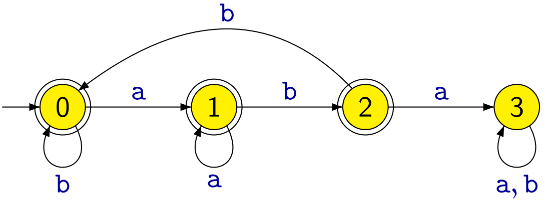
If languages $L_1, L_2 \subseteq \Sigma^*$ are regular then also the language $L_1 \cup L_2$ is regular.

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.