

Regular Expressions

Regular Expressions

Regular expressions describing languages over an alphabet Σ :

- \emptyset , ε , a (where $a \in \Sigma$) are regular expressions:
 - \emptyset ... denotes the empty language
 - ε ... denotes the language $\{\varepsilon\}$
 - a ... denotes the language $\{a\}$
- If α , β are regular expressions then also $(\alpha + \beta)$, $(\alpha \cdot \beta)$, (α^*) are regular expressions:
 - $(\alpha + \beta)$... denotes the union of languages denoted α and β
 - $(\alpha \cdot \beta)$... denotes the concatenation of languages denoted α and β
 - (α^*) ... denotes the iteration of a language denoted α
- There are no other regular expressions except those defined in the two points mentioned above.

Regular Expressions

Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.

Regular Expressions

Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.
- Since 0 and 1 are regular expression, $(0 + 1)$ is also a regular expression.

Regular Expressions

Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.
- Since 0 and 1 are regular expression, $(0 + 1)$ is also a regular expression.
- Since 0 is a regular expression, (0^*) is also a regular expression.

Regular Expressions

Example: alphabet $\Sigma = \{0, 1\}$

- According to the definition, 0 and 1 are regular expressions.
- Since 0 and 1 are regular expression, $(0 + 1)$ is also a regular expression.
- Since 0 is a regular expression, (0^*) is also a regular expression.
- Since $(0 + 1)$ and (0^*) are regular expressions, $((0 + 1) \cdot (0^*))$ is also a regular expression.

Regular Expressions

Example: alphabet $\Sigma = \{0, 1\}$

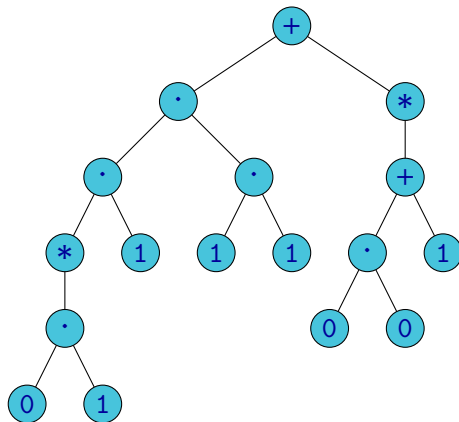
- According to the definition, 0 and 1 are regular expressions.
- Since 0 and 1 are regular expression, $(0 + 1)$ is also a regular expression.
- Since 0 is a regular expression, (0^*) is also a regular expression.
- Since $(0 + 1)$ and (0^*) are regular expressions, $((0 + 1) \cdot (0^*))$ is also a regular expression.

Remark: If α is a regular expression, by $\mathcal{L}(\alpha)$ we denote the language defined by the regular expression α .

$$\mathcal{L}(((0 + 1) \cdot (0^*))) = \{0, 1, 00, 10, 000, 100, 0000, 1000, 00000, \dots\}$$

Regular Expressions

The structure of a regular expression can be represented by an abstract syntax tree:



$$((((((0 \cdot 1)^*) \cdot 1) \cdot (1 \cdot 1)) + (((0 \cdot 0) + 1)^*))$$

The formal definition of semantics of regular expressions:

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$
- $\mathcal{L}(a) = \{a\}$
- $\mathcal{L}(\alpha^*) = \mathcal{L}(\alpha)^*$
- $\mathcal{L}(\alpha \cdot \beta) = \mathcal{L}(\alpha) \cdot \mathcal{L}(\beta)$
- $\mathcal{L}(\alpha + \beta) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$

Regular Expressions

To make regular expressions more lucid and succinct, we use the following conventions:

- The outward pair of parentheses can be omitted.
- We can omit parentheses that are superfluous due to associativity of operations of union (+) and concatenation (·).
- We can omit parentheses that are superfluous due to the defined priority of operators (iteration (*) has the highest priority, concatenation (·) has lower priority, and union (+) has the lowest priority).
- A dot denoting concatenation can be omitted.

Example: Instead of

$$((((0 \cdot 1)^*) \cdot 1) \cdot (1 \cdot 1)) + (((0 \cdot 0) + 1)^*)$$

we usually write

$$(01)^* 111 + (00 + 1)^*$$

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word a

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word **a**

ab ... the language containing the only word **ab**

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word a

ab ... the language containing the only word ab

$a + b$... the language containing two words a and b

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word a

ab ... the language containing the only word ab

$a + b$... the language containing two words a and b

a^* ... the language containing words ε , a , aa , aaa , ...

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word a

ab ... the language containing the only word ab

$a + b$... the language containing two words a and b

a^* ... the language containing words ε , a , aa , aaa , ...

$(ab)^*$... the language containing words ε , ab , $abab$, $ababab$, ...

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word **a**

ab ... the language containing the only word **ab**

a + b ... the language containing two words **a** and **b**

a^{*} ... the language containing words ε , **a**, **aa**, **aaa**, ...

(ab)^{*} ... the language containing words ε , **ab**, **abab**, **ababab**, ...

(a + b)^{*} ... the language containing all words over the alphabet $\{a, b\}$

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word a

ab ... the language containing the only word ab

$a + b$... the language containing two words a and b

a^* ... the language containing words ε , a , aa , aaa , ...

$(ab)^*$... the language containing words ε , ab , $abab$, $ababab$, ...

$(a + b)^*$... the language containing all words over the alphabet $\{a, b\}$

$(a + b)^*aa$... the language containing all words ending with aa

Regular Expressions

Examples: In all examples $\Sigma = \{a, b\}$.

a ... the language containing the only word a

ab ... the language containing the only word ab

$a + b$... the language containing two words a and b

a^* ... the language containing words ε , a , aa , aaa , ...

$(ab)^*$... the language containing words ε , ab , $abab$, $ababab$, ...

$(a + b)^*$... the language containing all words over the alphabet $\{a, b\}$

$(a + b)^*aa$... the language containing all words ending with aa

$(ab)^*bbb(ab)^*$... the language containing all words that contain a subword bbb preceded and followed by an arbitrary number of copies of the word ab

$(a + b)^*aa + (ab)^*bbb(ab)^*$... the language containing all words that either end with aa or contain a subwords bbb preceded and followed with some arbitrary number of words ab

$(a + b)^*aa + (ab)^*bbb(ab)^*$... the language containing all words that either end with aa or contain a subwords bbb preceded and followed with some arbitrary number of words ab

$(a + b)^*b(a + b)^*$... the language of all words that contain at least one occurrence of symbol b

Regular Expressions

$(a + b)^*aa + (ab)^*bbb(ab)^*$... the language containing all words that either end with aa or contain a subwords bbb preceded and followed with some arbitrary number of words ab

$(a + b)^*b(a + b)^*$... the language of all words that contain at least one occurrence of symbol b

$a^*(ba^*ba^*)^*$... the language containing all words with an even number of occurrences of symbol b

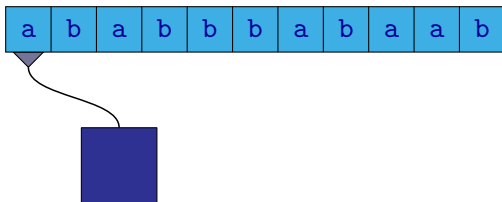
Finite Automata

Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

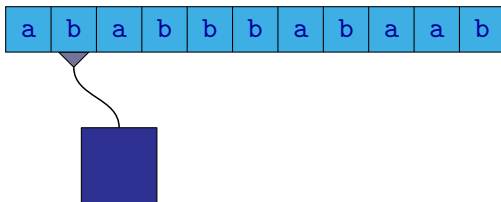


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

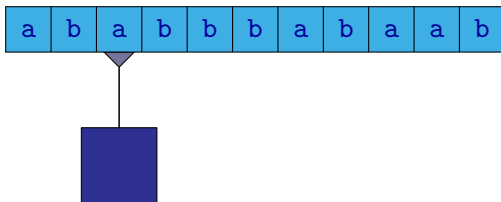


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

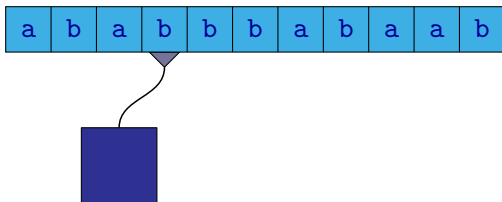


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

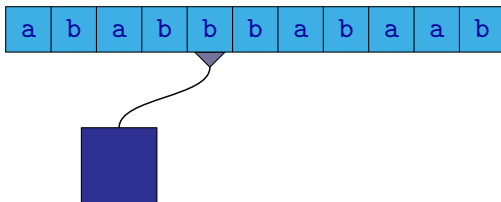


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

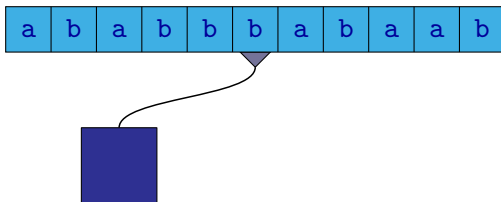


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

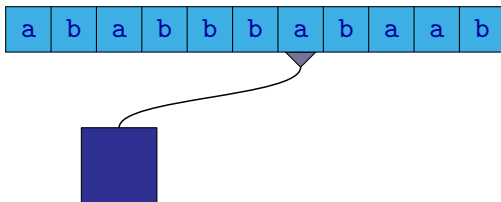


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

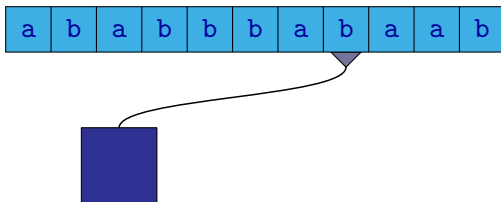


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

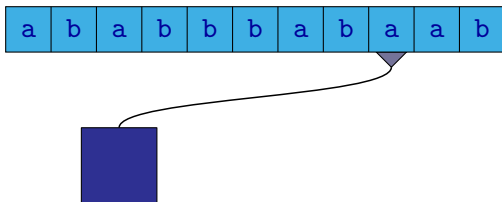


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

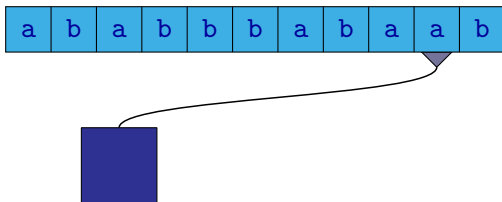


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

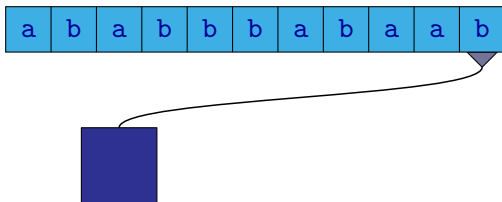


Recognition of a Language

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

We would like to recognize a language L consisting of words with even number of symbols 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.

