# Lecture 6 – Regular Expressions and Languages

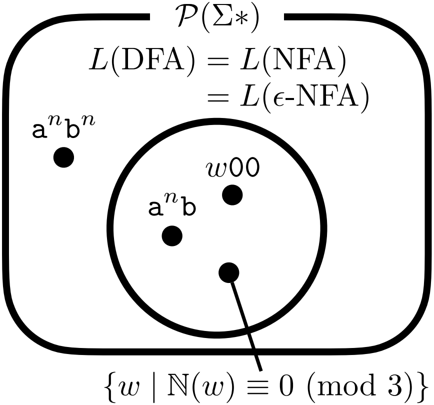
COSE215: Theory of Computation

Jihyeok Park

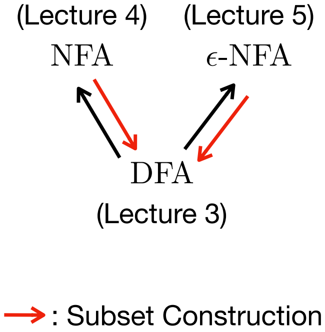
PLRG

2023 Spring

# Recall



$\mathcal{P}(\Sigma*)$
$L(\text{DFA}) = L(\text{NFA})$
$= L(\epsilon\text{-NFA})$
$\mathtt{a}^n\mathtt{b}^n$
$w\mathtt{00}$
$\mathtt{a}^n\mathtt{b}$
$\{w \mid \mathbb{N}(w) \equiv 0 \pmod 3\}$

(Lecture 4)
NFA
(Lecture 5)
$\epsilon$-NFA
DFA
(Lecture 3)
: Subset Construction

# Contents

# Operations in Languages

- The **union** of languages:

$$L_1 \cup L_2$$

- The **concatenation** of languages:

$$L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$$

- The **Kleene star** of a language:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$$

# Operations in Languages

- The **union** of languages:

$$L_1 \cup L_2$$

- The **concatenation** of languages:

$$L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$$

- The **Kleene star** of a language:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$$

$$L_1 = \{\mathtt{a}^n \mid n \geq 0\} \qquad L_2 = \{\mathtt{b}^n \mid n \geq 0\}$$

# Operations in Languages

- The **union** of languages:

$$L_1 \cup L_2$$

- The **concatenation** of languages:

$$L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$$

- The **Kleene star** of a language:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$$

$$L_1 = \{\mathtt{a}^n \mid n \geq 0\} \qquad L_2 = \{\mathtt{b}^n \mid n \geq 0\}$$

$$L_1 \cup L_2 = \{\mathtt{a}^n \text{ or } \mathtt{b}^n \mid n \geq 0\}$$

$$L_1 \cdot L_2 = \{\mathtt{a}^n\mathtt{b}^m \mid n, m \geq 0\}$$

$$L_1^* = L_1 = \{\mathtt{a}^n \mid n \geq 0\}$$

# Definition of Regular Expressions

## Definition (Regular Expressions)

A **regular expression** over a set of symbols $\Sigma$ is inductively defined as follows:

- **(Basis Case)** $\varnothing$, $\epsilon$, and $a \in \Sigma$ are regular expressions.
- **(Induction Case)** If $R_1$ and $R_2$ are regular expressions, then so are $R_1 \mid R_2$, $R_1 \cdot R_2$, $R^*$, and $(R)$.

$$
\begin{array}{rcll}
R & ::= & \varnothing & \text{(Empty)} \\
& \mid & \epsilon & \text{(Epsilon)} \\
& \mid & a & \text{(Symbol)} \\
& \mid & R \mid R & \text{(Union)} \\
& \mid & R \cdot R & \text{(Concatenation)} \\
& \mid & R^* & \text{(Kleene Star)} \\
& \mid & (R) & \text{(Parentheses)}
\end{array}
$$

# Definition of Regular Expressions

PLRG

## Definition (Regular Expressions)

A **regular expression** over a set of symbols $\Sigma$ is inductively defined as follows:

- **(Basis Case)** $\varnothing$, $\epsilon$, and $a \in \Sigma$ are regular expressions.
- **(Induction Case)** If $R_1$ and $R_2$ are regular expressions, then so are $R_1 \mid R_2$, $R_1 \cdot R_2$, $R^*$, and $(R)$.

