CENG 2010 - Programming Language Concepts

Week 3: Regular Expressions and Lexical Analysis

Burak Ekici

March 20, 2022

Table of Contents

Definitions

- alphabet is finite set; its elements are called symbols or letters

Definitions

- alphabet is finite set; its elements are called symbols or letters
- string over alphabet Σ is finite sequence of elements of Σ

Example

strings over $\Sigma = \{0, 1\}$: 0   0110

Definitions

- alphabet is finite set; its elements are called symbols or letters
- string over alphabet Σ is finite sequence of elements of Σ
- length $|x|$ of string x is number of symbols in x

Example

strings over $\Sigma = \{0, 1\}$: 0 0110

Definitions

- alphabet is finite set; its elements are called symbols or letters
- string over alphabet Σ is finite sequence of elements of Σ
- length $|x|$ of string x is number of symbols in x
- empty string is unique string of length 0 and denoted by ϵ

Example

strings over $\Sigma = \{0, 1\}$: 0   0110

Definitions

- alphabet is finite set; its elements are called symbols or letters
- string over alphabet Σ is finite sequence of elements of Σ
- length $|x|$ of string x is number of symbols in x
- empty string is unique string of length 0 and denoted by ε
- Σ^* is set of all strings over Σ ($\varnothing^* = \{\varepsilon\}$)

Example

strings over $\Sigma = \{0, 1\}$: 0   0110

Definitions

- alphabet is finite set; its elements are called symbols or letters
- string over alphabet Σ is finite sequence of elements of Σ
- length $|x|$ of string x is number of symbols in x
- empty string is unique string of length 0 and denoted by ε
- Σ^* is set of all strings over Σ ($\varnothing^* = \{\varepsilon\}$)
- language $\mathcal{L}$ over Σ is subset of Σ^*

Example

strings over $\Sigma = \{0, 1\} : 0 \quad 0110$
languages over Σ:

- $\{\varepsilon, 0, 1, 00, 01, 10, 11\}$ (all strings having at most two symbols)

Definitions

- alphabet is finite set; its elements are called symbols or letters
- string over alphabet Σ is finite sequence of elements of Σ
- length $|x|$ of string x is number of symbols in x
- empty string is unique string of length 0 and denoted by ε
- Σ^* is set of all strings over Σ ($\varnothing^* = \{\varepsilon\}$)
- language $\mathcal{L}$ over Σ is subset of Σ^*

Example

strings over $\Sigma = \{0, 1\} : 0 \quad 0110$
languages over Σ:

- $\{\varepsilon, 0, 1, 00, 01, 10, 11\}$ (all strings having at most two symbols)
- $\{x \mid x$ is valid program in some machine language$\}$

Definitions (Operations on Languages)

Let Σ be an alphabet and let $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ be languages over Σ

Definitions (Operations on Languages)

Let Σ be an alphabet and let $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ be languages over Σ

Concatenation $\quad \mathcal{L}_1\mathcal{L}_2 \quad := \quad \{xy \mid x \in \mathcal{L}_1 \ \wedge \ y \in \mathcal{L}_2\}$

Definitions (Operations on Languages)

Let Σ be an alphabet and let $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ be languages over Σ

$$\begin{array}{lll} \text{Concatenation} & \mathcal{L}_1\mathcal{L}_2 & := \left\{xy \mid x \in \mathcal{L}_1 \ \land \ y \in \mathcal{L}_2\right\} \\ \text{Union} & \mathcal{L}_1 \cup \mathcal{L}_2 & := \left\{x \mid x \in \mathcal{L}_1 \ \lor \ x \in \mathcal{L}_2\right\} \end{array}$$

Definitions (Operations on Languages)

Let Σ be an alphabet and let $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ be languages over Σ

$$
\begin{array}{lll}
\text{Concatenation} & \mathcal{L}_1\mathcal{L}_2 & := & \left\{xy \mid x \in \mathcal{L}_1 \;\wedge\; y \in \mathcal{L}_2\right\} \\
\text{Union} & \mathcal{L}_1 \cup \mathcal{L}_2 & := & \left\{x \mid x \in \mathcal{L}_1 \;\vee\; x \in \mathcal{L}_2\right\} \\
\text{Intersection} & \mathcal{L}_1 \cap \mathcal{L}_2 & := & \left\{x \mid x \in \mathcal{L}_1 \;\wedge\; x \in \mathcal{L}_2\right\}
\end{array}
$$

Definitions (Operations on Languages)

Let Σ be an alphabet and let $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ be languages over Σ

Concatenation	$\mathcal{L}_1\mathcal{L}_2$	$:=$	$\{xy \mid x \in \mathcal{L}_1 \ \wedge \ y \in \mathcal{L}_2\}$
Union	$\mathcal{L}_1 \cup \mathcal{L}_2$	$:=$	$\{x \mid x \in \mathcal{L}_1 \ \vee \ x \in \mathcal{L}_2\}$
Intersection	$\mathcal{L}_1 \cap \mathcal{L}_2$	$:=$	$\{x \mid x \in \mathcal{L}_1 \ \wedge \ x \in \mathcal{L}_2\}$
Kleene star	$\Sigma^* = \mathcal{L}$	$:=$	$\{x \mid x = \varepsilon \ \vee \ x \in \mathcal{L} \ \vee \ x \in \mathcal{L}\mathcal{L} \ \vee \ x \in \mathcal{L}\mathcal{L}\mathcal{L} \ \vee \ \ldots\}$

Definitions (Operations on Languages)

Let Σ be an alphabet and let $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ be languages over Σ

Concatenation	$\mathcal{L}_1\mathcal{L}_2$	$:=$	$\left\{xy \mid x \in \mathcal{L}_1 \ \wedge \ y \in \mathcal{L}_2\right\}$
Union	$\mathcal{L}_1 \cup \mathcal{L}_2$	$:=$	$\left\{x \mid x \in \mathcal{L}_1 \ \vee \ x \in \mathcal{L}_2\right\}$
Intersection	$\mathcal{L}_1 \cap \mathcal{L}_2$	$:=$	$\left\{x \mid x \in \mathcal{L}_1 \ \wedge \ x \in \mathcal{L}_2\right\}$
Kleene star	$\Sigma^* = \mathcal{L}$	$:=$	$\left\{x \mid x = \varepsilon \ \vee \ x \in \mathcal{L} \ \vee \ x \in \mathcal{L}\mathcal{L} \ \vee \ x \in \mathcal{L}\mathcal{L}\mathcal{L} \ \vee \ \ldots\right\}$
Kleene plus	Σ^+	$:=$	$\Sigma^* - \{\varepsilon\}$

Example (Operations on Languages)

- let

$$
\begin{array}{lll}
\Sigma & = & \{a, b, c, d\} \\
\mathcal{L}_1 & = & \{a, ab, c, d, \varepsilon\} \\
\mathcal{L}_2 & = & \{d\} \\
\mathcal{L}_3 & := & \mathcal{L}_1\mathcal{L}_2 \\
\mathcal{L}_4 & := & \mathcal{L}_1 \cup \mathcal{L}_2
\end{array}
$$

Example (Operations on Languages)

- let

$$\begin{array}{lll} \Sigma & = & \{a, b, c, d\} \\ \mathcal{L}_1 & = & \{a, ab, c, d, \varepsilon\} \\ \mathcal{L}_2 & = & \{d\} \\ \mathcal{L}_3 & := & \mathcal{L}_1\mathcal{L}_2 \\ \mathcal{L}_4 & := & \mathcal{L}_1 \cup \mathcal{L}_2 \end{array}$$

- which of the following strings are in $\mathcal{L}_3$? – a, abd, cd, d

Example (Operations on Languages)

