# CENG 2010 - Programming Language Concepts Week 3: Regular Expressions and Lexical Analysis

Burak Ekici

March 20, 2022

# **Table of Contents**

1 Languages

- 2 Compilers and Interpreter
- Pattern Matching and Regular Expression
- 4 Finite State Automator
- **5** Lexe

Languages 000

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

Languages

0000

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

#### Example

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

Languages

0000

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

#### Example

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

Languages

0000

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

#### Example

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

Languages

000

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

#### Example

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

Languages

000

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

#### Example

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

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

Languages

0000

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

#### Example

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

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

## Definitions (Operations on Languages)

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

# Definitions (Operations on Languages)

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

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

0000

## Definitions (Operations on Languages)

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

Concatenation  $\mathcal{L}_1\mathcal{L}_2$  :=  $\{xy \mid x \in \mathcal{L}_1 \land y \in \mathcal{L}_2\}$ Union  $\mathcal{L}_1 \cup \mathcal{L}_2$  :=  $\{x \mid x \in \mathcal{L}_1 \lor x \in \mathcal{L}_2\}$ 

0000

## Definitions (Operations on Languages)

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

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

0000

# Definitions (Operations on Languages)

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

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

Kleene star  $\Sigma^* = \mathcal{L} := \{x \mid x = \varepsilon \ v \ x \in \mathcal{L} \ v \ x \in \mathcal{LL} \ v \ x \in \mathcal{LLL} \ v \ \dots \}$ 

0000

# Definitions (Operations on Languages)

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

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

Kleene star  $\Sigma^* = \mathcal{L} := \{ x \mid x = \varepsilon \lor x \in \mathcal{L} \lor x \in \mathcal{LL} \lor x \in \mathcal{LLL} \lor \ldots \}$ 

Kleene plus  $\Sigma^+ := \Sigma^* - \{\varepsilon\}$ 

# Example (Operations on Languages)

let

Languages ○○○●

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

## Example (Operations on Languages)

let

Languages

000

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

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

## Example (Operations on Languages)

let

Languages

000

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

$$\mathcal{L}_1 = \{a, ab, c, d, \varepsilon\}$$

$$\mathcal{L}_2 = \{d\}$$

$$\mathcal{L}_3 := \mathcal{L}_1 \mathcal{L}_2$$

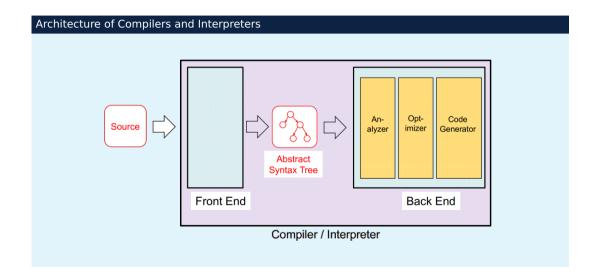
$$\mathcal{L}_4 := \mathcal{L}_1 \cup \mathcal{L}_2$$

- 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

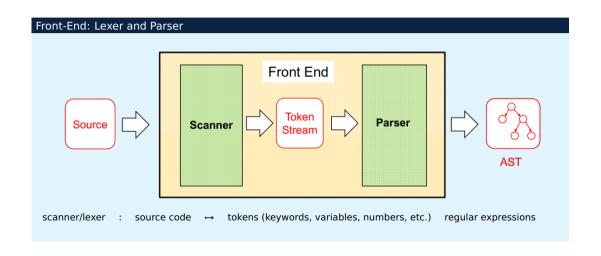
# **Table of Contents**

1 Language

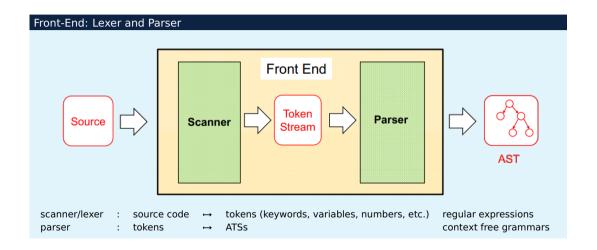
- 2 Compilers and Interpreters
- Pattern Matching and Regular Expression
- 4 Finite State Automator
- **5** Lexe



00.00



00.00



Languages 0000

• goal: map program texts into PTs/ASTs

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

- 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

- 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 {{...}} and parentheses (((...))), etc.

- 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  $\{\{...\}\}$  and parentheses (((...))), etc.
- regular expressions for tokenizing (scanning/lexing), and context free grammars for parsing tokens

# **Table of Contents**

1 Language

- 2 Compilers and Interpreter
- 3 Pattern Matching and Regular Expressions
- 4 Finite State Automator
- 5 Lexe

# Pattern matching is important for

• lexical analysis of programs

Languages 0000

- search engines (Google Code Search)
- scripting languages (Perl, Ruby)

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

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

#### Some Patterns

Languages

matches beginning of line

- 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

Finite State Automaton

matches character c

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

#### Some Patterns

C

Languages

matches beginning of line matches any character matches end of line [abc] matches a or b or c

Finite State Automaton

# 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
С	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

Languages

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

#### Example

Languages

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

Languages

matches beginning of line . matches any character matches end of line [abc] matches a or b or 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

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

## **Definitions**

Languages

lexical analysis of programs

• search engines (Google Code Search)

• scripting languages (Perl, Ruby)

DNA analysis

## Definitions

Languages

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

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

•

lexical analysis of programs

• search engines (Google Code Search)

• scripting languages (Perl, Ruby)

DNA analysis

## Definitions

Languages

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

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

•

lexical analysis of programs

• search engines (Google Code Search)

• scripting languages (Perl, Ruby)

DNA analysis

## Definitions

Languages

atomic pattern $lpha$	$L(\alpha)$
$\mathbf{a} \in \Sigma$	{a}
ε	$\{oldsymbol{arepsilon}\}$
Ø	Ø

lexical analysis of programs

• search engines (Google Code Search)

• scripting languages (Perl, Ruby)

DNA analysis

## Definitions

Languages

atomic pattern $lpha$	$L(\alpha)$
$\mathbf{a} \in \Sigma$	{a}
ε	$\{arepsilon\}$
Ø	Ø
#	$\Sigma$

- lexical analysis of programs
- scripting languages (Perl, Ruby)

- search engines (Google Code Search)
- DNA analysis

## Definitions

Languages

atomic pattern $lpha$	$L(\alpha)$
$\mathbf{a} \in \Sigma$	{a}
ε	{ε}
Ø	Ø
#	$\Sigma$
@	$\Sigma^*$

- lexical analysis of programs
- scripting languages (Perl, Ruby)

- search engines (Google Code Search)
- DNA analysis

## **Definitions**

Languages

	atomic pattern $\alpha$	L(a
	$\mathbf{a} \in \Sigma$	{a}
	ε	$\{oldsymbol{arepsilon}\}$
•	Ø	Ø
	#	$\Sigma$
	@	$\Sigma^*$

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

- lexical analysis of programs
- scripting languages (Perl, Ruby)

- search engines (Google Code Search)
- DNA analysis

## Definitions

Languages

	atomic pattern $\alpha$	$L(\alpha)$
	$\mathbf{a} \in \Sigma$	{a}
	ε	$\{\varepsilon\}$
•	Ø	Ø
	#	$\Sigma$
	@	$\Sigma^*$

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

- lexical analysis of programs
- scripting languages (Perl, Ruby)

- search engines (Google Code Search)
- DNA analysis

## Definitions

Languages

	atomic pattern $lpha$	$L(\alpha)$
	$\mathbf{a} \in \Sigma$	{a}
	ε	$\{oldsymbol{arepsilon}\}$
•	Ø	Ø
	#	$\Sigma$
	@	$\Sigma^*$

compound pattern $lpha$	$L(\alpha)$
$eta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
βγ	$L(\beta)L(\gamma)$

lexical analysis of programs

• search engines (Google Code Search)

• scripting languages (Perl, Ruby)

-t---!- --tt---- - / / --\

DNA analysis

## Definitions

Languages

	atomic pattern $\alpha$	$L(\alpha)$
	$\mathbf{a} \in \Sigma$	{a}
	ε	$\{oldsymbol{arepsilon}\}$
•	Ø	Ø
	#	$\Sigma$
	@	$\Sigma^*$

compound pattern $lpha$	$L(\alpha)$
$oldsymbol{eta}+oldsymbol{\gamma}$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$eta\gamma$	$L(\beta)L(\gamma)$
$oldsymbol{eta}^*$	$L(\beta)^*$

- lexical analysis of programs
- scripting languages (Perl, Ruby)

-t---!- --tt---- - / / --\

- search engines (Google Code Search)
- DNA analysis

## **Definitions**

Languages

	atomic pattern α	$L(\alpha)$	
	$\mathbf{a} \in \Sigma$	{a}	
	ε	{ε}	
•	Ø	Ø	
	#	$\Sigma$	
	@	$\Sigma^*$	

compound pattern $lpha$	$L(\alpha)$
$\beta + \gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$eta\gamma$	$L(\beta)L(\gamma)$
$oldsymbol{eta}^*$	$L(\beta)^*$
$oldsymbol{eta}^+$	$L(\beta)^+$

• lexical analysis of programs

search engines (Google Code Search)

scripting languages (Perl, Ruby)

DNA analysis

## Definitions

Languages

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

	atomic pattern $\alpha$	$L(\alpha)$
	$\mathbf{a} \in \Sigma$	{a}
	ε	$\{oldsymbol{arepsilon}\}$
•	Ø	Ø
	#	$\Sigma$
	@	$\Sigma^*$

-t---!- --tt---- - / / --\

compound pattern $lpha$	$L(\alpha)$
$oldsymbol{eta}+\gamma$	$L(\beta) \cup L(\gamma)$
$\beta \cap \gamma$	$L(\beta) \cap L(\gamma)$
$eta\gamma$	$L(\beta)L(\gamma)$
$oldsymbol{eta}^*$	$L(\beta)^*$
$oldsymbol{eta}^+$	$L(\beta)^+$
~ β	$\sim L(\beta) = \Sigma^* - L(\beta)$

lexical analysis of programs

• search engines (Google Code Search)

scripting languages (Perl, Ruby)

DNA analysis

## Definitions

Languages

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

	atomic pattern $lpha$	$L(\alpha)$
	$\mathbf{a} \in \Sigma$	{a}
	ε	$\{\varepsilon\}$
•	Ø	Ø
	#	$\Sigma$
	@	$\Sigma^*$

С	ompound pattern $lpha$	$L(\alpha)$
β	$r^2 + \gamma$	$L(\beta) \cup L(\gamma)$
β	$\cap \gamma$	$L(\beta) \cap L(\gamma)$
β	γ	$L(\beta)L(\gamma)$
β	*	$L(\beta)^*$
β	+	$L(\beta)^+$
~	β	$\sim L(\beta) = \Sigma^* - L(\beta)$

• string  $x \in \Sigma^*$  matches pattern  $\alpha$  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
@ <b>a</b> @ <b>b</b> @	strings containing a followed later by b

Example		
pattern	matched string	
@a@a@a@	strings containing at least 3 occurrences of a	
@ <b>a</b> @ <b>b</b> @	strings containing a followed later by b	
#∩ ~ a	single letters except a	

Example	
pattern	matched string
@ <b>a</b> @ <b>a</b> @ <b>a</b> @	strings containing at least 3 occurrences of a
@ <b>a</b> @ <b>b</b> @	strings containing a followed later by b
#∩ <b>~</b> a	single letters except a
(#∩ <b>~</b> a)*	strings without a

Languages 0000

Languages 0000

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

Languages 0000

Languages 0000

$$\alpha^+ \equiv \alpha \alpha^*$$

Languages 0000

$$\boldsymbol{\varepsilon}$$
  $\equiv$   $\sim (\#@) \equiv \boldsymbol{\mathcal{O}}^*$   $\equiv$   $\#^*$ 

$$\alpha^+ \equiv \alpha \alpha^*$$

$$\# \equiv a_1 \dots a_n$$

Languages 0000

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

Languages 0000

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

Languages 0000

which operators are redundant?