$$
\begin{array}{rcll}
R & ::= & \varnothing & \text{(Empty)} \\
 & \mid & \epsilon & \text{(Epsilon)} \\
 & \mid & a & \text{(Symbol)} \\
 & \mid & R \mid R & \text{(Union)} \\
 & \mid & R \cdot R & \text{(Concatenation)} \\
 & \mid & R^* & \text{(Kleene Star)} \\
 & \mid & (R) & \text{(Parentheses)}
\end{array}
$$

$$\mathtt{a} \mid \epsilon \cdot \mathtt{b}^*$$

$$(\mathtt{a} \mid \epsilon) \cdot \mathtt{b}^*$$

**Precedence** of operators:

$$\mid \; < \; \cdot \; < \; ^*$$

# Definition of Regular Expressions

```scala
// The type definitions of symbols
type Symbol = Char

// The definition of regular expressions
trait RE
case class REEmpty() extends RE
case class REEpsilon() extends RE
case class RESymbol(symbol: Symbol) extends RE
case class REUnion(left: RE, right: RE) extends RE
case class REConcat(left: RE, right: RE) extends RE
case class REStar(re: RE) extends RE
case class REParen(re: RE) extends RE

// Two examples of regular expressions
val re1: RE = REUnion(
  RESymbol('a'),
  REConcat(REEpsilon(), REStar(RESymbol('b'))),
)
val re2: RE = REConcat(
  REParen(REUnion(RESymbol('a'), REEpsilon())),
  REStar(RESymbol('b')),
)
```

# Language of Regular Expressions

PLRG

## Definition (Language of Regular Expressions)

For a given regular expression $R$ on a set of symbols $\Sigma$, the **language** $L(R)$ of $R$ is inductively defined as follows:

$$
\begin{array}{lcllcl}
L(\varnothing) & = & \varnothing & L(R_1 \mid R_2) & = & L(R_1) \cup L(R_2) \\
L(\epsilon) & = & \{\epsilon\} & L(R_1 \cdot R_2) & = & L(R_1) \cdot L(R_2) \\
L(a) & = & \{a\} & L(R^*) & = & L(R)^* \\
 & & & L((R)) & = & L(R)
\end{array}
$$

# Language of Regular Expressions

## Definition (Language of Regular Expressions)

For a given regular expression $R$ on a set of symbols $\Sigma$, the **language** $L(R)$ of $R$ is inductively defined as follows:

$$
\begin{array}{lcl@{\qquad}lcl}
L(\varnothing) & = & \varnothing & L(R_1 \mid R_2) & = & L(R_1) \cup L(R_2) \\
L(\epsilon) & = & \{\epsilon\} & L(R_1 \cdot R_2) & = & L(R_1) \cdot L(R_2) \\
L(a) & = & \{a\} & L(R^*) & = & L(R)^* \\
 & & & L((R)) & = & L(R)
\end{array}
$$

$$
\begin{array}{lclcl}
L(\mathtt{a} \mid \epsilon \cdot \mathtt{b}^*) & = & L(\mathtt{a}) \cup L(\epsilon \cdot \mathtt{b}^*) & = & \{\mathtt{a}\} \cup L(\epsilon) \cdot L(\mathtt{b})^* \\
 & = & \{\mathtt{a}\} \cup \{\epsilon\} \cdot \{\mathtt{b}\}^* & = & \{\mathtt{a}\} \cup \{\mathtt{b}\}^* \\
 & = & \{\mathtt{a} \text{ or } \mathtt{b}^n \mid n \geq 0\} & &
\end{array}
$$

$$
\begin{array}{lclcl}
L((\mathtt{a} \mid \epsilon) \cdot \mathtt{b}^*) & = & L((\mathtt{a} \mid \epsilon)) \cdot L(\mathtt{b}^*) & = & L(\mathtt{a} \mid \epsilon) \cdot L(\mathtt{b})^* \\
 & = & (L(\mathtt{a}) \cup L(\epsilon)) \cdot L(\mathtt{b})^* & = & (\{\mathtt{a}\} \cup \{\epsilon\}) \cdot \{\mathtt{b}\}^* \\
 & = & \{\mathtt{ab}^n \text{ or } \mathtt{b}^n \mid n \geq 0\} & &
\end{array}
$$

# Extended Regular Expressions

More operators:

$$
\begin{array}{rcll}
R & ::= & \cdots & \\
& | & R^{+} & \text{(Kleene plus)} \\
& | & R^{?} & \text{(Optional)}
\end{array}
$$

# Extended Regular Expressions

More operators:

$$
\begin{array}{rcll}
R & ::= & \cdots & \\
& \mid & R^{+} & \text{(Kleene plus)} \\
& \mid & R^{?} & \text{(Optional)}
\end{array}
$$