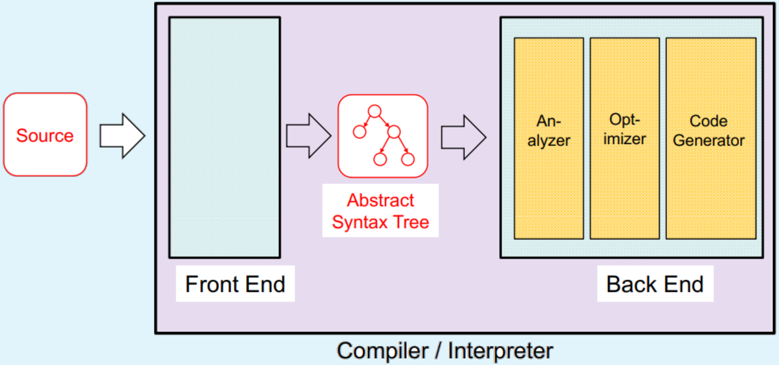
- let

$$\begin{array}{lcl} \Sigma & = & \{a, b, c, d\} \\ \mathcal{L}_1 & = & \{a, ab, c, d, \varepsilon\} \\ \mathcal{L}_2 & = & \{d\} \\ \mathcal{L}_3 & := & \mathcal{L}_1\mathcal{L}_2 \\ \mathcal{L}_4 & := & \mathcal{L}_1 \cup \mathcal{L}_2 \end{array}$$

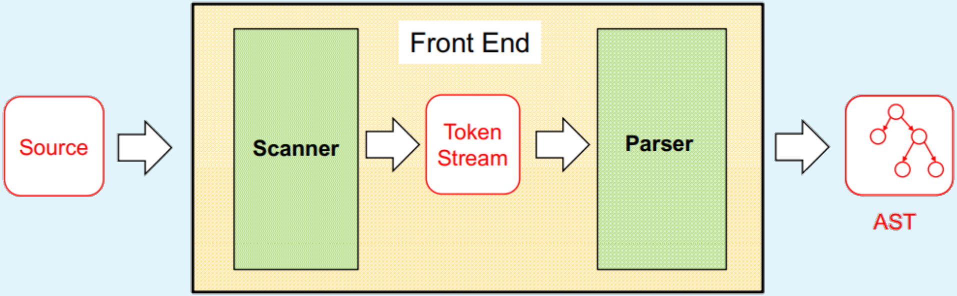
- which of the following strings are in $\mathcal{L}_3$? – a, abd, cd, d
- which of the following strings are in $\mathcal{L}_4$? – a, abd, cd, d

Table of Contents

Architecture of Compilers and Interpreters

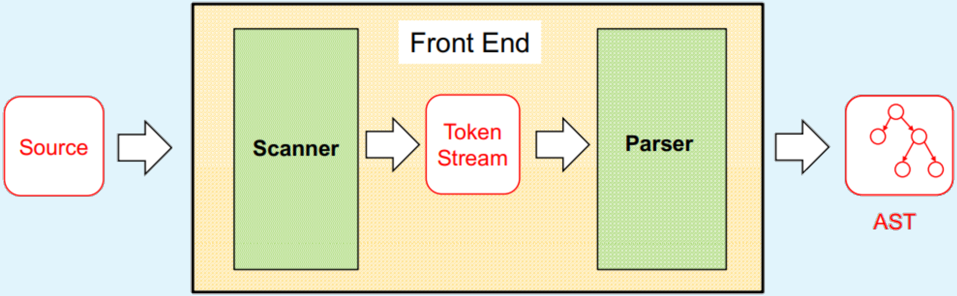


Front-End: Lexer and Parser



scanner/lexer : source code $\mapsto$ tokens (keywords, variables, numbers, etc.) regular expressions

Front-End: Lexer and Parser



scanner/lexer	:	source code	$\mapsto$	tokens (keywords, variables, numbers, etc.)	regular expressions
parser	:	tokens	$\mapsto$	ATSs	context free grammars

Implementation (Front-End)

- goal: map program texts into PTs/ASTs

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program
- front end use regular expressions at scanning/lexing

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program
- front end use regular expressions at scanning/lexing
- regular expressions cannot reliably parse paired braces $\{\{\ldots\}\}$ and parentheses $(((\ldots)))$, etc.

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program
- front end use regular expressions at scanning/lexing
- regular expressions cannot reliably parse paired braces $\{\{\ldots\}\}$ and parentheses $(((\ldots)))$, etc.
- regular expressions for tokenizing (scanning/lexing), and context free grammars for parsing tokens

Table of Contents

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

^	matches beginning of line

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

^	matches beginning of line
$	matches end of line

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

^	matches beginning of line
$	matches end of line
c	matches character c

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

^	matches beginning of line	.	matches any character
$	matches end of line		
c	matches character c		

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

^	matches beginning of line	.	matches any character
$	matches end of line	`[abc]`	matches a or b or c
c	matches character c		

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

^	matches beginning of line	`.`	matches any character
$	matches end of line	`[abc]`	matches a or b or c
c	matches character c	`[a-zA-Z]`	matches any letter

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

^	matches beginning of line	`.`	matches any character
$	matches end of line	`[abc]`	matches a or b or c
c	matches character c	`[a-zA-Z]`	matches any letter

Example

`grep "0" file` returns lines containing 0

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

`^`	matches beginning of line	`.`	matches any character
`$`	matches end of line	`[abc]`	matches a or b or c
`c`	matches character c	`[a-zA-Z]`	matches any letter

Example

`grep "0" file`	returns lines containing 0
`grep "0$" file`	returns lines ending with 0

Applications of Regular expressions: `grep`

- `grep foo file` returns lines in file containing pattern foo
- basis for more powerful tools like awk, `sed`, `perl`

Some Patterns

`^`	matches beginning of line	`.`	matches any character
`$`	matches end of line	`[abc]`	matches a or b or c
`c`	matches character c	`[a-zA-Z]`	matches any letter

Example

`grep "0" file`	returns lines containing 0
`grep "0$" file`	returns lines ending with 0
`grep "b.g" file`	returns lines containing e.g. bag, big, bug, buggy

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$

-

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$

-

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\emptyset}$	$\emptyset$

-

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\varnothing}$	$\varnothing$
$\#$	Σ

-

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\varnothing}$	$\varnothing$
$\#$	Σ
@	Σ^*

-

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\varnothing}$	$\varnothing$
$\#$	Σ
@	Σ^*

compound pattern α	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\varnothing}$	$\varnothing$
$\#$	Σ
$@$	Σ^*

compound pattern α	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$

-

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\emptyset}$	$\emptyset$
$\#$	Σ
$@$	Σ^*

compound pattern α	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$\beta\gamma$	$L(\beta)L(\gamma)$

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$
-

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\emptyset}$	$\emptyset$
$\#$	Σ
@	Σ^*

compound pattern α	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$\beta\gamma$	$L(\beta)L(\gamma)$
β^*	$L(\beta)^*$

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\varnothing}$	$\varnothing$
$\#$	Σ
$@$	Σ^*

compound pattern α	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$\beta\gamma$	$L(\beta)L(\gamma)$
β^*	$L(\beta)^*$
β^+	$L(\beta)^+$

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\varnothing}$	$\varnothing$
$\#$	Σ
@	Σ^*

compound pattern α	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$\beta\gamma$	$L(\beta)L(\gamma)$
β^*	$L(\beta)^*$
β^+	$L(\beta)^+$
$\sim\beta$	$\sim L(\beta) = \Sigma^* - L(\beta)$

Pattern matching is important for

- lexical analysis of programs
- scripting languages (Perl, Ruby)
- search engines (Google Code Search)
- DNA analysis

Definitions

- pattern is string α that represents set of strings $L(\alpha) \subseteq \Sigma^*$
-

atomic pattern α	$L(\alpha)$
$\mathbf{a} \in \Sigma$	$\{a\}$
$\boldsymbol{\varepsilon}$	$\{\varepsilon\}$
$\boldsymbol{\varnothing}$	$\varnothing$
$\#$	Σ
$@$	Σ^*

compound pattern α	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$\beta\gamma$	$L(\beta)L(\gamma)$
β^*	$L(\beta)^*$
β^+	$L(\beta)^+$
$\sim \beta$	$\sim L(\beta) = \Sigma^* - L(\beta)$

- string $x \in \Sigma^*$ matches pattern α if $x \in L(\alpha)$

Example

pattern	matched string
@a@a@a@	strings containing at least 3 occurrences of a

Example

pattern	matched string
@a@a@a@	strings containing at least 3 occurrences of a
@a@b@	strings containing a followed later by b

Example

pattern	matched string
$@a@a@a@$	strings containing at least 3 occurrences of a
$@a@b@$	strings containing a followed later by b
$\# \cap \sim a$	single letters except a

Example

pattern	matched string
$@a@a@a@$	strings containing at least 3 occurrences of a
$@a@b@$	strings containing a followed later by b
$\# \cap \sim a$	single letters except a
$(\# \cap \sim a)^*$	strings without a

Questions

- which operators are redundant?

Questions

- which operators are redundant?

$$\varepsilon \quad \equiv \quad \sim(\#@) \equiv \emptyset^*$$

Questions

- which operators are redundant?

$$
\begin{aligned}
\varepsilon &\equiv \sim(\#@) \equiv \emptyset^* \\
@ &\equiv \#^*
\end{aligned}
$$

Questions

- which operators are redundant?

$$
\begin{aligned}
\varepsilon &\equiv \sim(\#@) \equiv \varnothing^* \\
@ &\equiv \#^* \\
\alpha^+ &\equiv \alpha\alpha^*
\end{aligned}
$$

Questions

- which operators are redundant?

$$
\begin{array}{lll}
\varepsilon & \equiv & \sim(\#@) \equiv \varnothing^* \\
@ & \equiv & \#^* \\
\alpha^+ & \equiv & \alpha\alpha^* \\
\# & \equiv & a_1 \ldots a_n \qquad \text{if } \Sigma = \{a_1 \ldots a_n\}
\end{array}
$$

Questions

- which operators are redundant?

$$
\begin{array}{lll}
\boldsymbol{\varepsilon} & \equiv & \sim(\#@) \equiv \boldsymbol{\varnothing}^* \\
@ & \equiv & \#^* \\
\alpha^+ & \equiv & \alpha\alpha^* \\
\# & \equiv & a_1 \ldots a_n \qquad \text{if } \Sigma = \{a_1 \ldots a_n\} \\
\alpha \cap \beta & \equiv & \sim(\sim\alpha + \sim\beta)
\end{array}
$$

Questions

- which operators are redundant?

$$
\begin{array}{lll}
\varepsilon & \equiv & \sim(\#@) \equiv \emptyset^* \\
@ & \equiv & \#^* \\
\alpha^+ & \equiv & \alpha\alpha^* \\
\# & \equiv & a_1 \dots a_n \qquad \text{if } \Sigma = \{a_1 \dots a_n\} \\
\alpha \cap \beta & \equiv & \sim(\sim\alpha + \sim\beta) \\
\sim\alpha & \equiv & ?
\end{array}
$$

Questions

- which operators are redundant?

$$
\begin{array}{lll}
\boldsymbol{\varepsilon} & \equiv & \sim(\#@) \equiv \boldsymbol{\varnothing}^* \\
@ & \equiv & \#^* \\
\alpha^+ & \equiv & \alpha\alpha^* \\
\# & \equiv & a_1 \ldots a_n \qquad \text{if } \Sigma = \{a_1 \ldots a_n\} \\
\alpha \cap \beta & \equiv & \sim(\sim\alpha + \sim\beta) \\
\sim\alpha & \equiv & ?
\end{array}
$$

Notation

$\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$

Definition

regular expressions are restricted patterns which use only

$$\boldsymbol{a} \in \Sigma \quad \boldsymbol{\varepsilon} \quad \boldsymbol{\emptyset} \quad \alpha + \beta \quad \alpha^* \quad \alpha\beta$$

Definition

regular expressions are restricted patterns which use only

$$\boldsymbol{a} \in \Sigma \quad \boldsymbol{\varepsilon} \quad \boldsymbol{\emptyset} \quad \alpha + \beta \quad \alpha^* \quad \alpha\beta$$

Remark (Precedence)

Kleene closure * > concatenation > union $+$

$$\begin{array}{lcl}
\boldsymbol{ab} + \boldsymbol{c} & := & (\boldsymbol{ab}) + \boldsymbol{c} & = & \{ab, c\} \\
\boldsymbol{ab}^* & := & \boldsymbol{a}(\boldsymbol{b}^*) & = & \{a, ab, abb, \ldots\} \\
\boldsymbol{a} + \boldsymbol{b}^* & := & \boldsymbol{a} + (\boldsymbol{b}^*) & = & \{a, \varepsilon, b, bb, bbb, \ldots\}
\end{array}$$

Definition

a language A is called regular if it is generated by a regular expression α. Namely $A = L(\alpha)$

Example

$$
\begin{array}{llll}
\boldsymbol{a} & & & = \{a\} \\
\boldsymbol{a} + \boldsymbol{b} & := & \{a\} \cup \{b\} & = \{a, b\} \\
\boldsymbol{a}^* & := & \{\varepsilon\} \cup \{a\} \cup \{aa\} \cup \{aaa\} \cup \ \ldots & = \{\varepsilon, a, aa, aaa, \ \ldots\} \\
\boldsymbol{ab}^* (\boldsymbol{c} + \boldsymbol{\varepsilon}) & & & = \{a, ac, ab, abc, abb, abbc, \ \ldots\}
\end{array}
$$

Theorem

patterns, and regular expressions and regular languages are equivalent:

Theorem

patterns, and regular expressions and regular languages are equivalent:

for all	$A \subseteq \Sigma^*$	➊ A is regular
	$\iff$	➋ $A = L(\alpha)$ for some pattern α
	$\iff$	➌ $A = L(\alpha)$ for some regular expression α

Implementing Regular Expressions

- by finite automaton – “machines” recognize regular languages

Implementing Regular Expressions

- by finite automaton – "machines" recognize regular languages

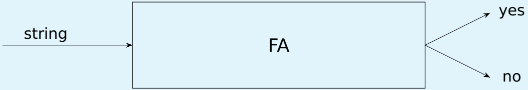


Table of Contents

Definitions

- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with

Definitions

- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states

Definitions

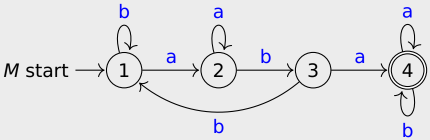
- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet

Definitions

- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\delta : Q \times \Sigma \to Q$: transition function

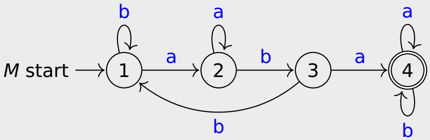
Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



Example (DFAs → Regular Sets)

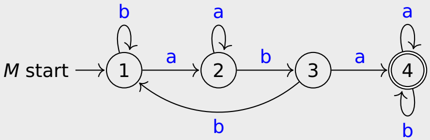
$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$

Example (DFAs → Regular Sets)

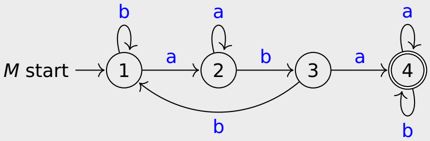
$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$

Example (DFAs → Regular Sets)

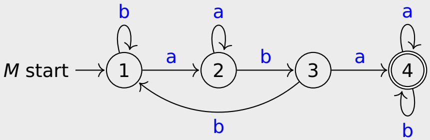
$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Definitions

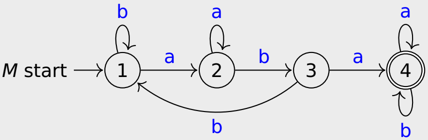
- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\delta : Q \times \Sigma \to Q$: transition function
  4. $s \in Q$: start state

Definitions

- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\delta : Q \times \Sigma \rightarrow Q$: transition function
  4. $s \in Q$: start state
  5. $F \subseteq Q$: final (accept) states

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$

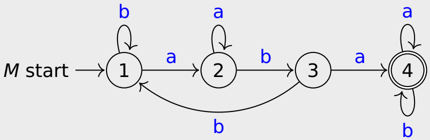


1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$

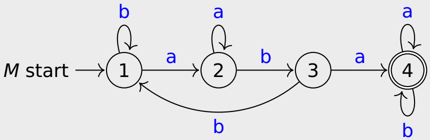


1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$

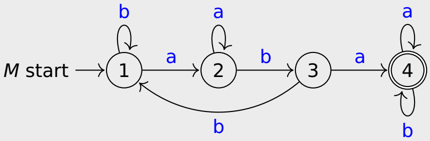


1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



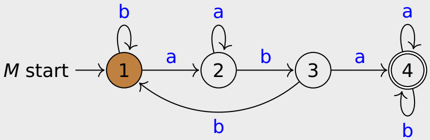
1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b	a	b	a	a
1	1	2	3	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