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

## Notation

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

## Definition

regular expressions are restricted patterns which use only

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

αβ

#### Definition

Languages

regular expressions are restricted patterns which use only

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

## Remark (Precedence)

Kleene closure \* > concatenation > union +

$$ab + c$$
 :=  $(ab) + c$  =  $\{ab, c\}$   
 $ab^*$  :=  $a(b^*)$  =  $\{a, ab, abb, ...\}$   
 $a + b^*$  :=  $a + (b^*)$  =  $\{a, \varepsilon, b, bb, bbb, ...\}$ 

#### Definition

Languages

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

#### Example

```
\begin{array}{lll} {\bf a} & & = \{a\} \\ {\bf a} + {\bf b} & := & \{a\} \cup \{b\} & = \{a,b\} \\ {\bf a}^* & := & \{\epsilon\} \cup \{a\} \cup \{aa\} \cup \ldots & = \{\epsilon,a,aa,aaa,\ldots\} \\ {\bf ab}^*({\bf c} + {\bf \epsilon}) & = \{a,ac,ab,abc,abb,abbc,\ldots\} \end{array}
```

#### Theorer

Languages

patterns, and regular expressions and regular languages are equivalent:

#### Theorem

Languages

patterns, and regular expressions and regular languages are equivalent:

for all  $A \subseteq \Sigma^*$  ① A is regular

 $\Leftrightarrow$  2  $A = L(\alpha)$  for some pattern  $\alpha$ 

 $\iff$   $\blacksquare$   $A = L(\alpha)$  for some regular expression  $\alpha$ 

# Implementing Regular Expressions

• by finite automaton – "machines" recognize regular languages

# Implementing Regular Expressions by finite automaton – "machines" recognize regular languages string FA

# **Table of Contents**

1 Language

- 2 Compilers and Interpreter
- Pattern Matching and Regular Expression
- 4 Finite State Automaton
- **5** Lexe

Languages 0000

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

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

Languages 0000

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

① Q : finite set of states

Languages

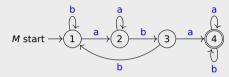
• deterministic finite automaton (DFA) is quintuple  $M = (Q, \Sigma, \delta, s, F)$  with

① Q : finite set of states

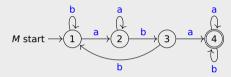
②  $\Sigma$ : input alphabet ③  $\delta$  :  $O \times \Sigma \rightarrow O$ : transition function

transition function

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

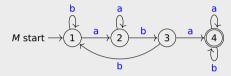


 $M = (Q, \Sigma, \delta, s, F)$ 



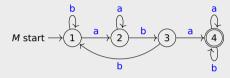
$$\bigcirc Q = \{1, 2, 3, 4\}$$

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



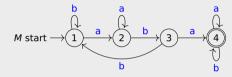
$$\Sigma = \{a, b\}$$

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



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

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



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

- 2 3
- 4

Languages

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

 $\bigcirc Q$ : finite set of states

②  $\Sigma$ : input alphabet  $\delta: O \times \Sigma \to O$ : transition function

**6**  $\delta: Q \times \Sigma \to Q:$  transition function start state

Languages

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

 $\bigcirc Q$ : finite set of states

②  $\Sigma$ : input alphabet  $\delta: Q \times \Sigma \to Q$ : transition function

 $4s \in Q$ : start state

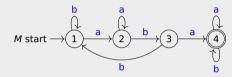
⑤  $F \subseteq Q$ : final (accept) states

Finite State Automaton

00000000000000000

## Example (DFAs → Regular Sets)

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

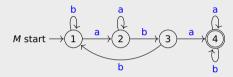


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$

- 2 3
- 4

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

Languages 0000



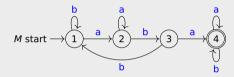
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$

**4** s = 1

- 4

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

Languages 0000



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

. 2 1

2 2 3

**4** s = 1

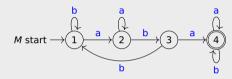
3 4

**6**  $F = \{4\}$ 

4 4 4

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

Languages 0000



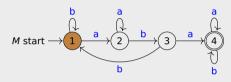
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$

2 3

**4** s = 1

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

Languages 0000

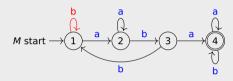


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

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

Languages 0000



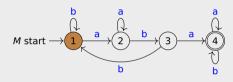
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$

- 2 2 3 4
- s = 1  $F = \{4\}$

4 4 4

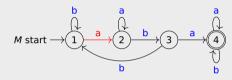
$$M = (Q, \Sigma, \delta, s, F)$$

Languages 0000



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- 2 3
- **4** s = 1

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

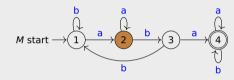


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$

- 2 3 **4** s = 1
- **6**  $F = \{4\}$

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

Languages 0000

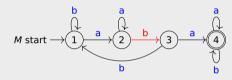


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

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

Languages 0000

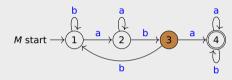


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

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

Languages 0000

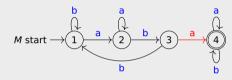


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

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

Languages 0000



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

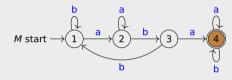
Finite State Automaton

00000000000000000

### Example (DFAs → Regular Sets)

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

Languages 0000

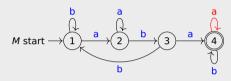


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

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

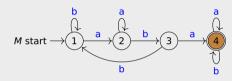
Languages 0000



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

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

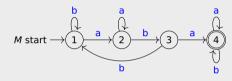


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- 2 3

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

Languages 0000



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

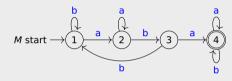
2 3

**6**  $F = \{4\}$ 

22/38

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

Languages 0000



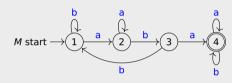
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1

2 3

**6**  $F = \{4\}$ 

22/38

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



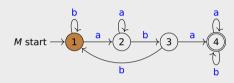
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- - 3

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



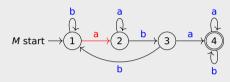
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- 3

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

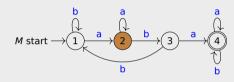


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3
- 4

- 3

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



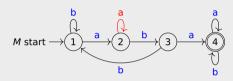
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- 3

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

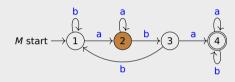


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3
- 4

- 3

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



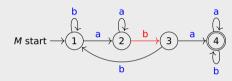
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- 3

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



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

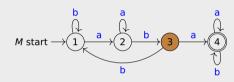
- 3

Finite State Automaton

00000000000000000

#### Example (DFAs → Regular Sets)

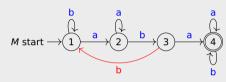
 $M = (Q, \Sigma, \delta, s, F)$ 



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3
- 4

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



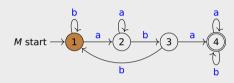
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- 3

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



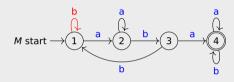
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- 3

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



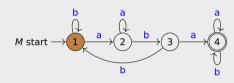
- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- 3

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

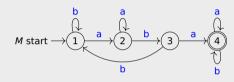


- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3
- 4

- 3

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



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3
- 4

- 3

#### **Definitions**

Languages

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

① Q : finite set of states

 $\bigcirc$   $\Sigma$ : input alphabet  $\bigcirc$   $\delta: O \times \Sigma \rightarrow O$ : transition function

⑤  $F \subseteq Q$ : final (accept) states

•  $\hat{\delta}$ :  $Q \times \Sigma^* \to Q$  is inductively defined by

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

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

$$\delta(\widehat{\delta}(q_0, abbaa), b)$$
  
 $\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$ 

first recursive call second recursive call

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

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

 $\begin{array}{l} \delta(\widehat{\delta}(q_0,abbaa),b) \\ \delta(\delta(\widehat{\delta}(q_0,abba),a),b) \\ \delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b) \end{array}$ 

first recursive call second recursive call third recursive call

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b)\\ &\delta(\delta(\widehat{\delta}(q_0,abba),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b), \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b) \\ &\delta(\delta(\widehat{\delta}(q_0,abba),a),b) \\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b) \\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b) \\ &\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b) \\ &\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,a),b),b),a),a),b) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b) \\ &\delta(\delta(\widehat{\delta}(q_0,abba),a),b) \\ &\delta(\delta(\delta\widehat{\delta}(q_0,abb),a),a),b) \\ &\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b) \\ &\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b) \\ &\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,a),b),b),a),a),b)) \\ &\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,\epsilon),a),b),b),a),a),b)) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b) \\ &\delta(\delta(\widehat{\delta}(q_0,abba),a),b) \\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b) \\ &\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b) \\ &\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b) \\ &\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,a),b),b),a),a),b) \\ &\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,\epsilon),a),b),b),a),a),b) \\ &\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,a),b),b),a),a),b) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b)\\ &\delta(\delta(\widehat{\delta}(q_0,abba),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\widehat{\delta}(\widehat{d}(q_0,ab),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\widehat{\delta}(\widehat{d}(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,\epsilon),a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\delta(q_0,\epsilon),a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(q_0,a),b),b),a),a),b) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

assuming  $\delta(q_0,a)=q_1$ 

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b)\\ &\delta(\delta(\widehat{\delta}(q_0,abba),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\widehat{\delta}(\widehat{\delta}(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,\epsilon),a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\delta(q_0,\epsilon),a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(q_0,b),a),a),b) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

assuming  $\delta(q_0,a)=q_1$  assuming  $\delta(q_1,b)=q_2$ 

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b)\\ &\delta(\delta(\widehat{\delta}(q_0,abba),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,e),a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\delta(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(q_0,a),b),a),a),b)\\ &\delta(\delta(\delta(\delta(q_0,a),a),a),b)\\ &\delta(\delta(\delta(\delta(q_0,a),a),a),b)\\ &\delta(\delta(\delta(q_0,a),a),a),b)\\ &\delta(\delta(\delta(q_0,a),a),a),b) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

assuming  $\delta(q_0,a)=q_1$ assuming  $\delta(q_1,b)=q_2$ assuming  $\delta(q_2,b)=q_3$ 

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

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

$$\begin{split} \delta(\widehat{\delta}(q_0,abbaa),b) \\ \delta(\delta(\widehat{\delta}(q_0,abba),a),b) \\ \delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b) \\ \delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b) \\ \delta(\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b) \\ \delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,a),b),b),a),a),b) \\ \delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,\epsilon),a),b),b),a),a),b) \\ \delta(\delta(\delta(\delta(\delta(\delta(q_0,\epsilon),a),b),b),a),a),b) \\ \delta(\delta(\delta(\delta(\delta(q_1,b),b),a),a),b) \\ \delta(\delta(\delta(\delta(\delta(q_1,b),b),a),a),b) \\ \delta(\delta(\delta(\delta(q_3,a),a),b) \\ \delta(\delta(\delta(q_4,a),b) \\ \delta(\delta(q_4,a),b) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

assuming  $\delta(q_0,a)=q_1$  assuming  $\delta(q_1,b)=q_2$  assuming  $\delta(q_2,b)=q_3$  assuming  $\delta(q_3,a)=q_4$ 

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

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

$$\begin{split} &\delta(\widehat{\delta}(q_0,abbaa),b)\\ &\delta(\widehat{\delta}(q_0,abbaa),a),b)\\ &\delta(\delta(\delta(q_0,abba),a),b)\\ &\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b)\\ &\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,ab),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\delta(\widehat{\delta}(q_0,\epsilon),a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(\delta(q_0,\epsilon),a),b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(q_1,b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(\delta(q_1,b),b),a),a),b)\\ &\delta(\delta(\delta(\delta(q_2,b),a),a),b)\\ &\delta(\delta(\delta(q_4,a),b)\\ &\delta(q_5,b) \end{split}$$

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

assuming  $\delta(q_0,a)=q_1$ assuming  $\delta(q_1,b)=q_2$ assuming  $\delta(q_2,b)=q_3$ assuming  $\delta(q_3,a)=q_4$ assuming  $\delta(q_4,a)=q_5$ 

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

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

 $\delta(\widehat{\delta}(a_0, abbaa), b)$  $\delta(\delta(\widehat{\delta}(q_0, abba), a), b)$  $\delta(\delta(\delta(\widehat{\delta}(q_0,abb),a),a),b)$  $\delta(\delta(\delta(\delta(\delta(g_0,ab),b),a),a),b)$  $\delta(\delta(\delta(\delta(\delta(\delta(g_0,a),b),b),a),a),b)$  $\delta(\delta(\delta(\delta(\delta(\delta(\delta(\delta(a_0, \varepsilon), a), b), b), a), a), b))$  $\delta(\delta(\delta(\delta(\delta(\delta(q_0,a),b),b),a),a),b)$  $\delta(\delta(\delta(\delta(\delta(q_1,b),b),a),a),b)$  $\delta(\delta(\delta(\delta(q_2,b),a),a),b)$  $\delta(\delta(\delta(q_3,a),a),b)$  $\delta(\delta(q_4,a),b)$  $\delta(q_5, b)$ 96

first recursive call second recursive call third recursive call fourth recursive call fifth recursive call sixth recursive call

assuming  $\delta(q_0,a)=q_1$ assuming  $\delta(q_1,b)=q_2$ assuming  $\delta(q_2,b)=q_3$ assuming  $\delta(q_3,a)=q_4$ assuming  $\delta(q_4,a)=q_5$ assuming  $\delta(q_5,b)=q_6$ 

#### **Definitions**

Languages

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

 $\bigcirc Q$ : finite set of states

 $\bigcirc$   $\Sigma$ : input alphabet  $\bigcirc$   $\delta: O \times \Sigma \rightarrow O$ : transition function

 $\triangle s \in O$ : start state

⑤  $F \subseteq Q$ : final (accept) states

•  $\hat{\delta}: Q \times \Sigma^* \to Q$  is inductively defined by

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

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

Finite State Automaton

#### **Definitions**

Languages

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

① Q: finite set of states

 $\bigcirc \Sigma$ : input alphabet

**6**  $\delta: Q \times \Sigma \to Q:$  transition function  $A \in O:$  start state

⑤ F ⊆ Q: final (accept) states

•  $\hat{\delta}: O \times \Sigma^* \to O$  is inductively defined by

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

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

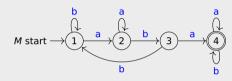
• string  $x \in \Sigma^*$  is rejected by M if  $\widehat{\delta}(s, x) \notin F$ 

Finite State Automaton

00000000000000000

#### Example (DFAs → Regular Sets)

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



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

- - 2 3

  - 4

- $\in L(M)$
- - $\notin L(M)$
- 2 3

#### **Definitions**

Languages

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

① Q : finite set of states

 $\bigcirc$   $\Sigma$ : input alphabet  $\bigcirc$   $\delta: O \times \Sigma \rightarrow O$ : transition function

**6**  $\delta: Q \times \Sigma \rightarrow Q:$  transition func  $A : S \in O:$  start state

⑤  $F \subseteq Q$ : final (accept) states

•  $\hat{\delta}$ :  $Q \times \Sigma^* \to Q$  is inductively defined by

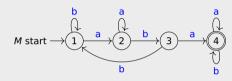
$$\widehat{\delta}(q, \varepsilon) := q$$
  $\widehat{\delta}(q, xa) := \delta(\widehat{\delta}(q, x), a)$ 

• 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  $L(M) := \{x \mid \widehat{\delta}(s, x) \in F\}$ 

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



- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **6**  $F = \{4\}$

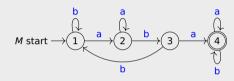
- - 2 3

  - 4

- $\in L(M)$ а
- $\notin L(M)$
- 3
  - $L(M) := \{x \mid$

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

Languages



Pattern Matching and Regular Expressions

- $\bigcirc Q = \{1, 2, 3, 4\}$
- $\Sigma = \{a, b\}$
- **4** s = 1
- **⑤**  $F = \{4\}$

- - 2 3

  - 4

- $\in L(M)$
- - $\notin L(M)$
- 3

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

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

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

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

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

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

- **(1)** Q =
- $\Sigma = \{a, b\}$

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

- **1** 0 =
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\mathbf{A} s =$

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

- **0** 0 =
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\triangle s =$
- $\bigcirc$  F =

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

 $M \text{ start} \rightarrow 1$ 

- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\triangle s =$
- $\bigcirc$  F =

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

 $M \text{ start} \longrightarrow 1$ 

2

$$\bigcirc Q =$$

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

$$\triangle s =$$

$$\bigcirc$$
  $F =$ 

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



- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\triangle s =$
- F =

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



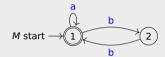
- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\triangle s =$
- F =

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



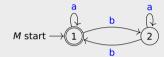
- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\triangle s =$
- F =

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



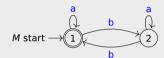
- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\triangle s =$
- F =

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



- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bullet$   $\delta: Q \times \Sigma \rightarrow Q$
- $\triangle s =$
- F =

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



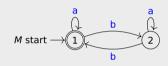
$$\bigcirc Q = \{1, 2\}$$

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

**4** 
$$s = 1$$

**⑤** 
$$F = \{1\}$$

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



a

b

b

$$\bigcirc Q = \{1, 2\}$$

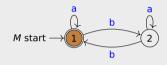
$$\Sigma = \{a, b\}$$

**6** 
$$F = \{1\}$$

b

#### Example (Regular Language → DFA)

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



а

b

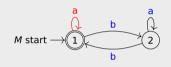
$$\bigcirc Q = \{1, 2\}$$

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

$$\begin{array}{c|cccc}
\hline
\bullet & \Sigma = \{a,b\} \\
\hline
\bullet & \delta : Q \times \Sigma \to Q \\
\hline
\bullet & s = 1 \\
\hline
\bullet & s = 2
\end{array}$$

**6** 
$$F = \{1\}$$

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



b

b

$$\bigcirc Q = \{1, 2\}$$

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

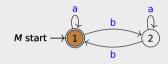
$$\begin{array}{c|cccc}
\bullet & \bullet & \bullet \\
\bullet & \delta : Q \times \Sigma \to Q & & 1 & 1 \\
\bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & & & & \\
\bullet & \bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & \bullet & & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet$$

**4** 
$$s = 1$$

**6** 
$$F = \{1\}$$

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

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



a

b

$$\bigcirc Q = \{1, 2\}$$

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

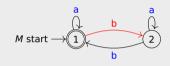
**4** 
$$s = 1$$

**⑤** 
$$F = \{1\}$$



b

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



a

b

b

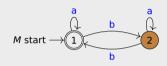
$$\bigcirc Q = \{1, 2\}$$

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

**4** 
$$s = 1$$

**⑤** 
$$F = \{1\}$$

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



$$\bigcirc Q = \{1, 2\}$$

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

**4** 
$$s = 1$$

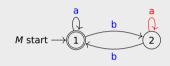
**⑤** 
$$F = \{1\}$$



a

b

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



a

b

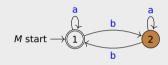
$$\bigcirc Q = \{1, 2\}$$

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

b

**⑤** 
$$F = \{1\}$$

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



a

b

b

$$\bigcirc Q = \{1, 2\}$$

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

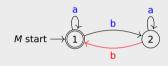
$$\delta: Q \times \Sigma \to 0$$

$$\delta: Q \times \Sigma \to 0$$

⑤ 
$$F = \{1\}$$



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



a

b

b

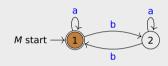
$$\bigcirc Q = \{1, 2\}$$

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

**⑤** 
$$F = \{1\}$$

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

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



a

$$\bigcirc Q = \{1, 2\}$$

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

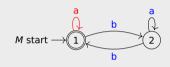
$$\begin{array}{ccc} \text{ } & \delta: Q \times \Sigma \to Q & \\ \text{ } & 1 & \\ \text{ } & 2 & \\ \end{array}$$

**6** 
$$F = \{1\}$$

b

b

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



a

b

$$\bigcirc Q = \{1, 2\}$$

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

$$0: Q \times$$

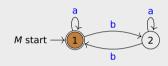
**6** 
$$F = \{1\}$$



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

b

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



a

b

b

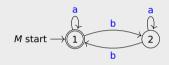
$$\bigcirc Q = \{1, 2\}$$

② 
$$\Sigma = \{a, b\}$$

**6** 
$$F = \{1\}$$

$$s = 1$$
  
 $F = \{1\}$ 

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



a

b

b

$$\bigcirc Q = \{1, 2\}$$

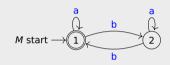
$$\Sigma = \{a, b\}$$

**4** 
$$s = 1$$

**6** 
$$F = \{1\}$$

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

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



a

b

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

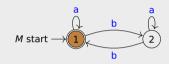
**4** 
$$s = 1$$

⑤ 
$$F = \{1\}$$

b

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

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



а

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

$$F = \{1\}$$

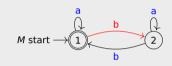


b

b

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

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



а

b

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

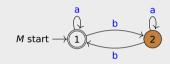
**6** 
$$F = \{1\}$$



b

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

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



а

b

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

⑤ 
$$F = \{1\}$$



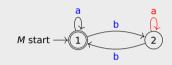
b

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

 $\in L(M)$ 

#### Example (Regular Language → DFA)

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



$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

⑤ 
$$F = \{1\}$$



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

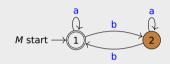
b

Finite State Automaton

00000000000000000

#### Example (Regular Language → DFA)

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



а

b

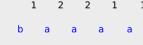
$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

**6** 
$$F = \{1\}$$

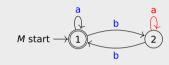


b

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

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

Languages



а

b

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

**6** 
$$F = \{1\}$$

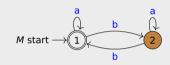


b

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

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

Languages



а

b

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

$$F = \{1\}$$



b

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

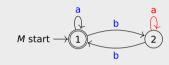
 $\in L(M)$ 

Finite State Automaton

00000000000000000

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

Languages



$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

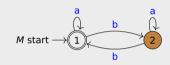
**4** 
$$s = 1$$

**6** 
$$F = \{1\}$$



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

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



b

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

⑤ 
$$F = \{1\}$$



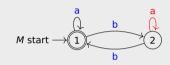
а



b

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

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



а

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

⑤ 
$$F = \{1\}$$







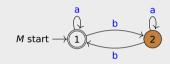
b



b

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

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



а

b

$$\bigcirc Q = \{1, 2\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

**6** 
$$F = \{1\}$$

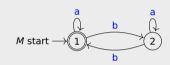


b

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

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

Languages



а

$$\bigcirc Q = \{1, 2\}$$

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

**4** 
$$s = 1$$

**6** 
$$F = \{1\}$$



b

b

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

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

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

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

$$\bigcirc O =$$

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

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

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

$$\delta: Q \times \Sigma \to Q$$

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

- **(1)** Q =
- $\Sigma = \{a, b\}$
- $\bigcirc s =$

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

- **0** 0 =
- $\Sigma = \{a, b\}$
- $\mathbf{a} s =$
- $\bigcirc F =$

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

$$M \text{ start} \longrightarrow 1$$

- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\mathbf{a} s =$
- $\bigcirc$  F =

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

$$M \operatorname{start} \longrightarrow 1$$
 2

$$\bigcirc Q =$$

$$\mathbf{a} s =$$

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

$$M \text{ start} \longrightarrow 1$$
  $\longrightarrow 2$ 

$$\bigcirc Q =$$

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

$$\mathbf{0}$$
  $s =$ 

$$\bigcirc$$
  $F =$ 

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

$$M \text{ start} \longrightarrow 1 \longrightarrow 2$$
 3

$$\bigcirc Q =$$

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

$$\bigcirc s =$$

$$\bigcirc F =$$

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



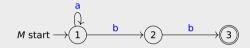
- **(1)** *Q* =
- $\Sigma = \{a, b\}$
- $\triangle s =$
- F =

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



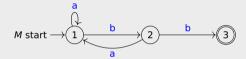
- **(1)** *Q* =
- $\Sigma = \{a, b\}$
- $\triangle s =$
- **⑤** *F* =

