# CM0081 Automata and Formal Languages
# § 3.1 Regular Expressions

Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2024-1

# Preliminaries

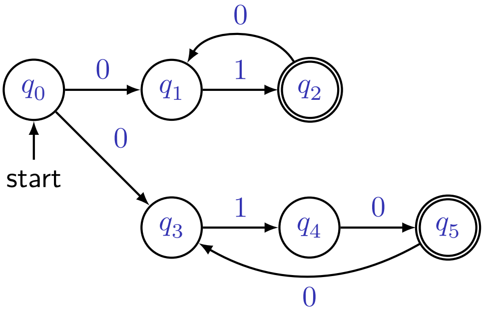
## Conventions

- The number and page numbers assigned to chapters, examples, exercises, figures, quotes, sections and theorems on these slides correspond to the numbers assigned in the textbook [Hopcroft, Motwani and Ullman 2007].
- The natural numbers include the zero, that is, $\mathbb{N} = \{0, 1, 2, \dots\}$.
- The power set of a set $A$, that is, the set of its subsets, is denoted by $\mathcal{P}\, A$.

# Introduction: Description of regular languages

$$(\mathbf{01})(\mathbf{01})^* + (\mathbf{010})(\mathbf{010})^*$$

Algebraic description




Machine-like description

# Introduction: Regular Expressions

## Features

- Algebraic description of regular languages

# Introduction: Regular Expressions

## Features

- Algebraic description of regular languages
- Declarative ('user-friendly') way to express the strings that belong to the language

# Introduction: Regular Expressions

## Features

- Algebraic description of regular languages
- Declarative (‘user-friendly’) way to express the strings that belong to the language

## Uses

- Search commands (e.g. GREP)

# Introduction: Regular Expressions

## Features

- Algebraic description of regular languages
- Declarative ('user-friendly') way to express the strings that belong to the language

## Uses

- Search commands (e.g. GREP)
- Lexical-analyzer generators (e.g. LEX and ALEX)

# Introduction: Regular Expressions

## Features

- Algebraic description of regular languages
- Declarative (‘user-friendly’) way to express the strings that belong to the language

## Uses

- Search commands (e.g. GREP)
- Lexical-analyzer generators (e.g. LEX and ALEX)
- Domain specific languages (DSLs)

# Operations on Languages

## Notation

The power set of a set $A$ is denoted $\mathcal{P}\, A$.

# Operations on Languages

## Definition

Let $L$, $L_1$ and $L_2$ be languages on an alphabet $\Sigma$.

(i) **Union** of languages:

$$\cup : \mathcal{P}\,\Sigma^* \times \mathcal{P}\,\Sigma^* \to \mathcal{P}\,\Sigma^*$$
$$L_1 \cup L_2 := \{\, x \mid x \in L_1 \text{ or } x \in L_2 \,\}.$$

(ii) **Concatenation** of languages:

$$\cdot : \mathcal{P}\,\Sigma^* \times \mathcal{P}\,\Sigma^* \to \mathcal{P}\,\Sigma^*$$
$$L_1 \cdot L_2 := \{\, x \cdot y \mid x \in L_1 \text{ and } y \in L_2 \,\}.$$

(iii) **Powers** of a language:

$$(-)^{(-)} : \mathcal{P}\,\Sigma^* \times \mathbb{N} \to \mathcal{P}\,\Sigma^*$$
$$L^0 := \{\varepsilon\},$$
$$L^{n+1} := L \cdot L^n.$$

(iv) **Kleene closure** of a language:

$$(-)^* : \mathcal{P}\,\Sigma^* \to \mathcal{P}\,\Sigma^*$$
$$L^* := \bigcup_{n \geq 0} L^n.$$

# Operations on Languages

## Examples

- If $L = \{0, 1\}$, then $L^*$ consists of all strings of $0$'s and $1$'s and the empty word.

# Operations on Languages

## Examples

- If $L = \{0, 1\}$, then $L^*$ consists of all strings of $0$'s and $1$'s and the empty word.
- If $L = \{\, 0^n \mid n \geq 1 \,\}$, then $L^* = L \cup \{\varepsilon\}$.

# Operations on Languages

## Examples

- If $L = \{0, 1\}$, then $L^*$ consists of all strings of $0$'s and $1$'s and the empty word.
- If $L = \{\, 0^n \mid n \geq 1 \,\}$, then $L^* = L \cup \{\varepsilon\}$.
- If $L = \{0, 11\}$, then $L^*$ consists of the empty word and those strings of $0$'s and $1$'s such that the $1$'s come in pairs.

# Operations on Languages

## Examples

- If $L = \{0, 1\}$, then $L^*$ consists of all strings of $0$'s and $1$'s and the empty word.
- If $L = \{\, 0^n \mid n \geq 1 \,\}$, then $L^* = L \cup \{\varepsilon\}$.
- If $L = \{0, 11\}$, then $L^*$ consists of the empty word and those strings of $0$'s and $1$'s such that the $1$'s come in pairs.
- Powers on $\emptyset$

$$
\begin{aligned}
\emptyset^0 &= \{\varepsilon\}, \\
\emptyset^i &= \emptyset, \quad \text{for } i \geq 1, \\
\emptyset^* &= \{\varepsilon\}.
\end{aligned}
$$

# What the Regular Expressions Are

## Definition

Let $\Sigma$ be an alphabet. The **regular expressions** (regex's) on $\Sigma$ are inductively defined by:

# What the Regular Expressions Are

## Definition

Let $\Sigma$ be an alphabet. The **regular expressions** (regex's) on $\Sigma$ are inductively defined by:

- Basis step
  - (i) $\varepsilon$ is a regex,
  - (ii) $\emptyset$ is regex and
  - (iii) If $a \in \Sigma$ then $\boldsymbol{a}$ is a regex.

# What the Regular Expressions Are

## Definition

Let $\Sigma$ be an alphabet. The **regular expressions** (regex's) on $\Sigma$ are inductively defined by:

- Basis step
  (i) $\varepsilon$ is a regex,
  (ii) $\emptyset$ is regex and
  (iii) If $a \in \Sigma$ then $\boldsymbol{a}$ is a regex.
- Inductive step
  If $E$ and $F$ are regex's then
  (i) $E + F$ is a regex,
  (ii) $E \cdot F$ is a regex,
  (iii) $E^*$ is a regex and
  (iv) $(E)$ is a regex.

# Precedence of Operators

## Order of precedence and associative

Precedence from highest to lowest: $()$, $^*$, $\cdot$ and $+$.

Associative: The operators $\cdot$ and $+$ are left-associative.

# Precedence of Operators

## Order of precedence and associative

Precedence from highest to lowest: $()$, $^*$, $\cdot$ and $+$.

Associative: The operators $\cdot$ and $+$ are left-associative.

### Example

$$\begin{aligned} \mathbf{01}^* + \mathbf{1} &= (\mathbf{0}(\mathbf{1}^*)) + \mathbf{1} \\ &\neq (\mathbf{01})^* + \mathbf{1} \\ &\neq \mathbf{0}(\mathbf{1}^* + \mathbf{1}) \end{aligned}$$

# Languages Denoted by Regular Expressions

### Definition

Let $E$ be a regular expression. The **language denoted** by $E$, denoted by $\mathrm{L}(E)$, is inductively defined by:

# Languages Denoted by Regular Expressions

## Definition

Let $E$ be a regular expression. The **language denoted** by $E$, denoted by $\mathrm{L}(E)$, is inductively defined by:

- Basis step

$$\begin{aligned} \mathrm{L}(\varepsilon) &:= \{\varepsilon\}, \\ \mathrm{L}(\emptyset) &:= \emptyset, \\ \mathrm{L}(\boldsymbol{a}) &:= \{a\}. \end{aligned}$$

# Languages Denoted by Regular Expressions

## Definition

Let $E$ be a regular expression. The **language denoted** by $E$, denoted by $\mathrm{L}(E)$, is inductively defined by:

- Basis step

$$\mathrm{L}(\varepsilon) := \{\varepsilon\},$$
$$\mathrm{L}(\emptyset) := \emptyset,$$
$$\mathrm{L}(\boldsymbol{a}) := \{a\}.$$

- Inductive step

Let $\mathrm{L}(E)$ and $\mathrm{L}(F)$ be the languages denoted by the regular expressions $E$ and $F$, then

$$\mathrm{L}(E + F) := \mathrm{L}(E) \cup \mathrm{L}(F),$$
$$\mathrm{L}(E \cdot F) := \mathrm{L}(E) \cdot \mathrm{L}(F),$$
$$\mathrm{L}(E^*) := (\mathrm{L}(E))^*,$$
$$\mathrm{L}((E)) := \mathrm{L}(E).$$

# Languages Denoted by Regular Expressions

## Example

| $E$ | $\mathrm{L}(E)$ |
|---|---|
| $\boldsymbol{a} + \boldsymbol{b}$ | $\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$ |

# Languages Denoted by Regular Expressions

## Example

| $E$ | $\mathrm{L}(E)$ |
|---|---|
| $\boldsymbol{a} + \boldsymbol{b}$ | $\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$ |
| $\boldsymbol{a}^*$ | $\{\varepsilon, a, aa, aaa, \ldots\}$ |

# Languages Denoted by Regular Expressions

## Example

| $E$ | $\mathrm{L}(E)$ |
|---|---|
| $\boldsymbol{a} + \boldsymbol{b}$ | $\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$ |
| $\boldsymbol{a}^*$ | $\{\varepsilon, a, aa, aaa, ...\}$ |
| $(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})$ | $\mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) \cdot \mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) = \{a, b\} \cdot \{a, b\} = \{aa, ab, ba, bb\}$ |

# Languages Denoted by Regular Expressions

## Example

| $E$ | $\mathrm{L}(E)$ |
|---|---|
| $\boldsymbol{a} + \boldsymbol{b}$ | $\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$ |
| $\boldsymbol{a}^*$ | $\{\varepsilon, a, aa, aaa, ...\}$ |
| $(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})$ | $\mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) \cdot \mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) = \{a, b\} \cdot \{a, b\} = \{aa, ab, ba, bb\}$ |
| $\boldsymbol{a} + (\boldsymbol{ab})^*$ | $\{a, \varepsilon, ab, abab, ababab, ...\}$ |

# Languages Denoted by Regular Expressions

## Example

| $E$ | $\mathrm{L}(E)$ |
|---|---|
| $\boldsymbol{a}+\boldsymbol{b}$ | $\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$ |
| $\boldsymbol{a}^*$ | $\{\varepsilon, a, aa, aaa, ...\}$ |
| $(\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b})$ | $\mathrm{L}(\boldsymbol{a}+\boldsymbol{b}) \cdot \mathrm{L}(\boldsymbol{a}+\boldsymbol{b}) = \{a, b\} \cdot \{a, b\} = \{aa, ab, ba, bb\}$ |
| $\boldsymbol{a}+(\boldsymbol{ab})^*$ | $\{a, \varepsilon, ab, abab, ababab, ...\}$ |
| $(\mathbf{0}+\mathbf{1})^*\mathbf{01}(\mathbf{0}+\mathbf{1})^*$ | $\{\, x01y \mid x, y \in \{0,1\}^* \,\}$ |

# Languages Denoted by Regular Expressions

## Example

| $E$ | $\mathrm{L}(E)$ |
|---|---|
| $\boldsymbol{a}+\boldsymbol{b}$ | $\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$ |
| $\boldsymbol{a}^*$ | $\{\varepsilon, a, aa, aaa, \dots\}$ |
| $(\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b})$ | $\mathrm{L}(\boldsymbol{a}+\boldsymbol{b}) \cdot \mathrm{L}(\boldsymbol{a}+\boldsymbol{b}) = \{a, b\} \cdot \{a, b\} = \{aa, ab, ba, bb\}$ |
| $\boldsymbol{a}+(\boldsymbol{ab})^*$ | $\{a, \varepsilon, ab, abab, ababab, \dots\}$ |
| $(\mathbf{0}+\mathbf{1})^*\mathbf{01}(\mathbf{0}+\mathbf{1})^*$ | $\{\, x01y \mid x, y \in \{0, 1\}^* \,\}$ |
| $\boldsymbol{a_i}(\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a_n})^*$ | $\{\, w \in \Sigma^* \mid w$ starts by $a_i \,\}$ |

# Languages Denoted by Regular Expressions

## Example

Write a regular expression for the language $L$ defined by

$$L = \{\, w \in \{0,1\}^* \mid 0 \text{ and } 1 \text{ alternate in } w \,\}.$$

# Languages Denoted by Regular Expressions

## Example

Write a regular expression for the language $L$ defined by

$$L = \{\, w \in \{0,1\}^* \mid 0 \text{ and } 1 \text{ alternate in } w \,\}.$$

*Solution.*

$$(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{0}(\mathbf{10})^* + \mathbf{1}(\mathbf{01})^*$$

# Languages Denoted by Regular Expressions

## Example

Write a regular expression for the language $L$ defined by

$$L = \{\, w \in \{0,1\}^* \mid 0 \text{ and } 1 \text{ alternate in } w \,\}.$$

*Solution.*

$$(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{0}(\mathbf{10})^* + \mathbf{1}(\mathbf{01})^*$$

*Other solution.*

$$(\varepsilon + \mathbf{1})(\mathbf{01})^*(\varepsilon + \mathbf{0})$$

# Languages Denoted by Regular Expressions

## Example

The regular expression

$$(\mathbf{10} + \mathbf{0})^*(\varepsilon + \mathbf{1})$$

denotes the set of strings of $0$'s and $1$'s that have no two adjacent $1$'s.

# Languages Denoted by Regular Expressions

## Example

Write a regular expression for denoting the set of strings over $\Sigma = \{0, 1\}$ not ending in $01$.

# Languages Denoted by Regular Expressions

## Example

Write a regular expression for denoting the set of strings over $\Sigma = \{0, 1\}$ not ending in $01$.

*Solution.*

$$\varepsilon + \mathbf{0} + \mathbf{1} + (\mathbf{0} + \mathbf{1})^*(\mathbf{00} + \mathbf{10} + \mathbf{11})$$

# Derivatives of Regular Expressions

## Observation

The material on derivatives of regular expressions is from [Brzozowski 1964].

# Derivatives of Regular Expressions

## Observation

The material on derivatives of regular expressions is from [Brzozowski 1964].

## Definition

Let $L \subseteq \Sigma^*$ be a language and $a \in \Sigma$ a symbol. We define the **derivative** of $L$ by $a$, denoted by $\partial_a L$, by

$$\begin{gathered} \partial_a : \mathcal{P}\,\Sigma^* \to \mathcal{P}\,\Sigma^* \\ \partial_a L = \{\, x \in \Sigma^* \mid ax \in L \,\}. \end{gathered}$$

# Derivatives of Regular Expressions

## Observation

The material on derivatives of regular expressions is from [Brzozowski 1964].

## Definition

Let $L \subseteq \Sigma^*$ be a language and $a \in \Sigma$ a symbol. We define the **derivative** of $L$ by $a$, denoted by $\partial_a L$, by

$$\partial_a : \mathcal{P}\,\Sigma^* \to \mathcal{P}\,\Sigma^*$$
$$\partial_a L = \{\, x \in \Sigma^* \mid ax \in L \,\}.$$

## Example

$$\partial_a \{abab, abba\} = \{bab, bba\},$$
$$\partial_a \mathrm{L}(\boldsymbol{ab}^*) = \mathrm{L}(\boldsymbol{b}^*),$$
$$\partial_b \mathrm{L}(\boldsymbol{ab}^*) = \emptyset.$$

# Derivatives of Regular Expressions

## Definition

Let $E$ be a regular expression on $\Sigma$ and let $a \in \Sigma$ be a symbol. We define recursively the **derivative** of $E$ by $a$, denoted $\partial_a E$, by

$$\partial_a : \mathrm{RegEx} \to \mathrm{RegEx}$$

$$\begin{aligned}
\partial_a \emptyset &= \emptyset, \\
\partial_a \varepsilon &= \emptyset, \\
\partial_a \boldsymbol{a} &= \varepsilon, \\
\partial_a \boldsymbol{b} &= \emptyset, \quad \text{for } a \neq b,
\end{aligned}$$

$$\begin{aligned}
\partial_a (E + F) &= \partial_a E + \partial_a F, \\
\partial_a (EF) &= \begin{cases} (\partial_a E)F + \partial_a F, & \text{if } \varepsilon \in \mathrm{L}(E), \\ (\partial_a E)F, & \text{otherwise}, \end{cases} \\
\partial_a (E^*) &= (\partial_a E)E^*.
\end{aligned}$$

# Derivatives of Regular Expressions

## Definition

Let $E$ be a regular expression on $\Sigma$ and let $w \in \Sigma^*$ be a string. We define recursively the **derivative** of $E$ by $w$, denoted $\partial_w E$, by

$$\begin{aligned} \partial_w &: \mathrm{RegEx} \to \mathrm{RegEx} \\ \partial_\varepsilon E &= E, \\ \partial_{ax} E &= \partial_a(\partial_x E). \end{aligned}$$

# Derivatives of Regular Expressions

Theorem (Brzozowski [1964], Theorem 4.2)

Let $E$ be a regular expression on $\Sigma$ and let $w \in \Sigma^*$ be a string. Then

$$w \in \mathrm{L}(E) \qquad \Leftrightarrow \qquad \varepsilon \in \mathrm{L}(\partial_w E).$$

# Libraries

## Observation

Theoretical regular expressions $\neq$ practical regular expressions.

# Libraries

## Observation

Theoretical regular expressions $\neq$ practical regular expressions.

## Some programming languages with support to regular expressions

.NET, C, Haskell, Java, Mathematica, MATLAB and Perl.

# Algorithms

## Algorithms

See the Haskell implementation of some algorithms on regular expressions in the course homepage.

# Applications

## Some programs that use regular expressions

| | |
|---|---|
| GREP: | Print lines matching a pattern |
| AWK: | Pattern scanning and processing language |
| SED: | Stream editor for filtering and transforming text |
| ALEX, FLEX and LEX: | Lexical-analyser generators |
| EMACS and VIM: | Test editors |
| MYSQL and ORACLE: | Databases |

# Applications

## Reading

§ 3.3. Applications of Regular Expressions.

# Applications

## Reading

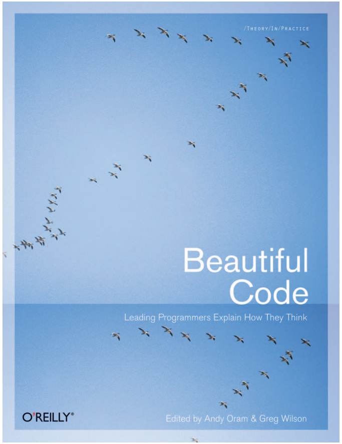
§ 3.3. Applications of Regular Expressions.

In the above section are defined:

$$E^+ := EE^* \qquad \text{(one or many times operator)}$$

$$E? := \varepsilon + E \qquad \text{(zero or one time operator)}$$

# An Implementation: A Regular Expression Matcher




*'Rob's implementation itself is a superb example of beautiful code: compact, elegant, efficient, and useful. It's one of the best examples of recursion that I have ever seen.'*

*Brian Kernighan, p. 3.*

# References

Brzozowski, J. A. (1964). Derivates of Regular Expressions. Journal of the ACM 11.4, pp. 481–494. DOI: 10.1145/321239.321249 (cit. on pp. 35–37, 40).

Hopcroft, J. E., Motwani, R. and Ullman, J. D. [1979] (2007). Introduction to Automata Theory, Languages, and Computation. 3rd ed. Pearson Education (cit. on p. 2).