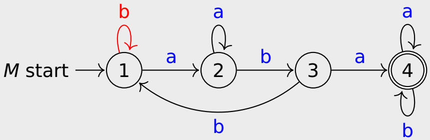
1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b	a	b	a	a	
1	1	2	3	4	4	

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



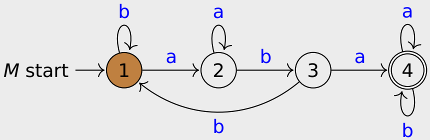
b	a	b	a	a	
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



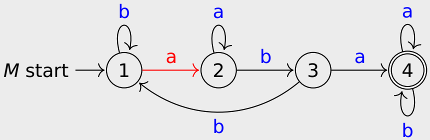
1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b	a	b	a	a
1	1	2	3	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



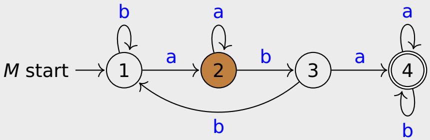
b	a	b	a	a	
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



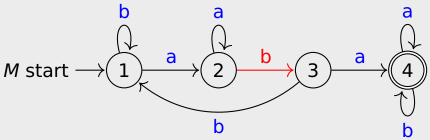
b	a	b	a	a	
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



b a b a a

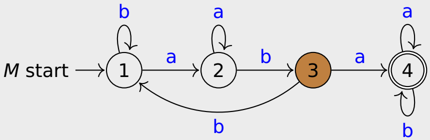
1 1 2 3 4 4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



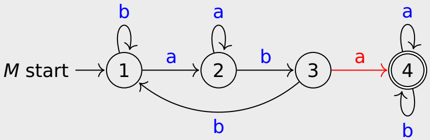
b	a	b	a	a	
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



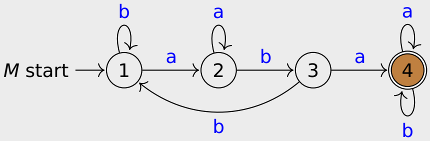
1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b	a	b	a	a	
1	1	2	3	4	4	

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



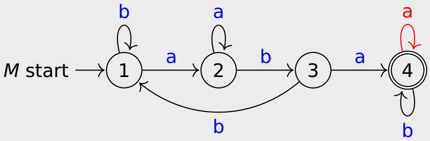
	b	a	b	a	a
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



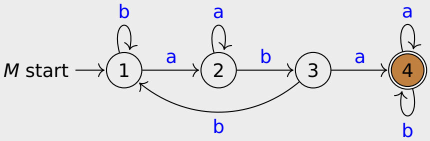
	b	a	b	a	a
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



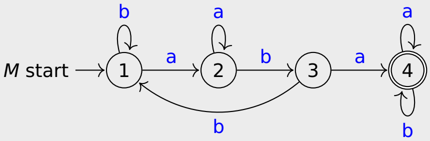
b	a	b	a	a	
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



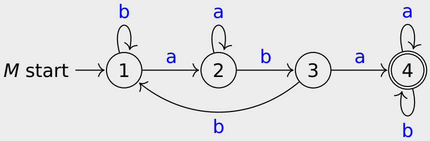
	b	a	b	a	a
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



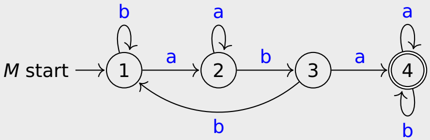
b	a	b	a	a	
1	1	2	3	4	4

1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

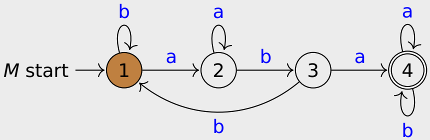
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

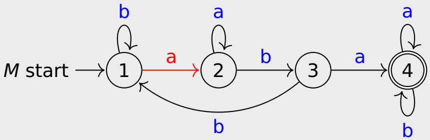
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

b a b a a

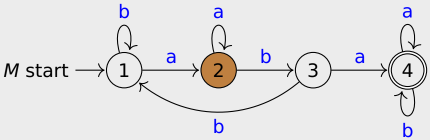
1 1 2 3 4 4

a a b b b

1 2 2 3 1 1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

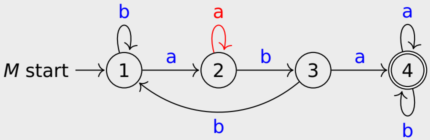
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b	a	b	a	a
1	1	2	3	4	4

	a	a	b	b	b
1	2	2	3	1	1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

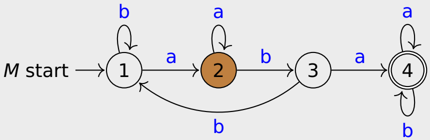
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

b a b a a

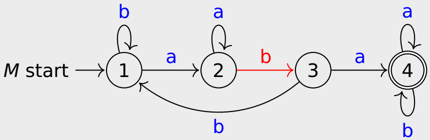
1 1 2 3 4 4

a a b b b

1 2 2 3 1 1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

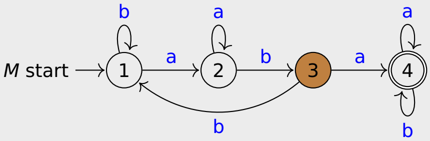
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

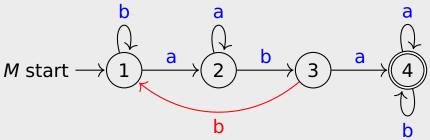
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

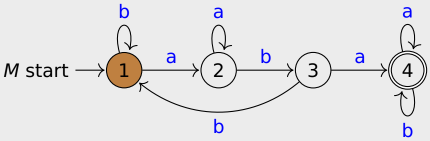
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b	a	b	a	a
1	1	2	3	4	4

	a	a	b	b	b
1	2	2	3	1	1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

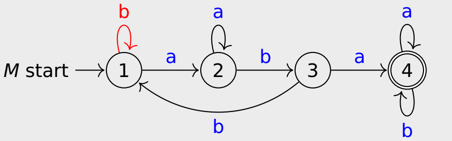
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

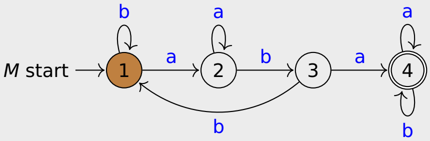
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b	a	b	a	a
1	1	2	3	4	4

	a	a	b	b	b
1	2	2	3	1	1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

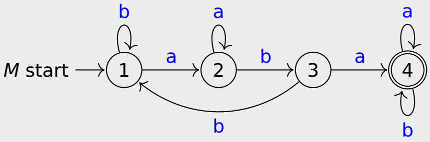
δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

	b		a		b		a		a	
1		1		2		3		4		4

	a		a		b		b		b	
1		2		2		3		1		1

Definitions

- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\delta : Q \times \Sigma \to Q$: transition function
  4. $s \in Q$: start state
  5. $F \subseteq Q$: final (accept) states
- $\widehat{\delta}: Q \times \Sigma^* \to Q$ is inductively defined by

$$\widehat{\delta}(q, \varepsilon) := q \qquad \widehat{\delta}(q, xa) := \delta(\widehat{\delta}(q, x), a)$$

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$ first recursive call

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(q_0, a), b), b), a), a), b)$	

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(q_0, a), b), b), a), a), b)$	
$\delta(\delta(\delta(\delta(\delta(q_1, b), b), a), a), b)$	assuming $\delta(q_0, a) = q_1$

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(q_0, a), b), b), a), a), b)$	
$\delta(\delta(\delta(\delta(\delta(q_1, b), b), a), a), b)$	assuming $\delta(q_0, a) = q_1$
$\delta(\delta(\delta(\delta(q_2, b), a), a), b)$	assuming $\delta(q_1, b) = q_2$

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(q_0, a), b), b), a), a), b)$	
$\delta(\delta(\delta(\delta(\delta(q_1, b), b), a), a), b)$	assuming $\delta(q_0, a) = q_1$
$\delta(\delta(\delta(\delta(q_2, b), a), a), b)$	assuming $\delta(q_1, b) = q_2$
$\delta(\delta(\delta(q_3, a), a), b)$	assuming $\delta(q_2, b) = q_3$