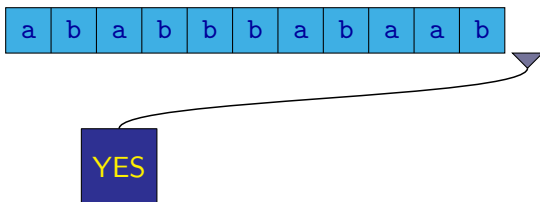


Recognition of a Language

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

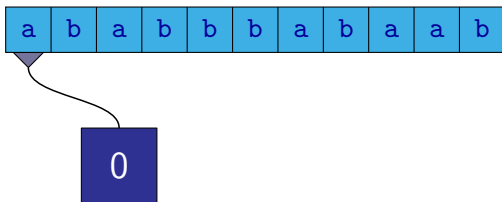
We would like to recognize a language L consisting of words with even number of symbols b .

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



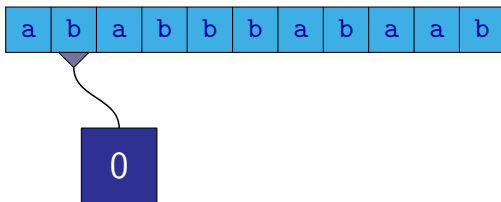
Recognition of a Language

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



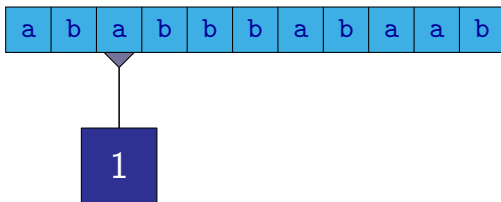
Recognition of a Language

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



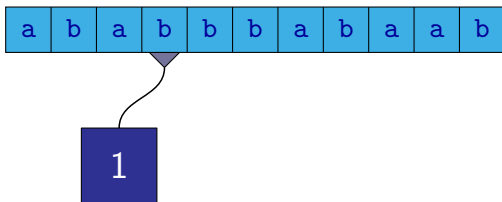
Recognition of a Language

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



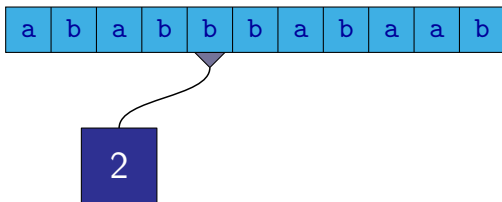
Recognition of a Language

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



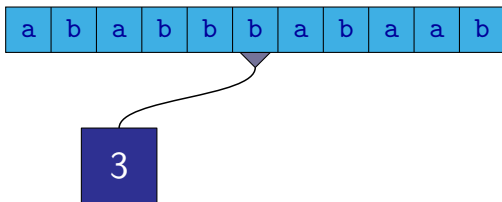
Recognition of a Language

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



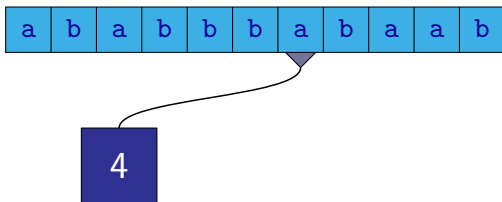
Recognition of a Language

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



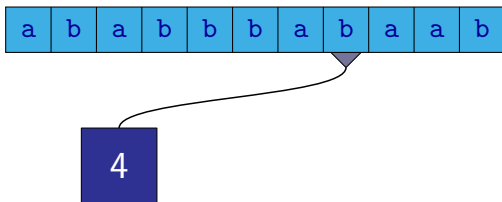
Recognition of a Language

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



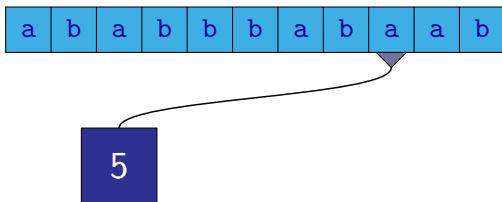
Recognition of a Language

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



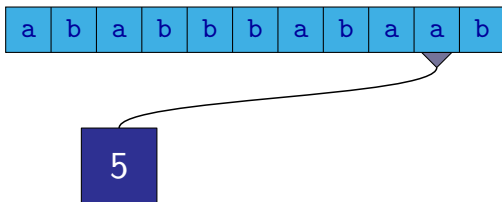
Recognition of a Language

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



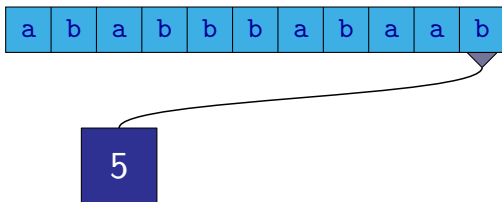
Recognition of a Language

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



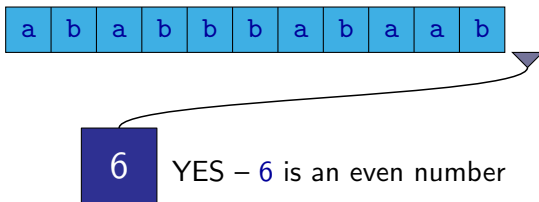
Recognition of a Language

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



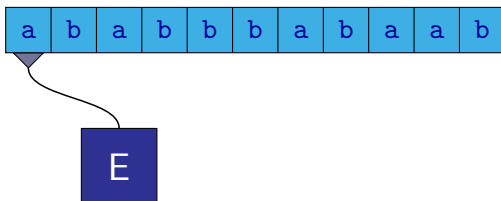
Recognition of a Language

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



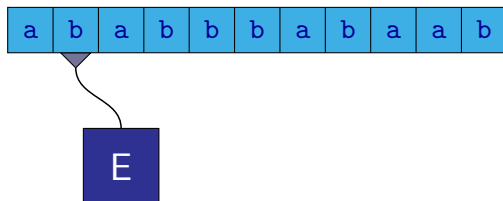
Recognition of a Language

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).



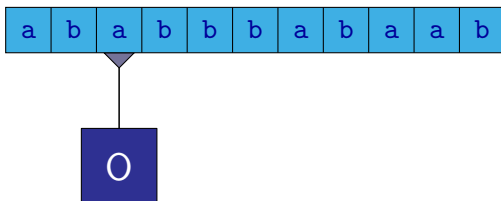
Recognition of a Language

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).



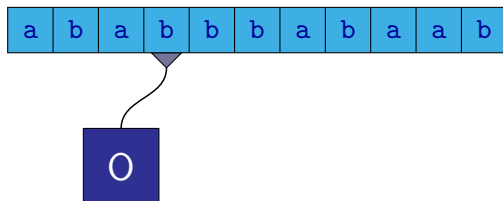
Recognition of a Language

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).



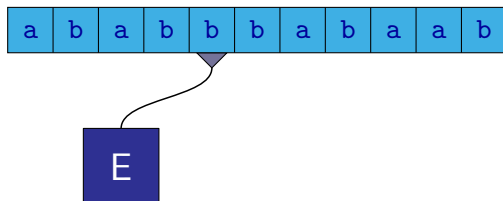
Recognition of a Language

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).



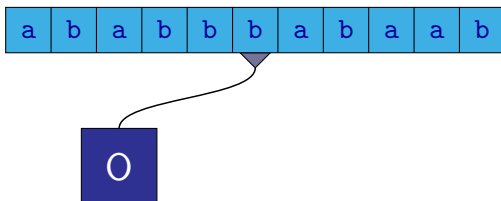
Recognition of a Language

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).



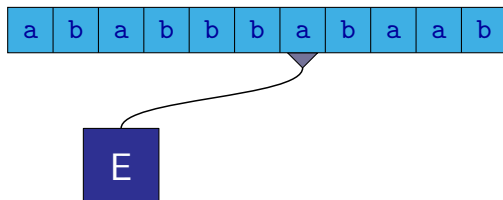
Recognition of a Language

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).



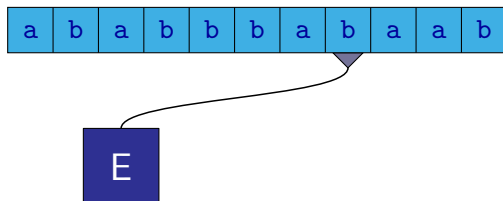
Recognition of a Language

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).



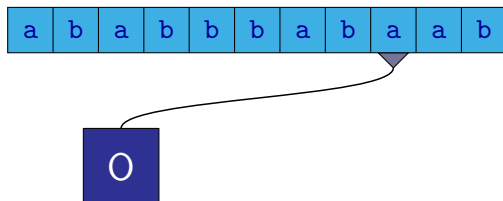
Recognition of a Language

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).



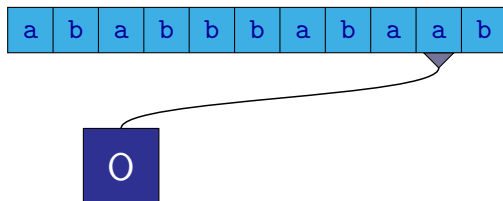
Recognition of a Language

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).



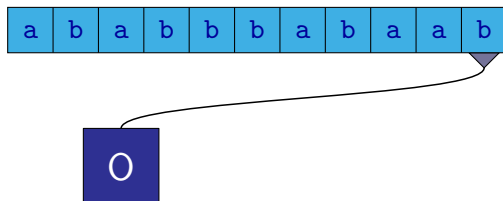
Recognition of a Language

The 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).



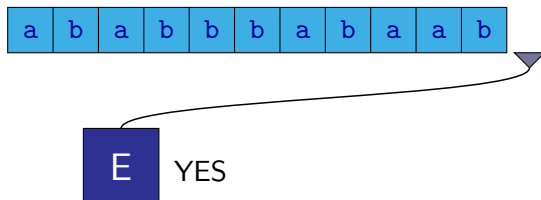
Recognition of a Language

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:



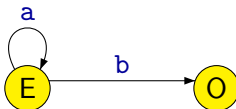
Recognition of a Language

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



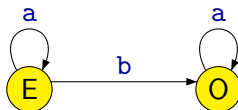
Recognition of a Language

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



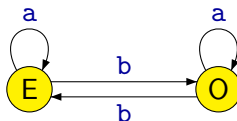
Recognition of a Language

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



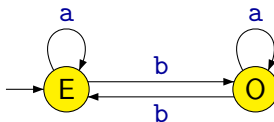
Recognition of a Language

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



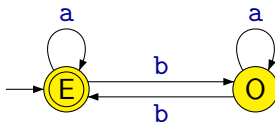
Recognition of a Language

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



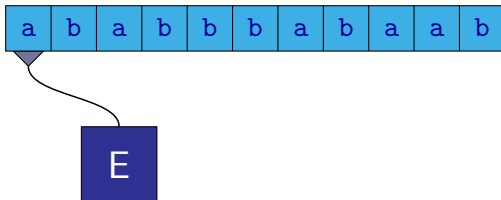
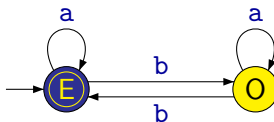
Recognition of a Language

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



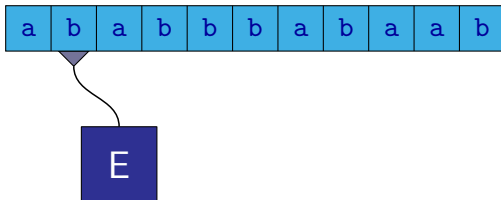
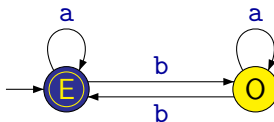
Recognition of a Language

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



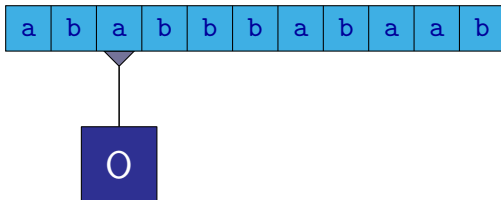
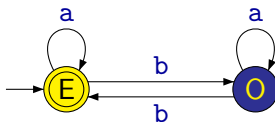
Recognition of a Language

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



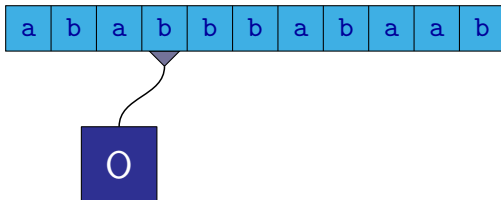
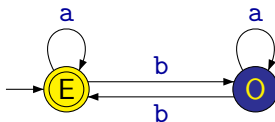
Recognition of a Language

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



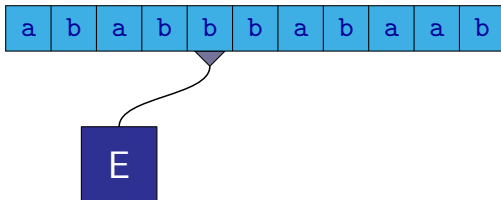
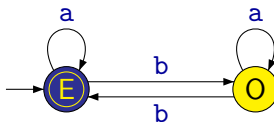
Recognition of a Language

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



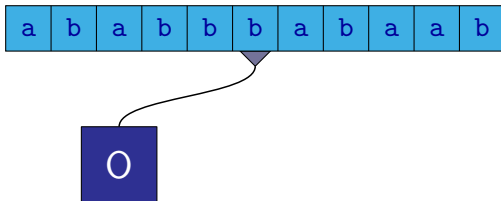
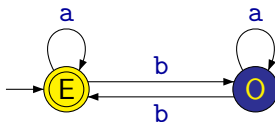
Recognition of a Language

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



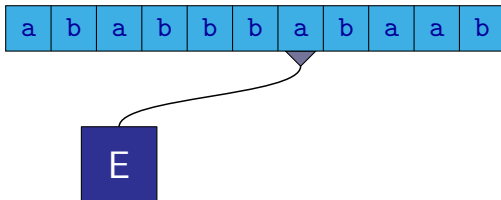
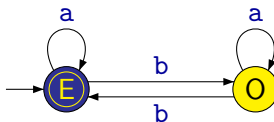
Recognition of a Language

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



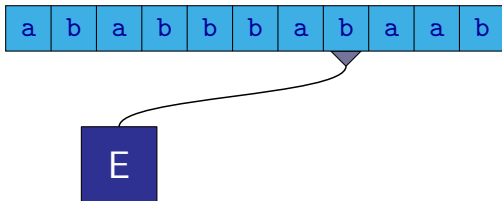
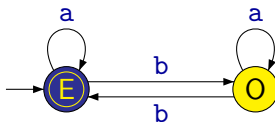
Recognition of a Language

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



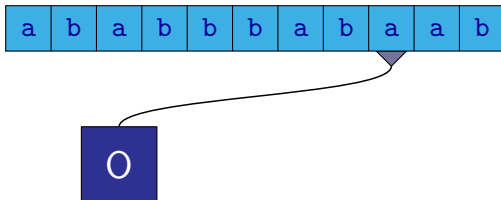
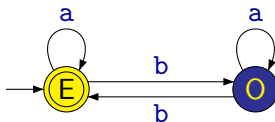
Recognition of a Language

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



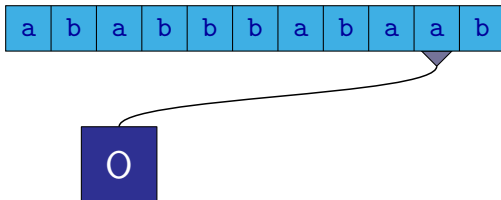
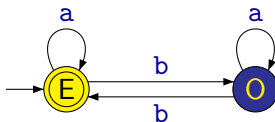
Recognition of a Language

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



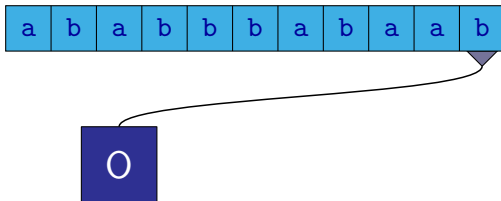
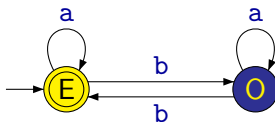
Recognition of a Language

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



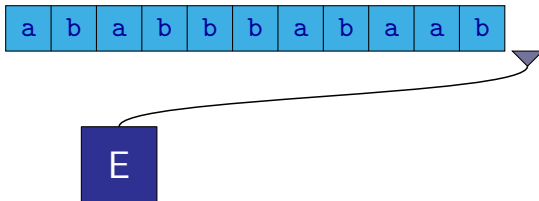
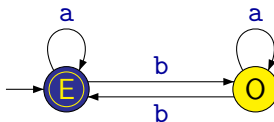
Recognition of a Language

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

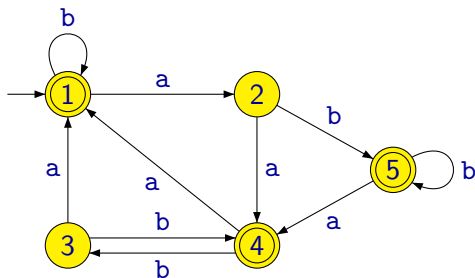


Recognition of a Language

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



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

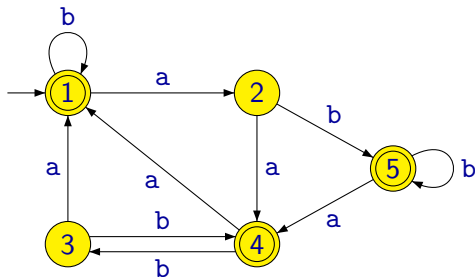
Formally, a **deterministic finite automaton (DFA)** is defined as a tuple

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

where:

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

Deterministic Finite Automaton



- $Q = \{1, 2, 3, 4, 5\}$

- $\Sigma = \{a, b\}$

- $q_0 = 1$

- $F = \{1, 4, 5\}$

$$\delta(1, a) = 2 \quad \delta(1, b) = 1$$

$$\delta(2, a) = 4 \quad \delta(2, b) = 5$$

$$\delta(3, a) = 1 \quad \delta(3, b) = 4$$

$$\delta(4, a) = 1 \quad \delta(4, b) = 3$$

$$\delta(5, a) = 4 \quad \delta(5, b) = 5$$

Deterministic Finite Automaton

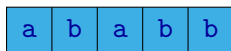
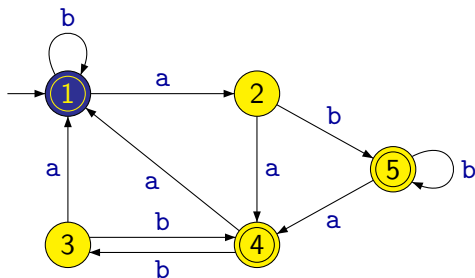
Instead of

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

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

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

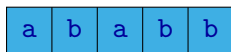
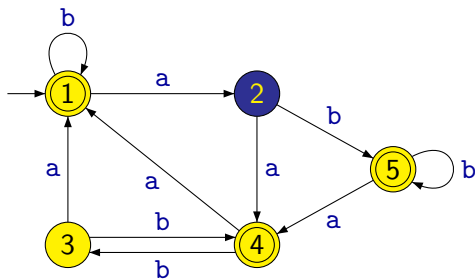
Deterministic Finite Automaton



1

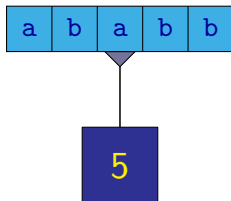
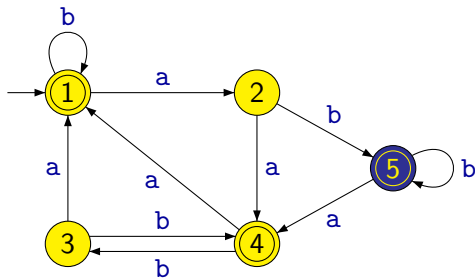


Deterministic Finite Automaton



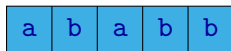
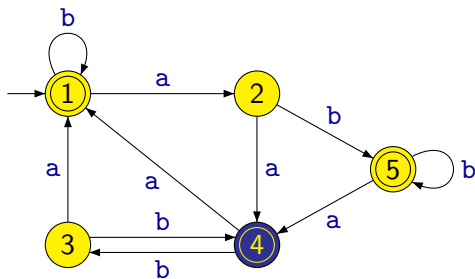
$1 \xrightarrow{a} 2$

Deterministic Finite Automaton



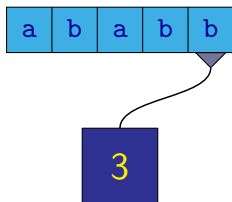
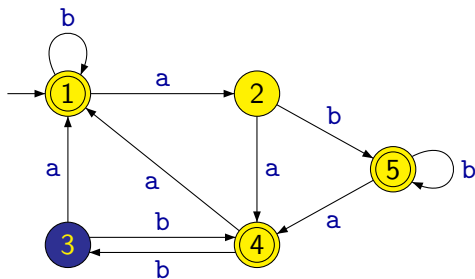
$1 \xrightarrow{a} 2 \xrightarrow{b} 5$

Deterministic Finite Automaton



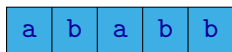
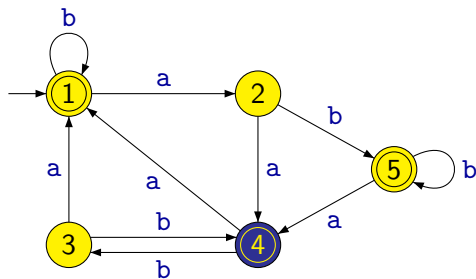
$1 \xrightarrow{a} 2 \xrightarrow{b} 5 \xrightarrow{a} 4$

Deterministic Finite Automaton



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

Deterministic Finite Automaton



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

Deterministic Finite Automaton

Definition

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

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

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

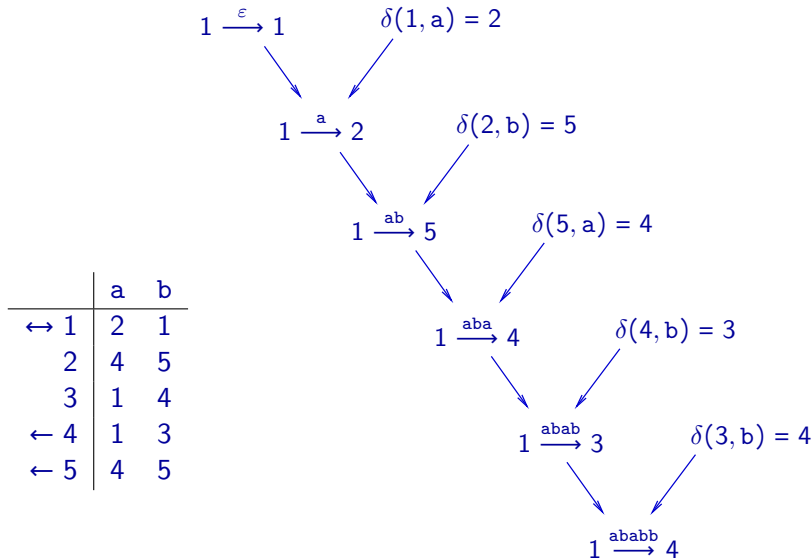
Instead of $(q, w, q') \in \longrightarrow$ we write $q \xrightarrow{w} q'$.

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

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

Deterministic Finite Automaton



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.,

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

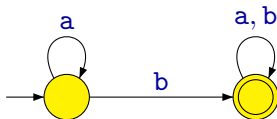
Definition

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 $\{a, b\}$ consisting of those words that contain at least one occurrence of symbol b , i.e.,

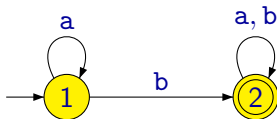
$$L = \{w \in \{a, b\}^* \mid |w|_b \geq 1\}$$



Examples of Deterministic Finite Automata

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

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

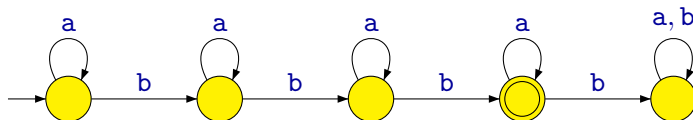


	a	b
→ 1	1	2
← 2	2	2

Examples of Deterministic Finite Automata

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

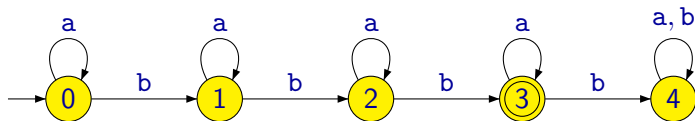
$$L = \{w \in \{a, b\}^* \mid |w|_b = 3\}$$



Examples of Deterministic Finite Automata

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

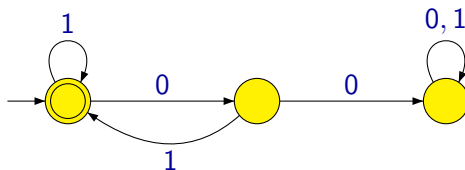
$$L = \{w \in \{a, b\}^* \mid |w|_b = 3\}$$



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

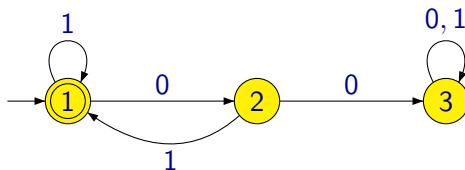
Examples of Deterministic Finite Automata

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

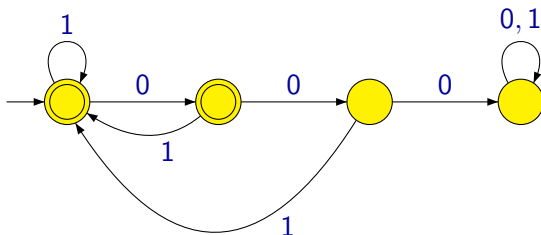
Example: An automaton recognizing the language over alphabet $\{0, 1\}$ consisting of those words where every occurrence of symbol 0 is immediately followed with symbol 1.



	0	1
↔ 1	2	1
2	3	1
3	3	3

Examples of Deterministic Finite Automata

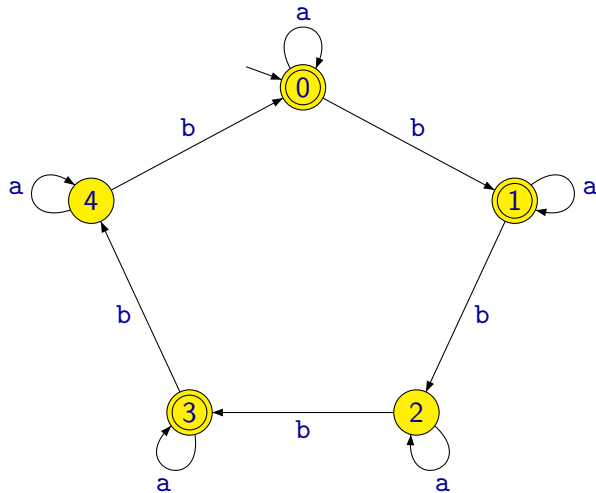
Example: An automaton recognizing the language over alphabet $\{0, 1\}$ consisting of those words where every pair of consecutive symbols 0 is immediately followed with symbol 1 .



Examples of Deterministic Finite Automata

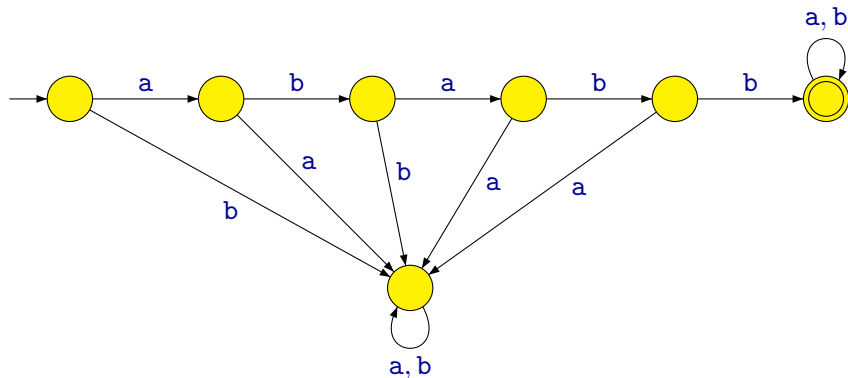
Example: An automaton recognizing the language

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



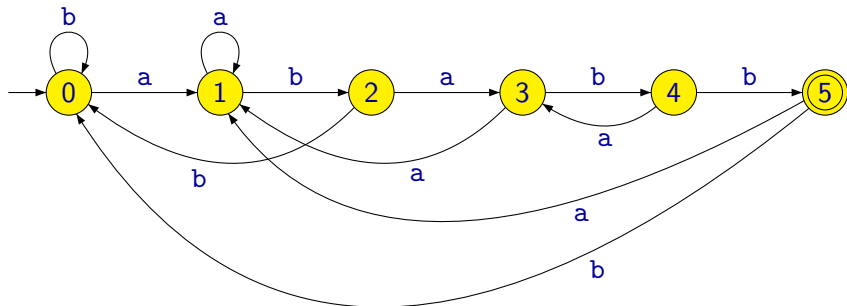
Examples of Deterministic Finite Automata

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



Examples of Deterministic Finite Automata

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:

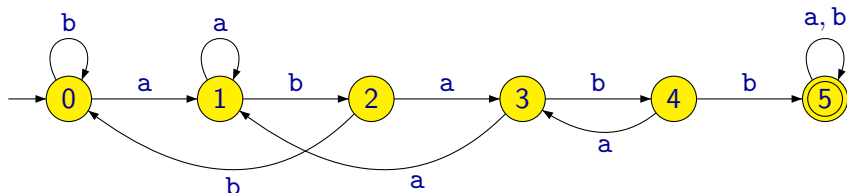
- Let us assume that we want to search for a word u of length n (i.e., $|u| = n$).
The states of the automaton are denoted with numbers $0, 1, \dots, n$.
- A state with number i corresponds to the situation when i is the length of the longest word that is at the same time:
 - a prefix of the pattern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

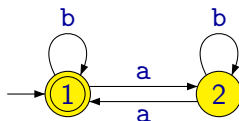
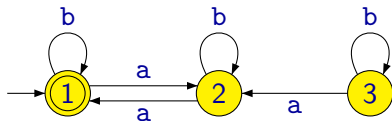
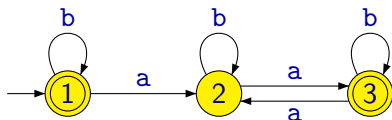
- | | | | | | |
|-----------|-----|---------------|-----------|-----|-------|
| • State 0 | ... | ε | • State 3 | ... | aba |
| • State 1 | ... | a | • State 4 | ... | abab |
| • State 2 | ... | ab | • State 5 | ... | ababb |

Examples of Deterministic Finite Automata

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



Equivalence of Automata



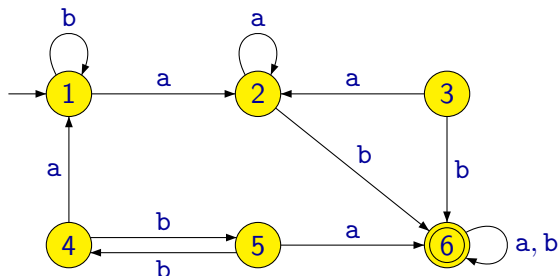
All three automata accept the language of all words with an even number of **a**'s.

Equivalence of Automata

Definition

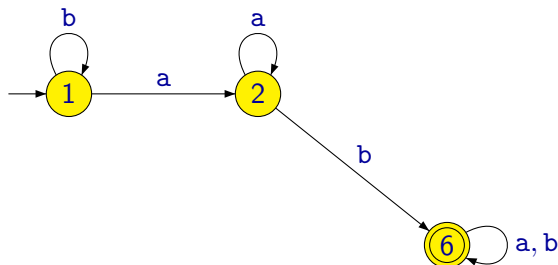
We say automata $\mathcal{A}_1, \mathcal{A}_2$ are **equivalent** if $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

Unreachable States of an Automaton



- The automaton accepts the language
 $L = \{w \in \{a, b\}^* \mid w \text{ contains subword } ab\}$
- 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 $L = \{w \in \{a, b\}^* \mid w \text{ contains subword } ab\}$
- 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 **reachable** 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.

Automata and Operations on Languages

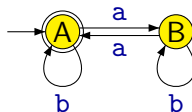
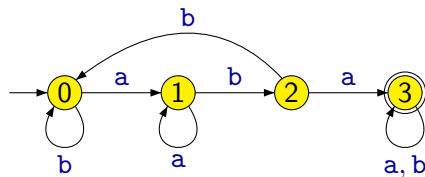
When we construct automata, it can be difficult to construct an automaton for a given language L directly.

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

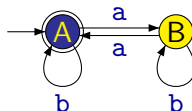
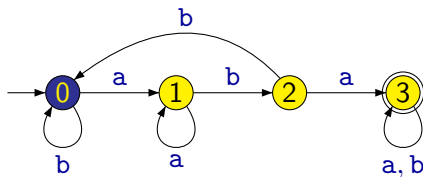
Let us have the following two automata:



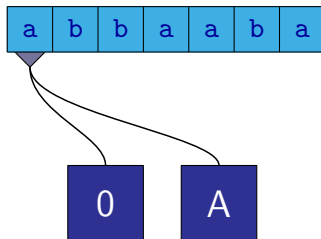
Do both of them accept the word **abbaaba**?

An Automaton for Intersection of Languages

Let us have the following two automata:

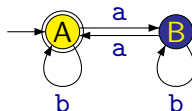
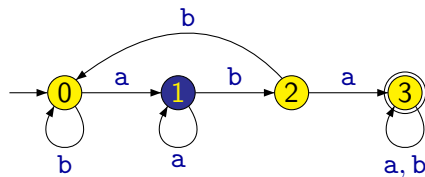


Do both of them accept the word **abbaaba**?

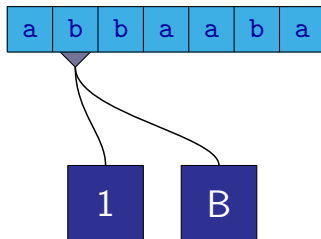


An Automaton for Intersection of Languages

Let us have the following two automata:

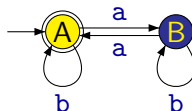
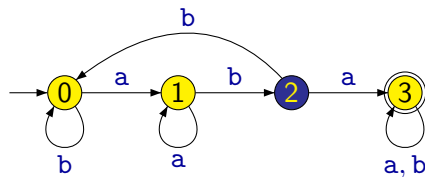


Do both of them accept the word **abbaaba**?

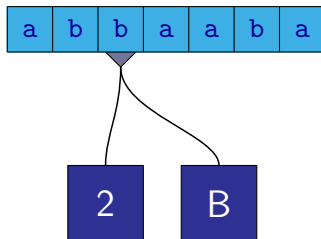


An Automaton for Intersection of Languages

Let us have the following two automata:

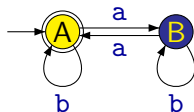
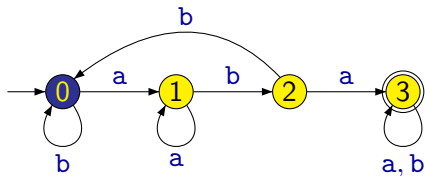


Do both of them accept the word **abbaaba**?

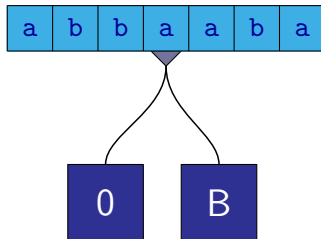


An Automaton for Intersection of Languages

Let us have the following two automata:

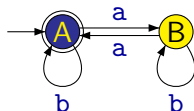
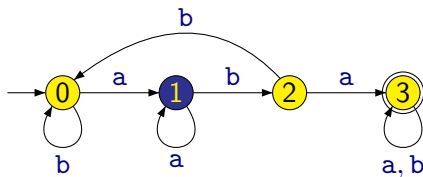


Do both of them accept the word **abbaaba**?

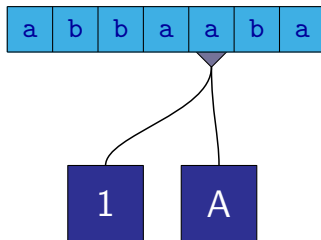


An Automaton for Intersection of Languages

Let us have the following two automata:

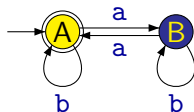
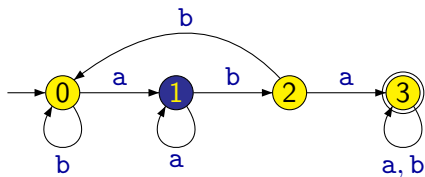


Do both of them accept the word **abbaaba**?

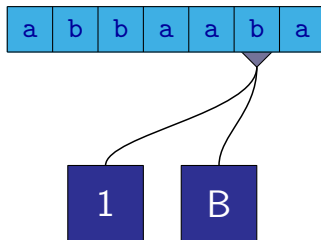


An Automaton for Intersection of Languages

Let us have the following two automata:

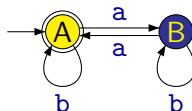
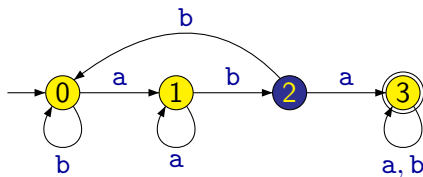


Do both of them accept the word **abbaaba**?

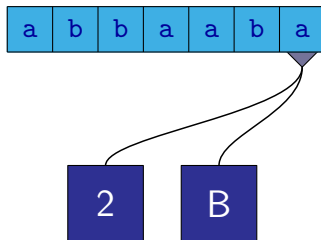


An Automaton for Intersection of Languages

Let us have the following two automata:

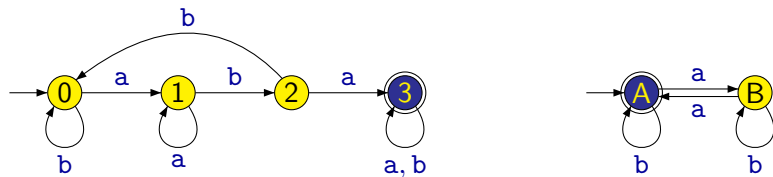


Do both of them accept the word **abbaaba**?

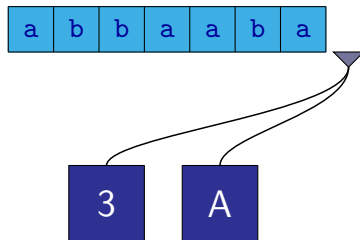


An Automaton for Intersection of Languages

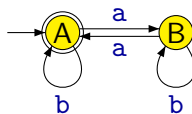
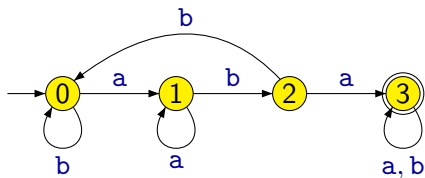
Let us have the following two automata:



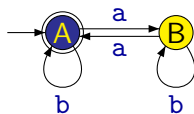
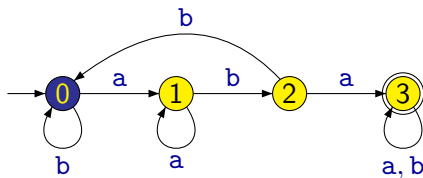
Do both of them accept the word **abbaaba**?



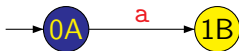
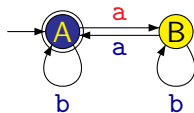
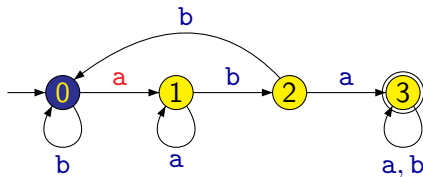
An Automaton for Intersection of Languages



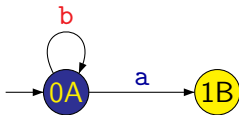
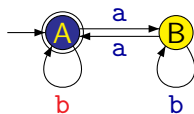
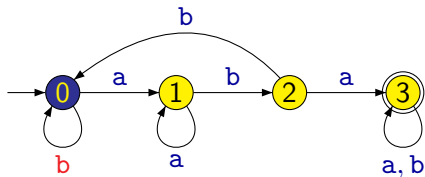
An Automaton for Intersection of Languages



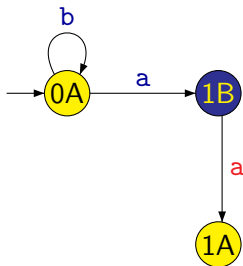
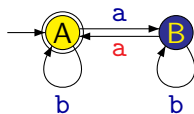
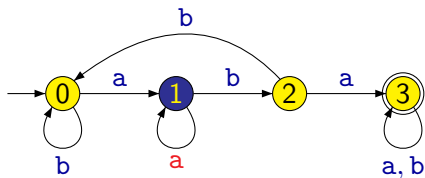
An Automaton for Intersection of Languages



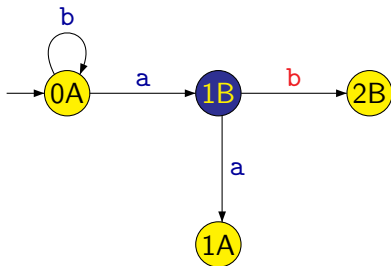
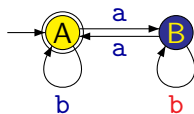
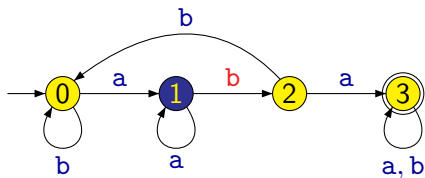
An Automaton for Intersection of Languages



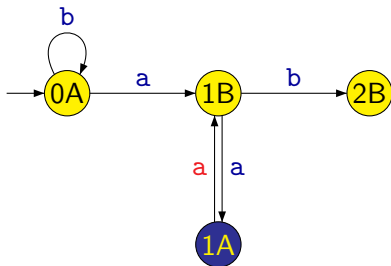
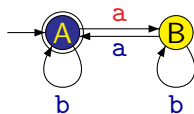
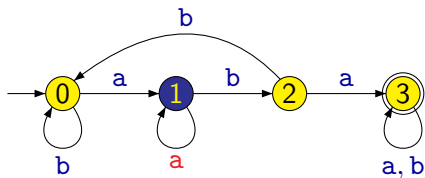
An Automaton for Intersection of Languages



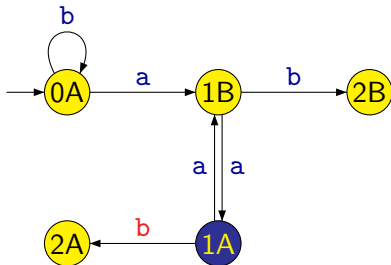
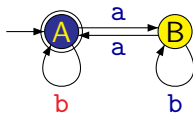
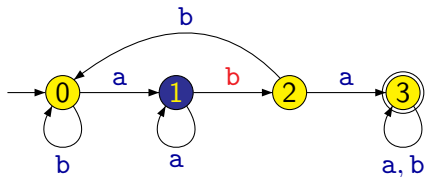
An Automaton for Intersection of Languages



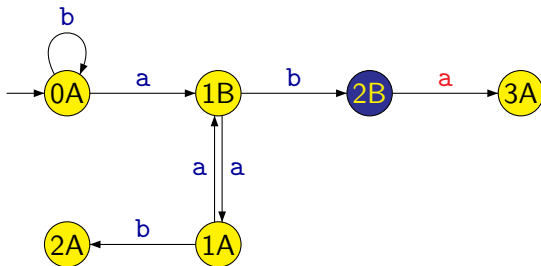
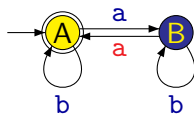
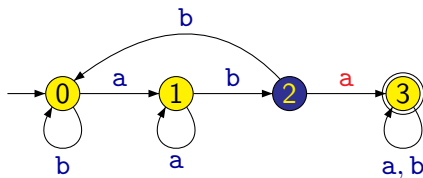
An Automaton for Intersection of Languages



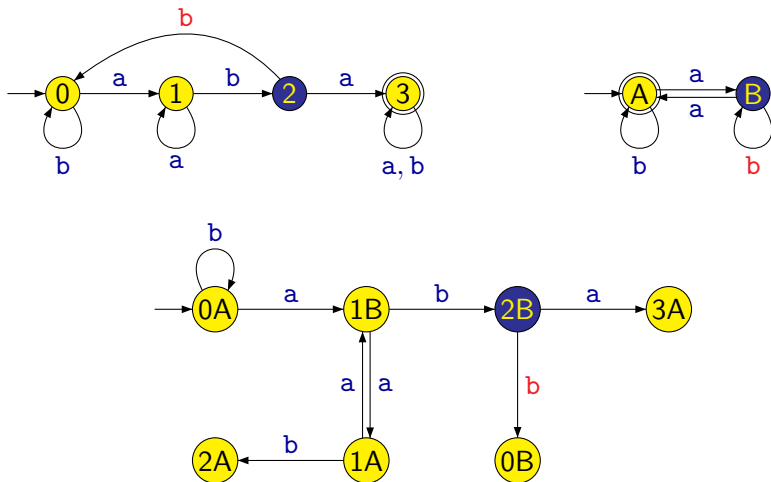
An Automaton for Intersection of Languages



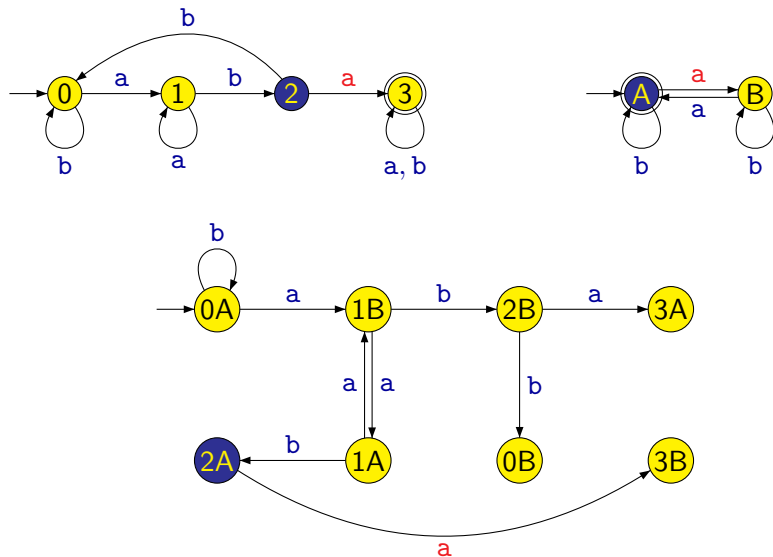
An Automaton for Intersection of Languages



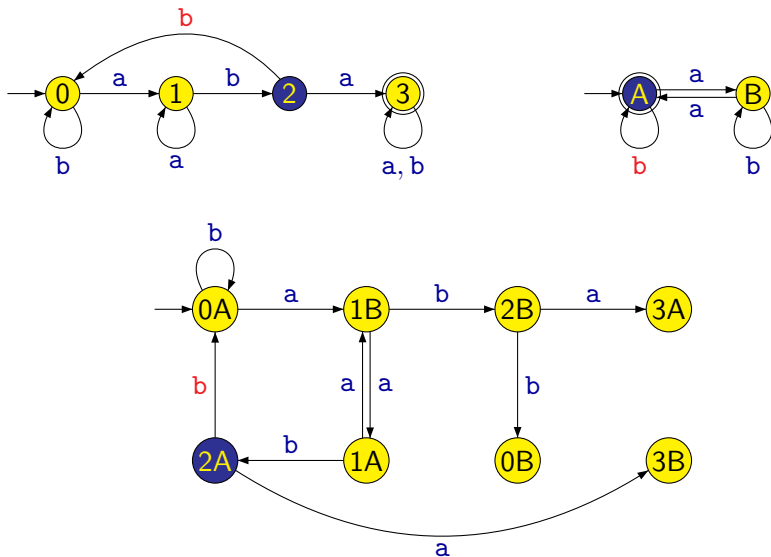
An Automaton for Intersection of Languages



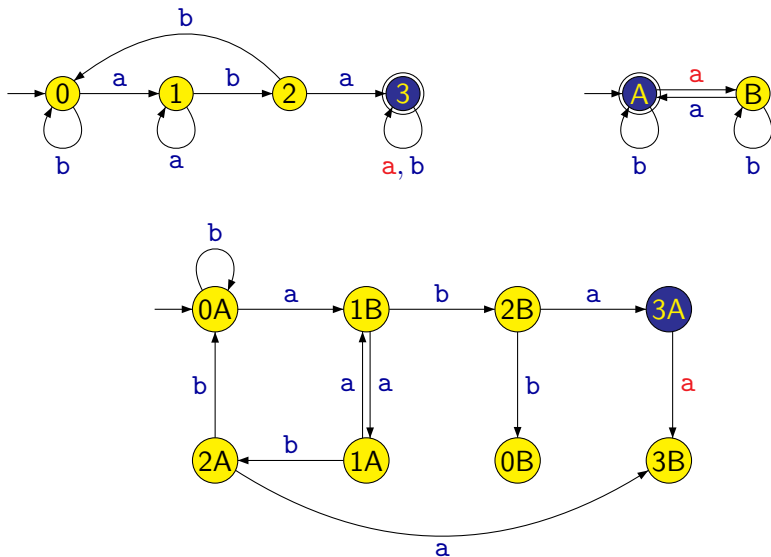
An Automaton for Intersection of Languages