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



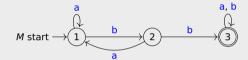
- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bigcirc s =$
- **⑤** *F* =

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



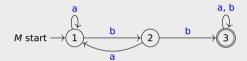
- $\bigcirc Q =$
- $\Sigma = \{a, b\}$
- $\bigcirc s =$
- F =

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



- **(1)** *Q* =
- $\Sigma = \{a, b\}$
- $\mathbf{0}$  s =
- F =

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



$$\bigcirc O = \{1, 2, 3\}$$

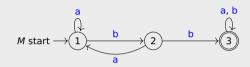
$$\Sigma = \{a, b\}$$

**4** 
$$s = 1$$

$$5 = 1$$
  
 $5 = {3}$ 

	_	
•	1	- 3
.	_	_

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



$$\bigcirc O = \{1, 2, 3\}$$

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

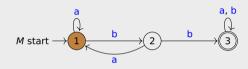
$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

$$5 = \{3\}$$

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

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



$$\bigcirc O = \{1, 2, 3\}$$

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

$$\delta: Q \times \Sigma \to Q$$

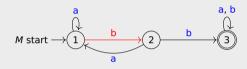
**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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

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

Languages



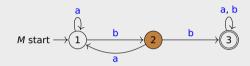
$$0 = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

**5** 
$$F = \{3\}$$

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



$$\bigcirc O = \{1, 2, 3\}$$

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

$$\delta: Q \times \Sigma \to Q$$

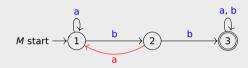
**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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

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

Languages



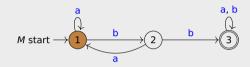
$$0 = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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



$$\bigcirc O = \{1, 2, 3\}$$

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

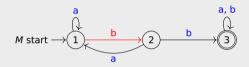
$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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

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



$$0 = \{1, 2, 3\}$$

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

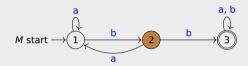
$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

$$5 = \{3\}$$

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

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



$$0 = \{1, 2, 3\}$$

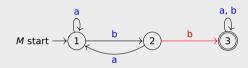
$$\Sigma = \{a, b\}$$

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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

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



$$0 = \{1, 2, 3\}$$

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

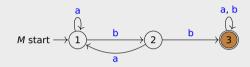
$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

$$5 = \{3\}$$

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

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



$$0 = \{1, 2, 3\}$$

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

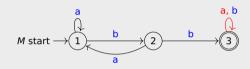
$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

$$F = \{3\}$$

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

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



$$0 = \{1, 2, 3\}$$

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

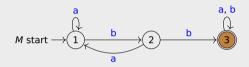
$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

$$5 = 1$$

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

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



$$\bigcirc O = \{1, 2, 3\}$$

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

$$\delta: Q \times \Sigma \to Q$$

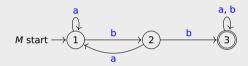
**4** 
$$s = 1$$

$$5 = \{3\}$$

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

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

Languages



$$\bigcirc O = \{1, 2, 3\}$$

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

$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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

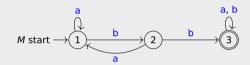
 $\in L(M)$ 

 $\in L(M)$ 

#### Example (Regular Language → DFA)

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

Languages



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

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

**4** 
$$s = 1$$

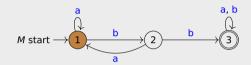
**6** 
$$F = \{3\}$$

δ	а	b
1	1	2
2	1	3
3	3	3



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

Languages



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

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

$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

δ	a	b
1	1	2
2	1	3



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

 $\in L(M)$ 

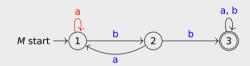
 $\in L(M)$ 

2

#### Example (Regular Language → DFA)

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

Languages



Pattern Matching and Regular Expressions

$$\bigcirc O = \{1, 2, 3\}$$

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

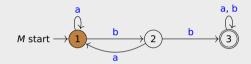
**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$



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

Languages



$$\bigcirc O = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

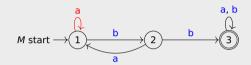
**6** 
$$F = \{3\}$$

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

 $\in L(M)$ 

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

Languages



$$\bigcirc O = \{1, 2, 3\}$$

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

$$\bullet$$
  $\delta: Q \times \Sigma \rightarrow Q$ 

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

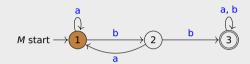
3

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

 $\in L(M)$ 

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

Languages



$$\bigcirc O = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

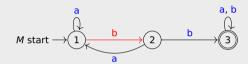


 $L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$ 

 $\in L(M)$ 

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

Languages



$$\bigcirc O = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

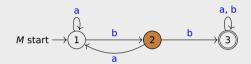
**6** 
$$F = \{3\}$$

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

 $\in L(M)$ 

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

Languages



Pattern Matching and Regular Expressions

$$0 = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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

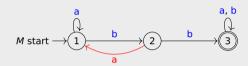
 $\in L(M)$ 

2

#### Example (Regular Language → DFA)

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

Languages



Pattern Matching and Regular Expressions

$$\bigcirc O = \{1, 2, 3\}$$

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

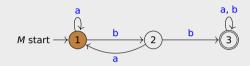
$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

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

Languages



$$\bigcirc O = \{1, 2, 3\}$$

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

$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

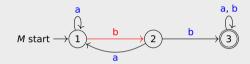
3

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

 $\in L(M)$ 

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

Languages

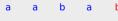


$$\bigcirc O = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

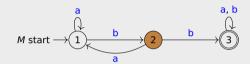


 $L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$ 

 $\in L(M)$ 

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

Languages



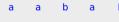
$$\bigcirc O = \{1, 2, 3\}$$

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

**4** 
$$s = 1$$

$$F = \{3\}$$

3

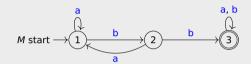


 $L(M) := \{x \mid x \text{ contains } bb \text{ as substring } \}$ 

 $\in L(M)$ 

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

Languages



$$0 = \{1, 2, 3\}$$

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

$$\delta: Q \times \Sigma \to Q$$

**4** 
$$s = 1$$

**6** 
$$F = \{3\}$$

δ	а	b
1	1	2
2	1	3



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

1	2	1	2	3	3

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

2

# Definitions

Languages 0000

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

Languages 0000

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

Languages 0000

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

① Q: ② Σ: finite set of states input alphabet

Languages 0000

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

① Q: finite set of states ②  $\Sigma$ : input alphabet

**⑤**  $\Delta: Q \times \Sigma \rightarrow 2^Q:$  transition function

Languages 0000

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

① Q: finite set of states  $\Sigma$ : input alphabet

 $\triangle : Q \times \Sigma \rightarrow 2^Q :$ transition function  $49.5 \subseteq Q$ :