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(q_0, a), b), b), a), a), b)$	
$\delta(\delta(\delta(\delta(\delta(q_1, b), b), a), a), b)$	assuming $\delta(q_0, a) = q_1$
$\delta(\delta(\delta(\delta(q_2, b), a), a), b)$	assuming $\delta(q_1, b) = q_2$
$\delta(\delta(\delta(q_3, a), a), b)$	assuming $\delta(q_2, b) = q_3$
$\delta(\delta(q_4, a), b)$	assuming $\delta(q_3, a) = q_4$

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(q_0, a), b), b), a), a), b)$	
$\delta(\delta(\delta(\delta(\delta(q_1, b), b), a), a), b)$	assuming $\delta(q_0, a) = q_1$
$\delta(\delta(\delta(\delta(q_2, b), a), a), b)$	assuming $\delta(q_1, b) = q_2$
$\delta(\delta(\delta(q_3, a), a), b)$	assuming $\delta(q_2, b) = q_3$
$\delta(\delta(q_4, a), b)$	assuming $\delta(q_3, a) = q_4$
$\delta(q_5, b)$	assuming $\delta(q_4, a) = q_5$

Example (Unfolding of the multistep function $\widehat{\delta}$)

Let $x = abbaab$ over the alphabet $\Sigma = \{a, b\}$

$\delta(\widehat{\delta}(q_0, abbaa), b)$	first recursive call
$\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$	second recursive call
$\delta(\delta(\delta(\widehat{\delta}(q_0, abb), a), a), b)$	third recursive call
$\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, ab), b), a), a), b)$	fourth recursive call
$\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, a), b), b), a), a), b)$	fifth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \varepsilon), a), b), b), a), a), b)$	sixth recursive call
$\delta(\delta(\delta(\delta(\delta(\delta(q_0, a), b), b), a), a), b)$	
$\delta(\delta(\delta(\delta(\delta(q_1, b), b), a), a), b)$	assuming $\delta(q_0, a) = q_1$
$\delta(\delta(\delta(\delta(q_2, b), a), a), b)$	assuming $\delta(q_1, b) = q_2$
$\delta(\delta(\delta(q_3, a), a), b)$	assuming $\delta(q_2, b) = q_3$
$\delta(\delta(q_4, a), b)$	assuming $\delta(q_3, a) = q_4$
$\delta(q_5, b)$	assuming $\delta(q_4, a) = q_5$
q_6	assuming $\delta(q_5, b) = q_6$

Definitions

- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\delta : Q \times \Sigma \to Q$: transition function
  4. $s \in Q$: start state
  5. $F \subseteq Q$: final (accept) states
- $\widehat{\delta}: Q \times \Sigma^* \to Q$ is inductively defined by

$$\widehat{\delta}(q, \varepsilon) := q \qquad \widehat{\delta}(q, xa) := \delta(\widehat{\delta}(q, x), a)$$

- string $x \in \Sigma^*$ is accepted by M if $\widehat{\delta}(s, x) \in F$

Definitions

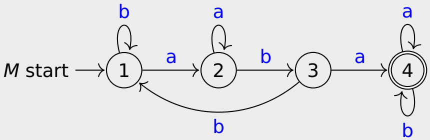
- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\delta : Q \times \Sigma \to Q$: transition function
  4. $s \in Q$: start state
  5. $F \subseteq Q$: final (accept) states
- $\widehat{\delta} : Q \times \Sigma^* \to Q$ is inductively defined by

$$\widehat{\delta}(q, \epsilon) := q \qquad\qquad \widehat{\delta}(q, xa) := \delta(\widehat{\delta}(q, x), a)$$

- string $x \in \Sigma^*$ is accepted by M if $\widehat{\delta}(s, x) \in F$
- string $x \in \Sigma^*$ is rejected by M if $\widehat{\delta}(s, x) \notin F$

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

b a b a a $\in L(M)$

1 1 2 3 4 4

a a b b b $\notin L(M)$

1 2 2 3 1 1

Definitions

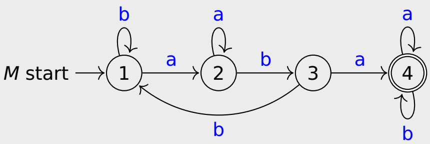
- deterministic finite automaton (DFA) is quintuple $M = (Q, \Sigma, \delta, s, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\delta : Q \times \Sigma \to Q$: transition function
  4. $s \in Q$: start state
  5. $F \subseteq Q$: final (accept) states
- $\widehat{\delta}: Q \times \Sigma^* \to Q$ is inductively defined by

$$\widehat{\delta}(q, \varepsilon) := q \qquad \widehat{\delta}(q, xa) := \delta(\widehat{\delta}(q, x), a)$$

- string $x \in \Sigma^*$ is accepted by M if $\widehat{\delta}(s, x) \in F$
- string $x \in \Sigma^*$ is rejected by M if $\widehat{\delta}(s, x) \notin F$
- language accepted by M is given by $L(M) := \{x \mid \widehat{\delta}(s, x) \in F\}$

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

b a b a a $\in L(M)$

1 1 2 3 4 4

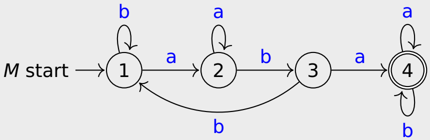
a a b b b $\notin L(M)$

1 2 2 3 1 1

$L(M) := \{x \mid \qquad \}$

Example (DFAs → Regular Sets)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3, 4\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{4\}$

δ	a	b
1	2	1
2	2	3
3	4	1
4	4	4

b a b a a $\in L(M)$

1 1 2 3 4 4

a a b b b $\notin L(M)$

1 2 2 3 1 1

$L(M) := \{x \mid x$ contains aba as substring$\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

2. $\Sigma = \{a, b\}$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

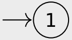
$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

M start 

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

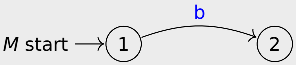
$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

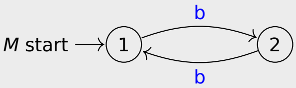


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

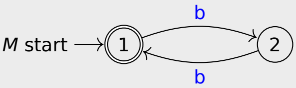


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x$ contains even number of bs over $\Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

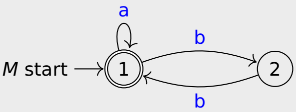


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

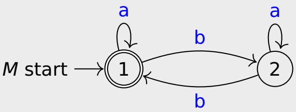


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

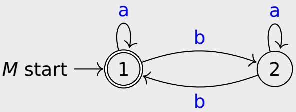


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



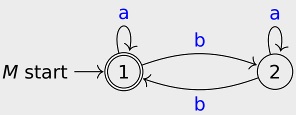
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a		b		a		b		a	
1	1	2	2	1	1				

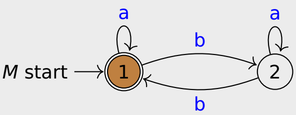
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a

1 1 2 2 1 1

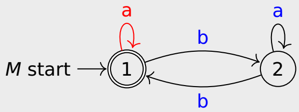
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a

1 1 2 2 1 1

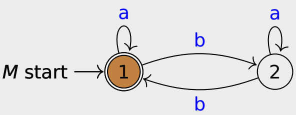
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a
1	1	2	2	1	1

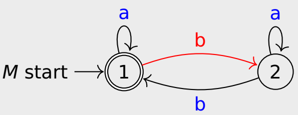
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a

1 1 2 2 1 1

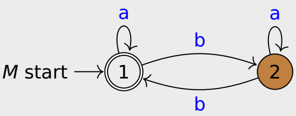
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a	
1	1	2	2	1	1	

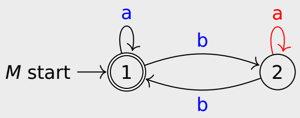
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a	b	a	b	a	
1	1	2	2	1	1

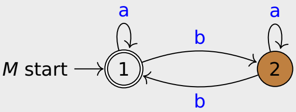
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a	b	a	b	a	
1	1	2	2	1	1

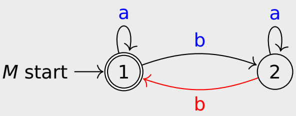
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a
1	1	2	2	1	1

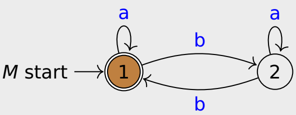
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a

1 1 2 2 1 1

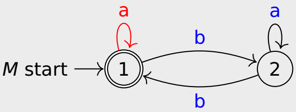
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a

1 1 2 2 1 1

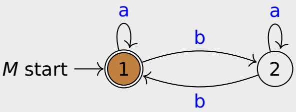
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a

1 1 2 2 1 1

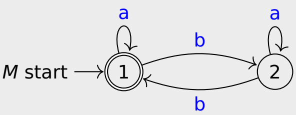
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a $\in L(M)$

1 1 2 2 1 1

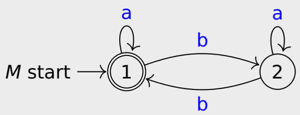
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a	$\in L(M)$
1	1	2	2	1	1	

	b	a	a	a	a
1	2	2	2	2	2

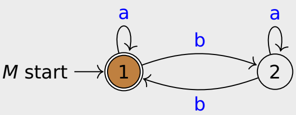
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a	$\in L(M)$
	1	2	2	1	1	
	b	a	a	a	a	
1	2	2	2	2	2	

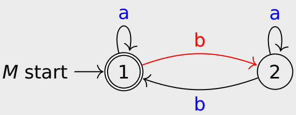
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a $\in L(M)$

1 2 2 1 1

b a a a a

1 2 2 2 2 2

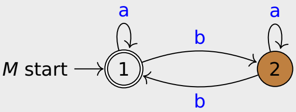
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a $\in L(M)$

1 2 2 1 1

b a a a a

1 2 2 2 2 2

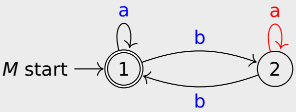
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a $\in L(M)$

1 2 2 1 1

b a a a a

1 2 2 2 2 2

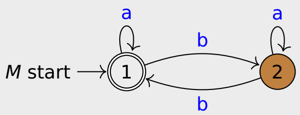
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a	$\in L(M)$
	1	2	2	1	1	
	b	a	a	a	a	
1	2	2	2	2	2	

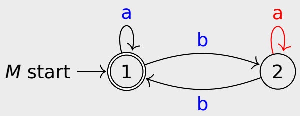
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a	$\in L(M)$
1	2	2	1	1		

	b	a	a	a	a
1	2	2	2	2	2

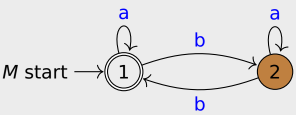
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	a	b	a	b	a	$\in L(M)$
	1	2	2	1	1	

	b	a	a	a	a
1	2	2	2	2	2

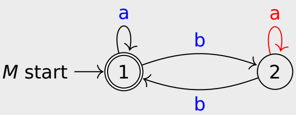
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a	b	a	b	a	$\in L(M)$
1	2	2	1	1	

	b	a	a	a	a
1	2	2	2	2	2

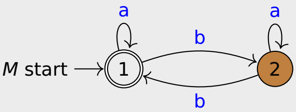
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a $\in L(M)$

1 2 2 1 1

b a a a a

1 2 2 2 2 2

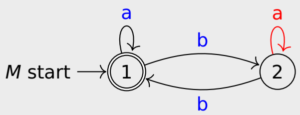
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a $\in L(M)$

1 2 2 1 1

b a a a a

1 2 2 2 2 2

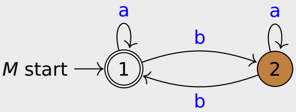
1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



a b a b a $\in L(M)$

1 2 2 1 1

1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

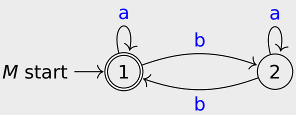
b a a a a

1 2 2 2 2 2

$L(M) := \{x \mid x \text{ contains even number of } b\text{s over } \Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{1\}$

δ	a	b
1	1	2
2	2	1

a b a b a $\in L(M)$

1 2 2 1 1

b a a a a $\notin L(M)$

1 2 2 2 2 2

$L(M) := \{x \mid x$ contains even number of bs over $\Sigma\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

2. $\Sigma = \{a, b\}$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

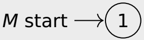
$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

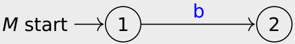
$M = (Q, \Sigma, \delta, s, F)$

1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

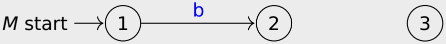


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

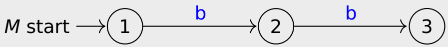


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

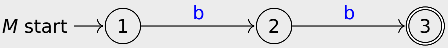


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

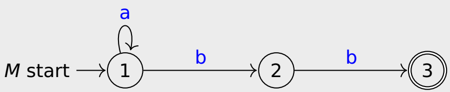


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

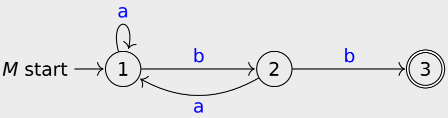


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

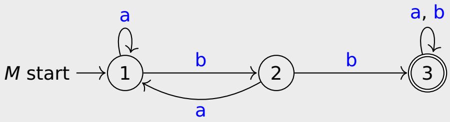


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$

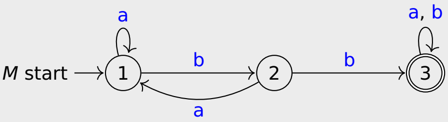


1. $Q =$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s =$
5. $F =$

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



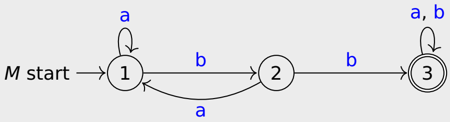
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

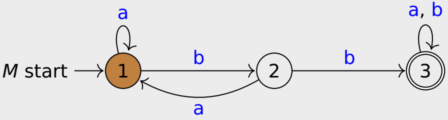
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

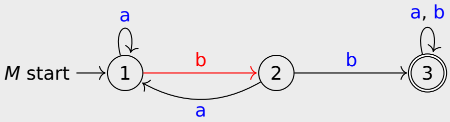
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b	a	b	b	a	
1	2	1	2	3	3

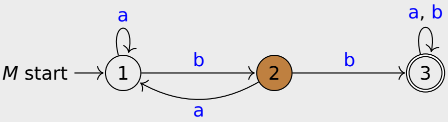
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	b	a	b	b	a
1	2	1	2	3	3

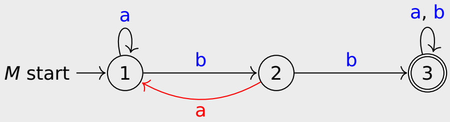
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

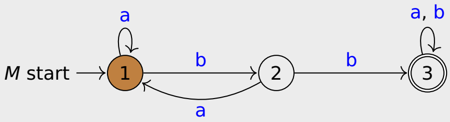
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

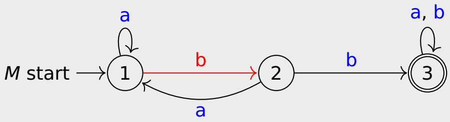
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

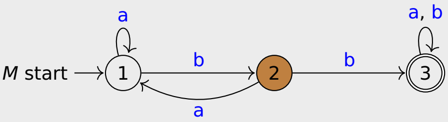
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b	a	b	b	a	
1	2	1	2	3	3

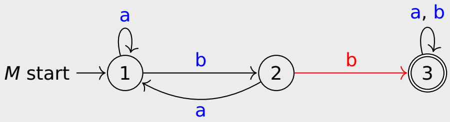
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b	a	b	b	a	
1	2	1	2	3	3

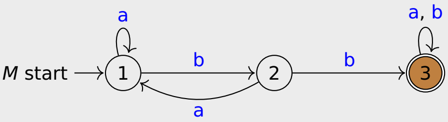
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

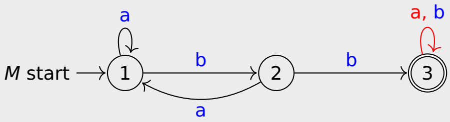
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

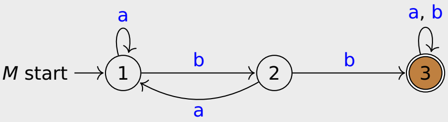
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



b a b b a

1 2 1 2 3 3

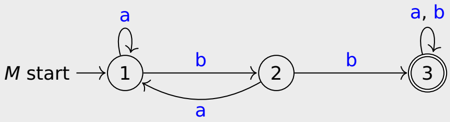
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	

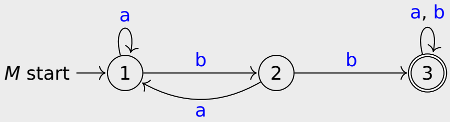
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	

	a	a	b	a	b
1	1	1	2	1	2

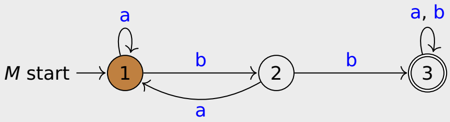
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

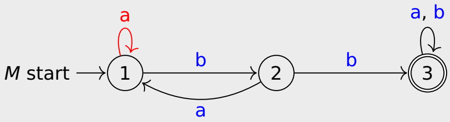
δ	a	b
1	1	2
2	1	3
3	3	3

	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	
	a	a	b	a	b	
1	1	1	2	1	2	

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

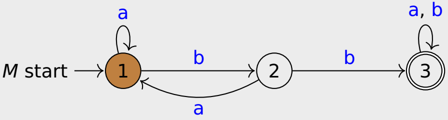
	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	

	a	a	b	a	b
1	1	1	2	1	2

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

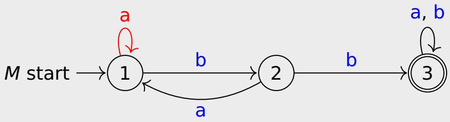
	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	

	a	a	b	a	b
1	1	1	2	1	2

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

b a b b a $\in L(M)$

1 2 1 2 3 3

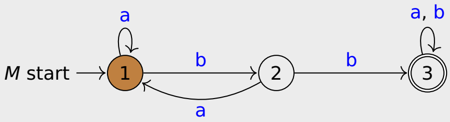
a a b a b

1 1 1 2 1 2

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

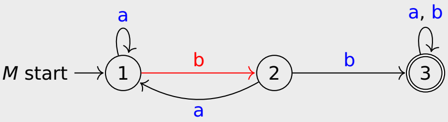
δ	a	b
1	1	2
2	1	3
3	3	3

	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	
	a	a	b	a	b	
1	1	1	2	1	2	

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	

	a	a	b	a	b
1	1	1	2	1	2

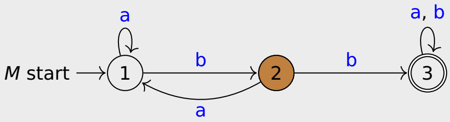
1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \rightarrow Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

b a b b a $\in L(M)$

1 2 1 2 3 3

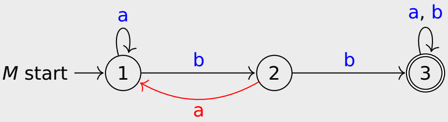
a a b a b

1 1 1 2 1 2

$L(M) := \{x \mid x$ contains bb as substring $\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

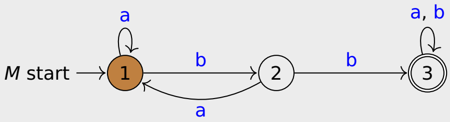
δ	a	b
1	1	2
2	1	3
3	3	3

	b	a	b	b	a	$\in L(M)$
1	2	1	2	3	3	
	a	a	b	a	b	
1	1	1	2	1	2	

$L(M) :=$ {$x \mid x$ contains *bb* as substring }

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

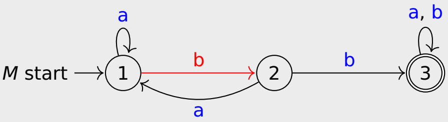
	b		a		b		b		a		$\in L(M)$
1		2		1		2		3		3	

	a		a		b		a		b	
1		1		1		2		1		2

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

b a b b a $\in L(M)$

1 2 1 2 3 3

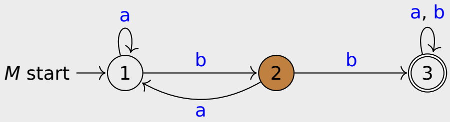
a a b a b

1 1 1 2 1 2

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

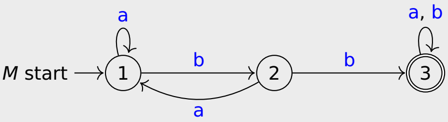
δ	a	b
1	1	2
2	1	3
3	3	3

	b	a	b	b	a $\in L(M)$
1	2	1	2	3	3
	a	a	b	a	b
1	1	1	2	1	2

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring }\}$

Example (Regular Language → DFA)

$M = (Q, \Sigma, \delta, s, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\delta : Q \times \Sigma \to Q$
4. $s = 1$
5. $F = \{3\}$

δ	a	b
1	1	2
2	1	3
3	3	3

	b		a		b		b		a		$\in L(M)$
1		2		1		2		3		3	

	a		a		b		a		b		$\notin L(M)$
1		1		1		2		1		2	

$L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$

Definitions

- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with

Definitions

- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with
  1. Q : finite set of states

Definitions

- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet

Definitions

- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\Delta : Q \times \Sigma \to 2^Q$: transition function

Definitions

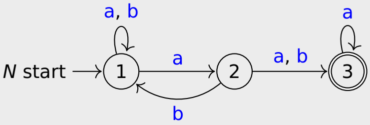
- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\Delta : Q \times \Sigma \rightarrow 2^Q$: transition function
  4. $S \subseteq Q$: set of start states

Definitions

- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\Delta : Q \times \Sigma \to 2^Q$: transition function
  4. $S \subseteq Q$: set of start states
  5. $F \subseteq Q$: final (accept) states

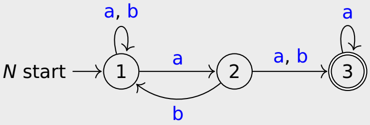
Example

$N = (Q, \Sigma, \Delta, S, F)$



Example

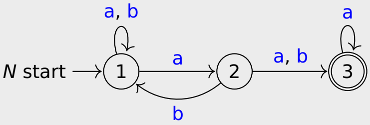
$N = (Q, \Sigma, \Delta, S, F)$



1. $Q = \{1, 2, 3\}$

Example

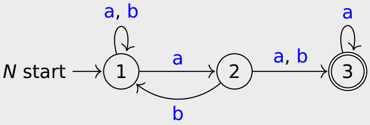
$N = (Q, \Sigma, \Delta, S, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$

Example

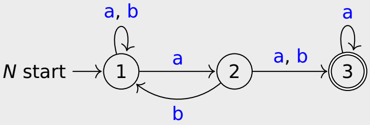
$N = (Q, \Sigma, \Delta, S, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\Delta : Q \times \Sigma \rightarrow 2^Q$

Example

$N = (Q, \Sigma, \Delta, S, F)$

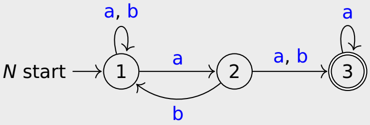


1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\Delta : Q \times \Sigma \to 2^Q$

Δ	a	b
1	$\{1, 2\}$	$\{1\}$
2	$\{3\}$	$\{1, 3\}$
3	$\{3\}$	$\varnothing$

Example

$N = (Q, \Sigma, \Delta, S, F)$

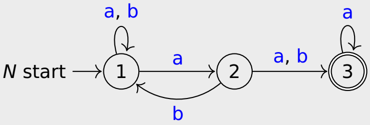


1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\Delta : Q \times \Sigma \to 2^Q$
4. $S = \{1\}$

Δ	a	b
1	$\{1, 2\}$	$\{1\}$
2	$\{3\}$	$\{1, 3\}$
3	$\{3\}$	$\varnothing$

Example

$N = (Q, \Sigma, \Delta, S, F)$



1. $Q = \{1, 2, 3\}$
2. $\Sigma = \{a, b\}$
3. $\Delta : Q \times \Sigma \to 2^Q$
4. $S = \{1\}$
5. $F = \{3\}$