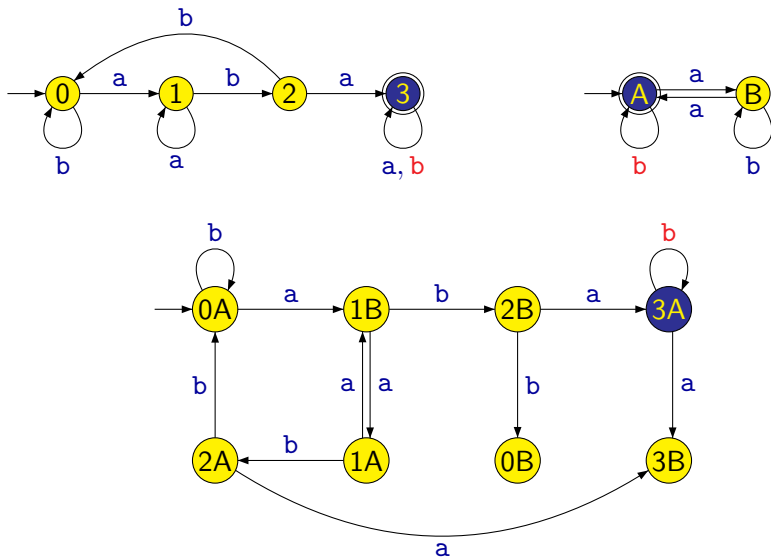
An Automaton for Intersection of Languages



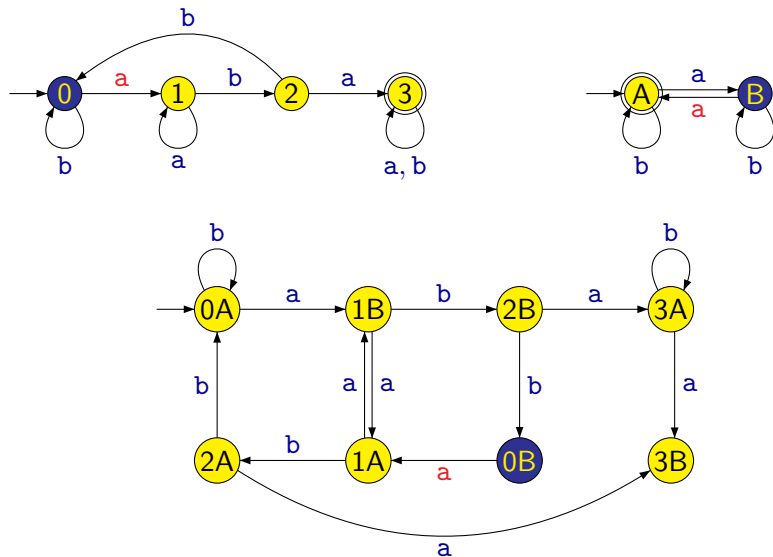
An Automaton for Intersection of Languages



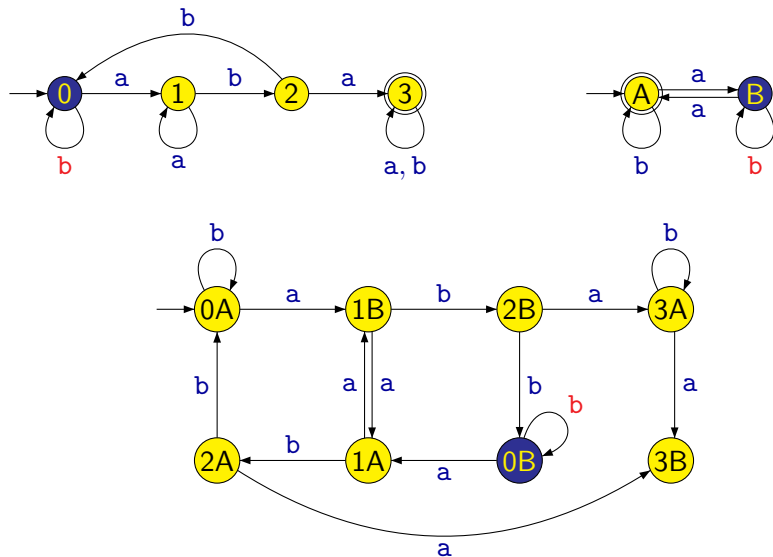
An Automaton for Intersection of Languages



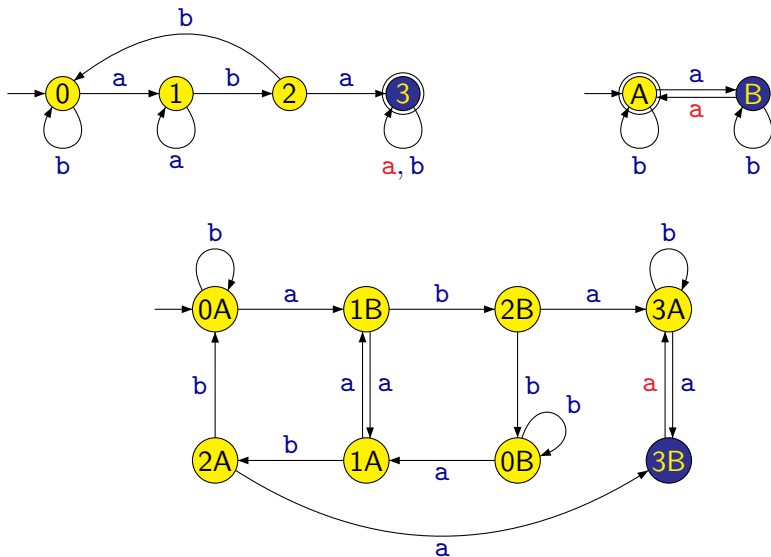
An Automaton for Intersection of Languages



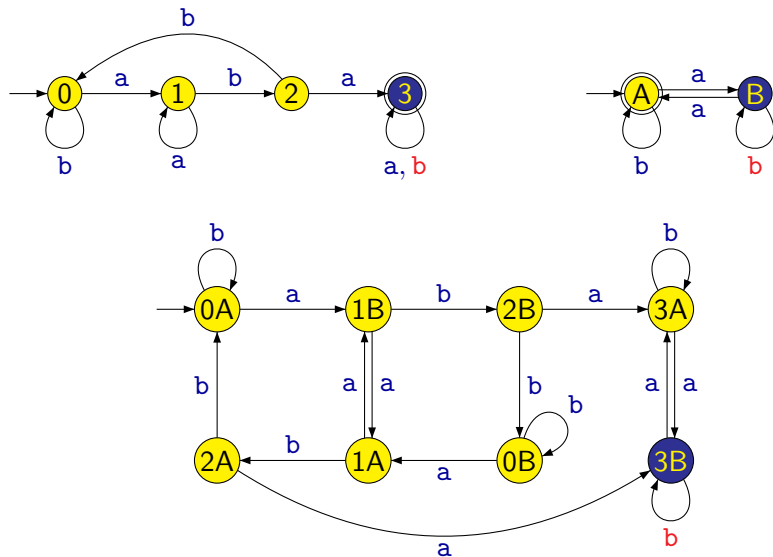
An Automaton for Intersection of Languages



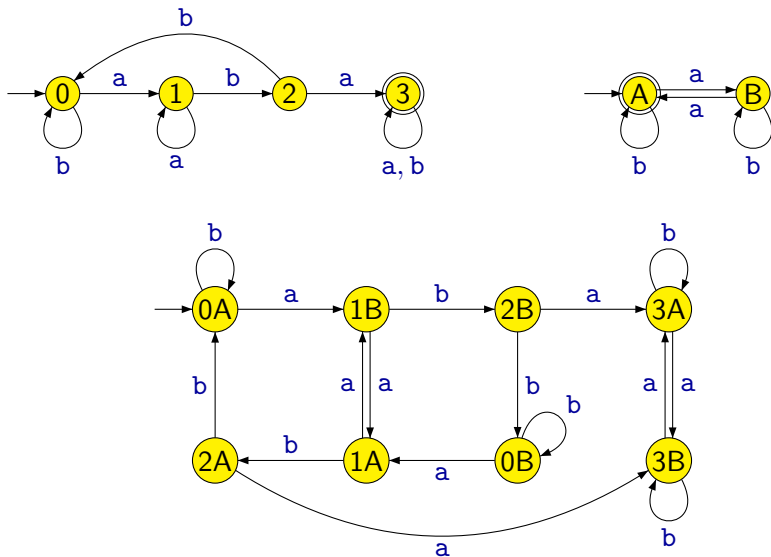
An Automaton for Intersection of Languages



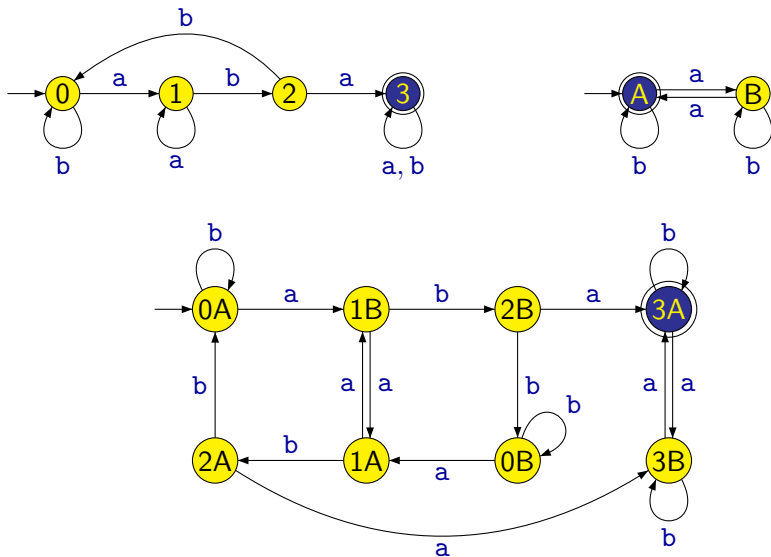
An Automaton for Intersection of Languages



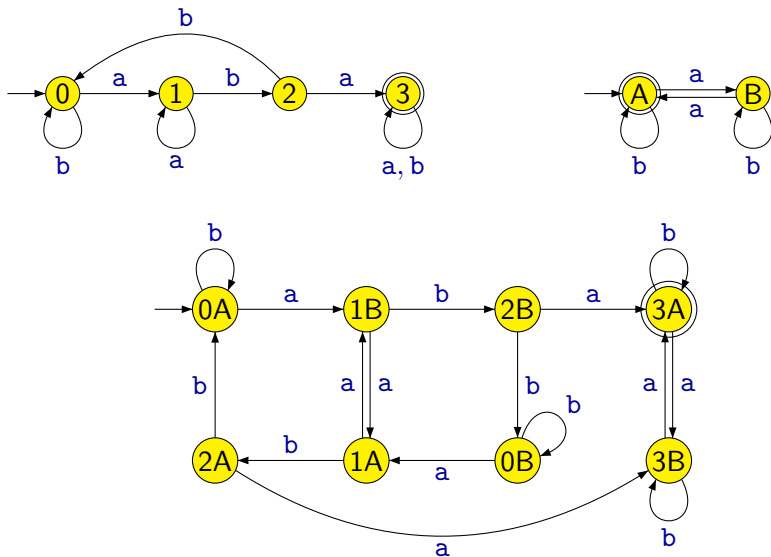
An Automaton for Intersection of Languages



An Automaton for Intersection of Languages



An Automaton for Intersection of Languages



An Automaton for Intersection of Languages

Formally, the construction can be described as follows:

We assume we have two deterministic finite automata

$\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$.

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

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

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

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

Intersection of Regular Languages

Theorem

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

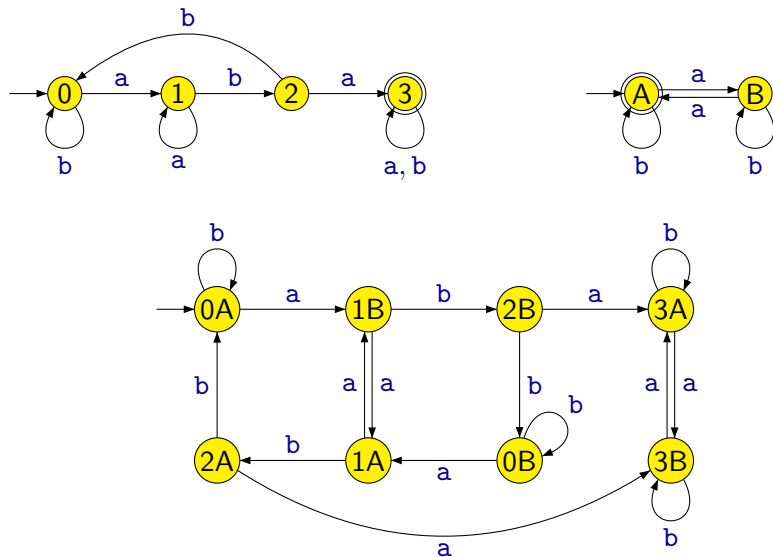
Proof: Let us assume that \mathcal{A}_1 and \mathcal{A}_2 are deterministic finite automata such that

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

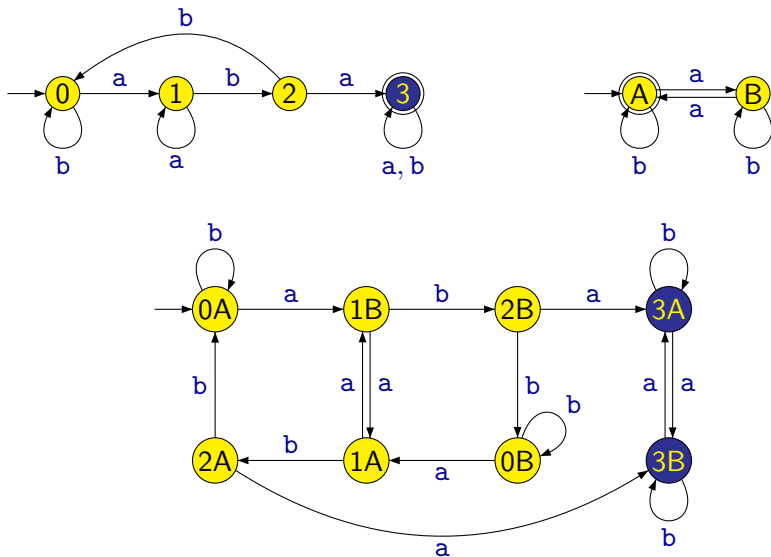
Using the described construction, we can construct a deterministic finite automaton \mathcal{A} such that

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

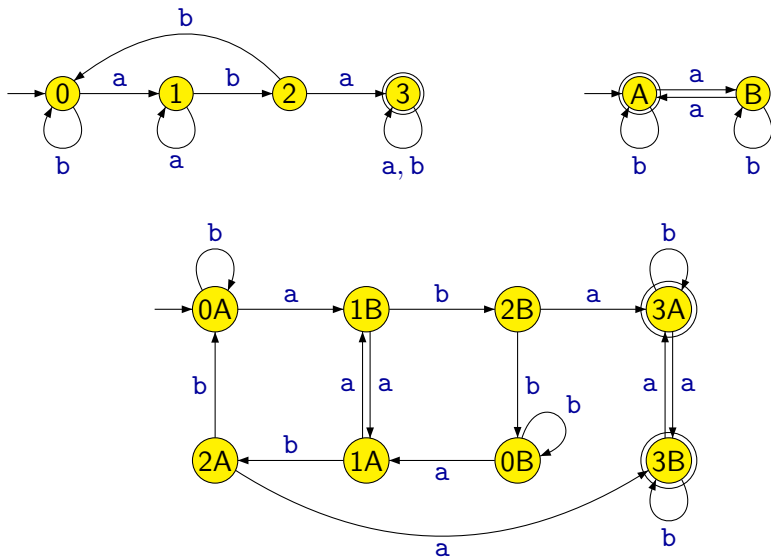
An Automaton for the Union of Languages



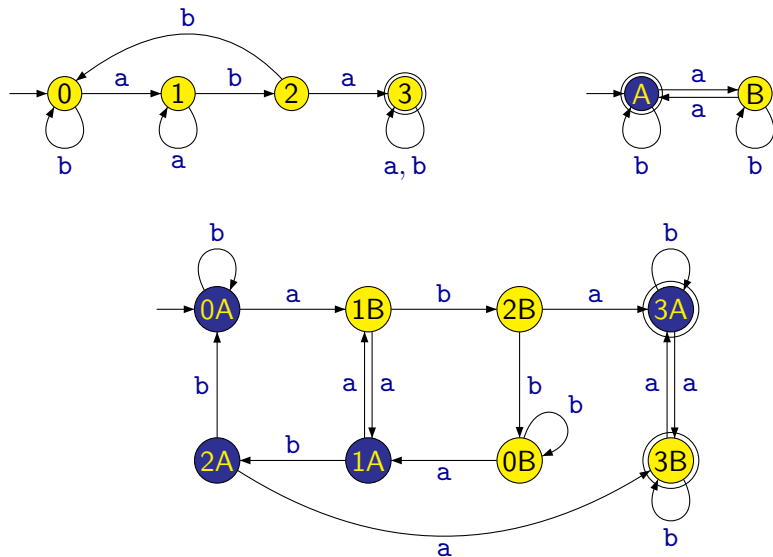
An Automaton for the Union of Languages



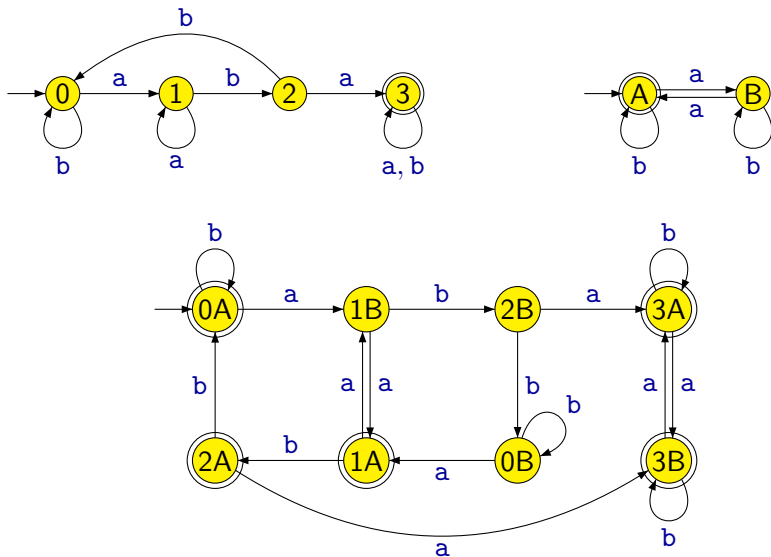
An Automaton for the Union of Languages



An Automaton for the Union of Languages



An Automaton for the Union of Languages



Union of Regular Languages

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

$$\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$$

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

$$\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$$

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

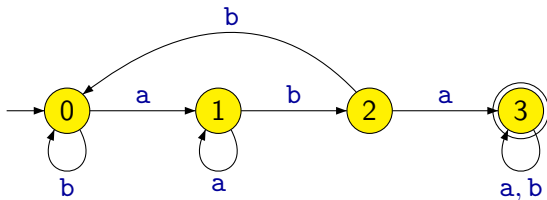
The only difference is the set of accepting states:

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

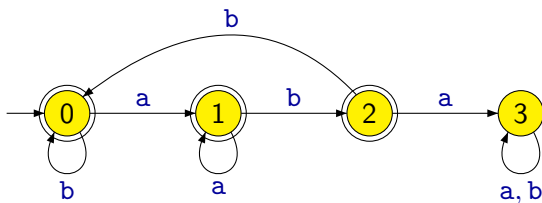
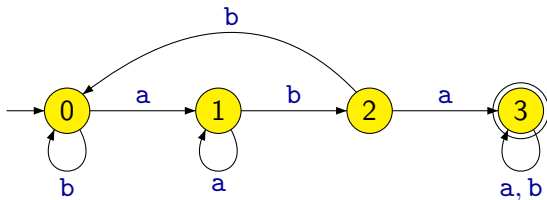
Theorem

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 \bar{L} is regular.