Languages

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

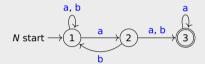
① Q: finite set of states  $\Sigma$ : input alphabet

 $\triangle : Q \times \Sigma \rightarrow 2^Q :$ transition function

**4 5** ⊆ *Q* : set of start states  $\bigcirc$   $F \subseteq Q$ : final (accept) states

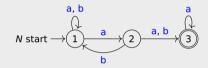
### Example

Languages 0000



### Exampl

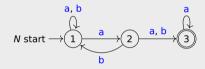
Languages 0000



① 
$$Q = \{1, 2, 3\}$$

### Example

Languages 0000

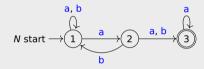


① 
$$Q = \{1, 2, 3\}$$

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

### Example

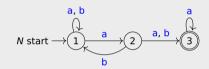
Languages 0000



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

### Exampl

Languages 0000



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

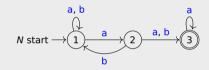
$\Delta$	a	b
1	{1,2}	{1}
2	{3}	{1,3}
3	{3}	Ø

Finite State Automaton

000000000000000000

### Example

Languages 0000



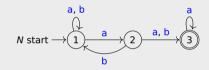
① 
$$Q = \{1, 2, 3\}$$

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

$$\P$$
  $S = \{1\}$ 

$\Delta$	a	b
1	{1,2}	{1}
2	{3}	{1,3}
3	{3}	Ø

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



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

$$\P$$
  $S = \{1\}$ 

**6** 
$$F = \{3\}$$

$$\begin{array}{c|cccc} \Delta & {\sf a} & {\sf b} \\ \hline 1 & \{1,2\} & \{1\} \\ 2 & \{3\} & \{1,3\} \\ 3 & \{3\} & \varnothing \\ \end{array}$$

Ø

{3}

Languages

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

1 Q: finite set of states Q: input alphabet

**S**  $\Delta: O \times \Sigma \rightarrow 2^{Q}$ : transition function

 $4 \le Q$ : set of start states

⑤  $F \subseteq Q$ : final (accept) states

•  $\widehat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$  is inductively defined by

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

Languages

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

**1** Q: finite set of states Q: input alphabet

**⑤**  $\Delta: O \times \Sigma \rightarrow 2^{Q}:$  transition function

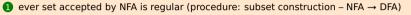
 $\P$   $S \subseteq Q$ : set of start states

⑤  $F \subseteq Q$ : final (accept) states

•  $\widehat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$  is inductively defined by

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

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



- $oldsymbol{0}$  ever set accepted by NFA is regular (procedure: subset construction NFA ightarrow DFA)
- 2 every DFA M minimizes into a DFA M' such that L(M) = L(M') (unless M is minimal)

- $oldsymbol{0}$  ever set accepted by NFA is regular (procedure: subset construction NFA ightarrow DFA)
- 2 every DFA M minimizes into a DFA M' such that L(M) = L(M') (unless M is minimal)

Languages

- 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)

#### Theoren

regular languages are closed under

union

Languages

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

#### Theoren

- union
- intersection

Languages

- 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)

#### Theoren

- union
- intersection
- complementation

Languages

- 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)

### Theoren

- union
- intersection
- complementation
- concatenation

#### Theorem

Languages

- 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

- union
- intersection
- complementation
- concatenation
- asterate

Languages

- 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

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

# **Table of Contents**

1 Language

- 2 Compilers and Interpreter
- Pattern Matching and Regular Expression
- 4 Finite State Automator
- 5 Lexer

Languages

translator tool : strings (programs) → list of tokens (or error)

• simulate finite state automaton to create tokens

Languages

- simulate finite state automaton to create tokens
- lexer generators

Languages

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

Languages

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

Languages

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

Languages

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

### ocamllex

Languages

• 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

Languages

• 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
              define functions used in tokenization
     header
                                                       (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"
     FOL → "FOL"
     IDENT s \rightarrow "IDENT=[" ^ s ^ "]"
     NUM x \rightarrow "NUM=" ^ string of int x ^ "]"
  let printToken (t: token): unit = printf "%s." (token2String t)
```

Finite State Automaton

```
Example (ocamllex: Balanced Parentheses – Rules)
```

```
rule tokenize = parse
                                          BLANK }
                                          LPAREN }
                                          RPAREN }
                                          IDENT s }
     '0'-'9' ]+ as i
                                          NUM (int of string i) }
                                          EOL }
   eof
  _ 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 \rightarrow t :: qetTokensFromBuffer b
let getTokensFromString (s: string): token list =
 getTokensFromBuffer (from string s)
```

# Remark (in OCaml "Lexing" library)

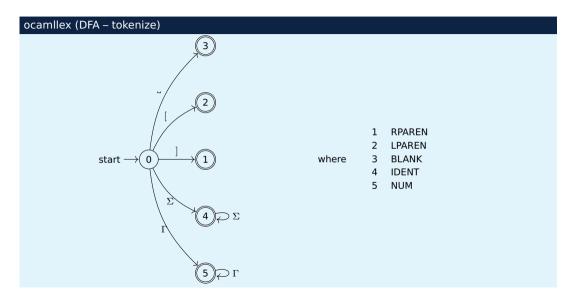
#### where

Languages

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 (I: token list): string =
 match I with
  | x::xs → token2String x ^ "_" ^ tokenList2String xs
let printTokenList (I: token list): unit =
  printf "%s\n" (tokenList2String I)
let main: unit =
  let tl = getTokensFromString "[][][[[2023][burak]]" in
  printTokenList tl:
```



Thanks! & Questions?