Δ	a	b
1	$\{1, 2\}$	$\{1\}$
2	$\{3\}$	$\{1, 3\}$
3	$\{3\}$	$\varnothing$

Definitions

- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\Delta : Q \times \Sigma \to 2^Q$: transition function
  4. $S \subseteq Q$: set of start states
  5. $F \subseteq Q$: final (accept) states
- $\widehat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$ is inductively defined by

$$\widehat{\Delta}(A, \varepsilon) := A \qquad \widehat{\Delta}(A, xa) := \bigcup_{q \in \widehat{\Delta}(A,x)} \Delta(q, a)$$

Definitions

- nondeterministic finite automaton (NFA) is quintuple $N = (Q, \Sigma, \Delta, S, F)$ with
  1. Q : finite set of states
  2. Σ : input alphabet
  3. $\Delta : Q \times \Sigma \to 2^Q$: transition function
  4. $S \subseteq Q$: set of start states
  5. $F \subseteq Q$: final (accept) states
- $\widehat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$ is inductively defined by

$$\widehat{\Delta}(A, \varepsilon) := A \qquad \widehat{\Delta}(A, xa) := \bigcup_{q \in \widehat{\Delta}(A,x)} \Delta(q, a)$$

- string $x \in \Sigma^*$ is accepted by N if $\widehat{\Delta}(S, x) \cap F \neq \emptyset$

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA → DFA)

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA → DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA → DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA → DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

regular languages are closed under

- union

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA $\rightarrow$ DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

regular languages are closed under

- union
- intersection

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA → DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

regular languages are closed under

- union
- intersection
- complementation

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA → DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

regular languages are closed under

- union
- intersection
- complementation
- concatenation

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA $\longrightarrow$ DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

regular languages are closed under

- union
- intersection
- complementation
- concatenation
- asterate

Theorem

1. ever set accepted by NFA is regular (procedure: subset construction – NFA → DFA)
2. every DFA M minimizes into a DFA M' such that $L(M) = L(M')$ (unless M is minimal)

Theorem

regular languages are closed under

- union
- intersection
- complementation
- concatenation
- asterate
- homomorphic image and preimage

Table of Contents

Lexical Analyzers (Lexers)

translator tool : strings (programs) $\longrightarrow$ list of tokens (or error)

- simulate finite state automaton to create tokens

Lexical Analyzers (Lexers)

translator tool : strings (programs) $\longrightarrow$ list of tokens (or error)

- simulate finite state automaton to create tokens
- lexer generators

Lexical Analyzers (Lexers)

translator tool : strings (programs) $\rightarrow$ list of tokens (or error)

- simulate finite state automaton to create tokens
- lexer generators
    - Alex for Haskell

Lexical Analyzers (Lexers)

translator tool : strings (programs) $\longrightarrow$ list of tokens (or error)

- simulate finite state automaton to create tokens
- lexer generators
  - Alex for Haskell
  - JLex for Java

Lexical Analyzers (Lexers)

translator tool : strings (programs) $\longrightarrow$ list of tokens (or error)

- simulate finite state automaton to create tokens
- lexer generators
  - Alex for Haskell
  - JLex for Java
  - Flex for C

Lexical Analyzers (Lexers)

translator tool : strings (programs) $\longrightarrow$ list of tokens (or error)

- simulate finite state automaton to create tokens
- lexer generators
  - Alex for Haskell
  - JLex for Java
  - Flex for C
  - *ocamllex* for OCaml

ocamllex

- generates lexers in compatible with OCaml programs

input	:	list of *lexing rules* in the form of *regular expressions* with corresponding *tokens*
output	:	a lexer as a DFA accepting the language generated by the input expressions

ocamllex

- generates lexers in compatible with OCaml programs

input	:	list of *lexing rules* in the form of *regular expressions* with corresponding *tokens*
output	:	a lexer as a DFA accepting the language generated by the input expressions

- recognizes patterns specified by the rules, associates the corresponding tokens

ocamllex (Specifications)

```
{ header }

rule entrypoint = parse
  | regexp { action }
  | ...
  | regexp { action }
  | ...

{ trailer }
```

ocamllex (Specifications)

```
{ header }

rule entrypoint = parse
  | regexp { action }
  | ...
  | regexp { action }
  | ...

{ trailer }
```

where

actions are OCaml expressions of the same type

ocamllex (Specifications)

```
{ header }

rule entrypoint = parse
  | regexp { action }
  | ...
  | regexp { action }
  | ...

{ trailer }
```

where

actions	are OCaml expressions of the same type	
header	define functions used in tokenization	(before tokenization)

ocamllex (Specifications)

```
{ header }

rule entrypoint = parse
  | regexp { action }
  | ...
  | regexp { action }
  | ...

{ trailer }
```

where

actions	are OCaml expressions of the same type	
header	define functions used in tokenization	(before tokenization)
trailer	define functions using tokenization	(after tokenization)

Example (ocamllex: Balanced Parentheses – Header)

```
(* lexer.mll *)
{
  open Lexing
  open Printf
  exception Bad_char of char

  type token =
    | BLANK  : token
    | LPAREN : token
    | RPAREN : token
    | EOL    : token
    | IDENT  : string → token
    | NUM    : int    → token

  let rec token2String (t: token): string =
    match t with
    | BLANK   → "BLANK"
    | LPAREN  → "LPAREN"
    | RPAREN  → "RPAREN"
    | EOL     → "EOL"
    | IDENT s → "IDENT=[" ^ s ^ "]"
    | NUM x   → "NUM=" ^ string_of_int x ^ "]"

  let printToken (t: token): unit = printf "%s␣" (token2String t)
}
```

Example (ocamllex: Balanced Parentheses – Rules)

```
rule tokenize = parse
  | ' '                                    { BLANK }
  | '['                                    { LPAREN }
  | ']'                                    { RPAREN }
  | [ α'-'z' 'A'-'Z' '_' ]+  as s          { IDENT s }
  | [ '0'-'9' ]+ as i                      { NUM (int_of_string i) }
  | eof                                    { EOL }
  | _ as c                                 { raise (Bad_char c) }
```

Example (ocamllex: Balanced Parentheses – Trailer)

```
{
  let rec getTokensFromBuffer (b: lexbuf): token list =
    let tkn = tokenize b in
    match tkn with
    | EOL → [EOL]
    | t   → t :: getTokensFromBuffer b

  let getTokensFromString (s: string): token list =
    getTokensFromBuffer (from_string s)
}
```

Example (ocamllex: Balanced Parentheses – Trailer)

```
{
  let rec getTokensFromBuffer (b: lexbuf): token list =
    let tkn = tokenize b in
    match tkn with
    | EOL → [EOL]
    | t   → t :: getTokensFromBuffer b

  let getTokensFromString (s: string): token list =
    getTokensFromBuffer (from_string s)
}
```

Remark (in OCaml "Lexing" library)

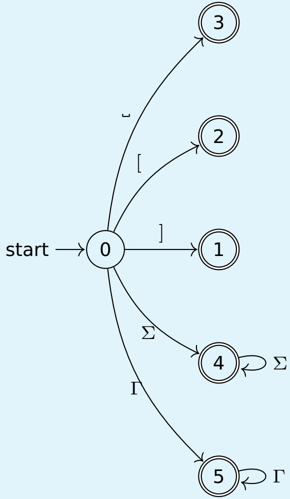
where

lexbuf	input stream that delivers characters one at a time
from_string: string → lexbuf	function converts the input string into a stream

Example (ocamllex: Balanced Parentheses – Main)

```
(* main.ml *)

open Printf
open Lexer

let rec tokenList2String (l: token list): string =
  match l with
  | []    → ""
  | x::xs → token2String x ^ "␣" ^ tokenList2String xs

let printTokenList (l: token list): unit =
  printf "%s\n" (tokenList2String l)

let main: unit =
  let tl = getTokensFromString "[][][][[2023][burak]]" in
  printTokenList tl;
```

ocamllex (DFA – tokenize)



where

1	RPAREN
2	LPAREN
3	BLANK
4	IDENT
5	NUM

Thanks! & Questions?