Actually, they are just **syntactic sugar** for the existing operators:

$$
\begin{array}{rcccl}
L(R^{+}) & = & L(RR^{*}) & = & L(R) \cdot L(R^{*}) \\
L(R^{?}) & = & L(R \mid \epsilon) & = & L(R) \cup L(\epsilon)
\end{array}
$$

# Extended Regular Expressions

More operators:

$$\begin{array}{rcll} R & ::= & \cdots & \\ & | & R^+ & \text{(Kleene plus)} \\ & | & R^? & \text{(Optional)} \end{array}$$

Actually, they are just **syntactic sugar** for the existing operators:

$$\begin{array}{rcccl} L(R^+) & = & L(RR^*) & = & L(R) \cdot L(R^*) \\ L(R^?) & = & L(R \mid \epsilon) & = & L(R) \cup L(\epsilon) \end{array}$$

For examples,

$$\begin{array}{rcl} L(\mathtt{a}^+) & = & L(\mathtt{a}\mathtt{a}^*) = L(\mathtt{a}) \cdot L(\mathtt{a}^*) = \{\mathtt{a}\} \cdot \{\mathtt{a}\}^* = \{\mathtt{a}^n \mid n \geq 1\} \\ L(\mathtt{b}^?) & = & L(\mathtt{b} \mid \epsilon) = L(\mathtt{b}) \cup L(\epsilon) = \{\mathtt{b}\} \cup \{\epsilon\} = \{\mathtt{b}, \epsilon\} \end{array}$$

# Examples

- $L = \{\epsilon, \mathtt{a}\}$
- $L = \{w \in \{0, 1\}^* \mid w \text{ contains at least two } 0's\}$
- $L = \{w \in \{0, 1\}^* \mid w \text{ contains exactly two } 0's\}$
- $L = \{w \in \{0, 1\}^* \mid w \text{ has three consecutive } 0's\}$
- $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \ \mathtt{a} \text{ and } \mathtt{b} \text{ alternate in } w\}$

# Examples

- $L = \{\mathrm{a}^n\mathrm{b}^m \mid n \geq 3 \wedge m \equiv 0 (\mathrm{mod}\ 2)\}$

- $L = \{\mathrm{a}^n\mathrm{b}^m \mid n + m \equiv 0 (\mathrm{mod}\ 2)\}$

- $L = \{w \in \{0, 1\}^* \mid \text{ the number of 0's is divisible by 3}\}$

- $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod 3\}$
  where $\mathbb{N}(w)$ is a natural number represented by $w$.

# Examples

- $L = \{\mathrm{a}^n\mathrm{b}^m \mid n \geq 3 \wedge m \equiv 0 (\mathrm{mod}\ 2)\}$

- $L = \{\mathrm{a}^n\mathrm{b}^m \mid n + m \equiv 0 (\mathrm{mod}\ 2)\}$

- $L = \{w \in \{0, 1\}^* \mid \text{ the number of 0's is divisible by 3}\}$

- $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod 3\}$
  where $\mathbb{N}(w)$ is a natural number represented by $w$.

  $$\texttt{(0|1(01}^*\texttt{0)}^*\texttt{1)}^*$$

- $L = \{\mathrm{a}^n\mathrm{b}^n \mid n \geq 0\}$

# Examples

- $L = \{a^n b^m \mid n \geq 3 \land m \equiv 0 (\mathrm{mod}\ 2)\}$

- $L = \{a^n b^m \mid n + m \equiv 0 (\mathrm{mod}\ 2)\}$

- $L = \{w \in \{0, 1\}^* \mid \text{ the number of 0's is divisible by 3}\}$

- $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod{3}\}$
  where $\mathbb{N}(w)$ is a natural number represented by $w$.

$$\texttt{(0|1(01}^*\texttt{0)}^*\texttt{1)}^*$$

- $L = \{a^n b^n \mid n \geq 0\}$ – IMPOSSIBLE ($\nexists$ RE $R$. $L(R) = L$)

# Exercise #1

- Please see https://github.com/ku-plrg-classroom/docs/tree/main/fa-examples.
- You don't have to submit it. This is just exercise for your practice.
- The goal is to implement the finite automata (FA) objects in the `Implementation.scala` file.

# Summary

1. Operations in Languages
   - Union
   - Concatenation
   - Kleene Star

2. Regular Expressions
   - Definition
   - Language of Regular Expressions
   - Extended Regular Expressions
   - Examples

# Next Lecture

- Equivalence of Regular Expressions and Finite Automata


Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr