Lecture 6 – Regular Expressions and Languages

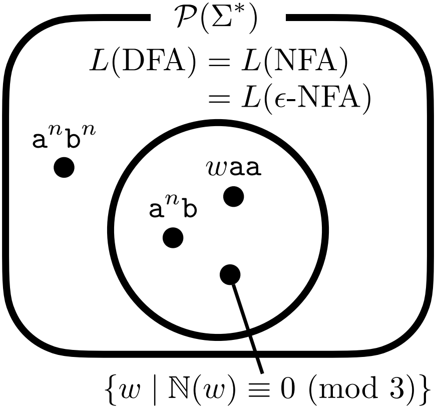
COSE215: Theory of Computation

Jihyeok Park

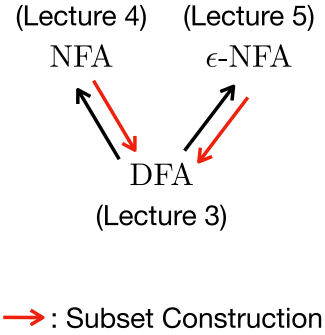
PLRG

2024 Spring

Recall



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



(Lecture 4)
NFA
(Lecture 5)
ϵ-NFA
DFA
(Lecture 3)
: Subset Construction

Contents

Contents

Recall: Operations in Languages

We already learned the following **operations** on languages:

- **Union** of languages: $L_1 \cup L_2$
- **Concatenation** of languages: $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$
- **Kleene star** of a language: $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$

Recall: Operations in Languages

We already learned the following **operations** on languages:

- **Union** of languages: $L_1 \cup L_2$
- **Concatenation** of languages: $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$
- **Kleene star** of a language: $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$

For example, consider the following languages over symbols $\Sigma = \{\mathtt{a}, \mathtt{b}\}$:

$$L_1 = \{\mathtt{a}^n \mid n \geq 1\} \qquad\qquad L_2 = \{\mathtt{b}^n \mid n \geq 1\}$$

Recall: Operations in Languages

We already learned the following **operations** on languages:

- **Union** of languages: $L_1 \cup L_2$
- **Concatenation** of languages: $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$
- **Kleene star** of a language: $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$

For example, consider the following languages over symbols $\Sigma = \{\texttt{a}, \texttt{b}\}$:

$$L_1 = \{\texttt{a}^n \mid n \geq 1\} \qquad\qquad L_2 = \{\texttt{b}^n \mid n \geq 1\}$$

$$L_1 \cup L_2 \quad = \quad \{\texttt{a}^n \text{ or } \texttt{b}^n \mid n \geq 1\}$$

Recall: Operations in Languages

We already learned the following **operations** on languages:

- **Union** of languages: $L_1 \cup L_2$
- **Concatenation** of languages: $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$
- **Kleene star** of a language: $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$

For example, consider the following languages over symbols $\Sigma = \{\mathtt{a}, \mathtt{b}\}$:

$$L_1 = \{\mathtt{a}^n \mid n \geq 1\} \qquad L_2 = \{\mathtt{b}^n \mid n \geq 1\}$$

$$\begin{array}{rcl} L_1 \cup L_2 & = & \{\mathtt{a}^n \text{ or } \mathtt{b}^n \mid n \geq 1\} \\ L_1 L_2 & = & \{\mathtt{a}^n \mathtt{b}^m \mid n, m \geq 1\} \neq \{\mathtt{a}^n \mathtt{b}^n \mid n \geq 1\} \end{array}$$

Recall: Operations in Languages

We already learned the following **operations** on languages:

- **Union** of languages: $L_1 \cup L_2$
- **Concatenation** of languages: $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$
- **Kleene star** of a language: $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$

For example, consider the following languages over symbols $\Sigma = \{\mathtt{a}, \mathtt{b}\}$:

$$L_1 = \{\mathtt{a}^n \mid n \geq 1\} \qquad L_2 = \{\mathtt{b}^n \mid n \geq 1\}$$

$$\begin{array}{rcllcl} L_1 \cup L_2 &=& \{\mathtt{a}^n \text{ or } \mathtt{b}^n \mid n \geq 1\} & & & \\ L_1 L_2 &=& \{\mathtt{a}^n\mathtt{b}^m \mid n, m \geq 1\} & \neq & \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 1\} \\ L_1^* &=& \{\mathtt{a}^n \mid n \geq 0\} & \neq & \{\mathtt{a}^n \mid n \geq 1\} \end{array}$$

Recall: Operations in Languages

We already learned the following **operations** on languages:

- **Union** of languages: $L_1 \cup L_2$
- **Concatenation** of languages: $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \land w_2 \in L_2\}$
- **Kleene star** of a language: $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{n \geq 0} L^n$

For example, consider the following languages over symbols $\Sigma = \{\mathtt{a}, \mathtt{b}\}$:

$$L_1 = \{\mathtt{a}^n \mid n \geq 1\} \qquad L_2 = \{\mathtt{b}^n \mid n \geq 1\}$$

$$\begin{array}{rcl} L_1 \cup L_2 & = & \{\mathtt{a}^n \text{ or } \mathtt{b}^n \mid n \geq 1\} \\ L_1 L_2 & = & \{\mathtt{a}^n\mathtt{b}^m \mid n, m \geq 1\} \quad \neq \quad \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 1\} \\ L_1^* & = & \{\mathtt{a}^n \mid n \geq 0\} \quad \neq \quad \{\mathtt{a}^n \mid n \geq 1\} \end{array}$$

Regular expressions (REs) provide a new way to define languages with above **operations** without using finite automata!

Definition of Regular Expressions

Definition (Regular Expressions)

A **regular expression** over a set of symbols Σ is inductively defined as:

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

Definition of Regular Expressions

Definition (Regular Expressions)

A **regular expression** over a set of symbols Σ is inductively defined as:

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

The following is the **syntax** of regular expressions and examples:

$$
\begin{array}{rclll}
R & ::= & \varnothing & \text{(Empty)} & \mid \; R \mid R \quad \text{(Union)} \\
 & \mid & \epsilon & \text{(Epsilon)} & \mid \; R\;R \quad \text{(Concatenation)} \\
 & \mid & x & \text{(Symbol)} & \mid \; R^* \quad \text{(Kleene Star)} \\
 & & & & \mid \; (R) \quad \text{(Parentheses)}
\end{array}
$$

$\varnothing$	ϵ	a	a|b	ab
a^*	$a(\varnothing \mid c)^*$	$(a\epsilon) \mid b^*$	$(a(bc^*b)^*)^*$	$(a\varnothing a) \mid b^*$

Precedence Order

Arithmetic expressions have the following **precedence order**:

$$\times \qquad > \qquad +$$

It means that multiplication ($\times$) has higher precedence than addition ($+$). For example,

$$1 + 2 \times 3 \qquad \text{means} \qquad 1 + (2 \times 3)$$

Precedence Order

Arithmetic expressions have the following **precedence order**:

$$\times \qquad > \qquad +$$

It means that multiplication ($\times$) has higher precedence than addition ($+$). For example,

$$1 + 2 \times 3 \qquad \text{means} \qquad 1 + (2 \times 3)$$

Similarly, **regular expressions** have the following **precedence order**:

$$* \qquad > \qquad \cdot \qquad > \qquad |$$

Precedence Order

Arithmetic expressions have the following **precedence order**:

$$\times \quad > \quad +$$

It means that multiplication ($\times$) has higher precedence than addition ($+$). For example,

$$1 + 2 \times 3 \qquad \text{means} \qquad 1 + (2 \times 3)$$

Similarly, **regular expressions** have the following **precedence order**:

$$^* \quad > \quad \cdot \quad > \quad \mid$$

For example,

$$\texttt{a} \mid \epsilon \texttt{b}^* \qquad \text{means} \qquad \texttt{a} \mid (\epsilon (\texttt{b}^*))$$

$$(\texttt{a} \mid \epsilon) \texttt{b}^* \qquad \text{means} \qquad (\texttt{a} \mid \epsilon) (\texttt{b}^*)$$

Definition of Regular Expressions

```
// The definition of regular expressions
enum RE:
  case Emp                                      // ∅
  case Eps                                      // ϵ
  case Sym(symbol: Symbol)                      // x
  case Union(left: RE, right: RE)               // R1 | R2
  case Concat(left: RE, right: RE)              // R1 R2
  case Star(re: RE)                             // R*
```

Definition of Regular Expressions

```
// The definition of regular expressions
enum RE:
  case Emp                                    // ∅
  case Eps                                    // ϵ
  case Sym(symbol: Symbol)                    // x
  case Union(left: RE, right: RE)             // R1 | R2
  case Concat(left: RE, right: RE)            // R1 R2
  case Star(re: RE)                           // R*
```

In the algebraic data type (ADT) of regular expressions, we do **not need to explicitly define the parentheses** because it is already handled by the structure of the ADT.

```
// import all constructors (Emp, Eps, Sym, Union, Concat, Star) of RE
import RE.*

// a | ϵ b*
val re1: RE = Union(Sym('a'), Concat(Eps, Star(Sym('b'))))

// (a | ϵ) b*
val re2: RE = Concat(Union(Sym('a'), Eps), Star(Sym('b')))
```

Language of Regular Expressions

Definition (Language of Regular Expressions)

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

$$
\begin{array}{rcl}
L(\varnothing) & = & \varnothing \\
L(\epsilon) & = & \{\epsilon\} \\
L(x) & = & \{x\}
\end{array}
\qquad
\begin{array}{rcl}
L(R_1 \mid R_2) & = & L(R_1) \cup L(R_2) \\
L(R_1 R_2) & = & L(R_1)L(R_2) \\
L(R^*) & = & L(R)^* \\
L((R)) & = & L(R)
\end{array}
$$

Language of Regular Expressions

Definition (Language of Regular Expressions)

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

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

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

Language of Regular Expressions

Definition (Language of Regular Expressions)

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

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

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

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

Extended Regular Expressions

More operators can be added to regular expressions:

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

(Note that $^{+}$ and $^{?}$ have same precedence as *.)

Extended Regular Expressions

More operators can be added to regular expressions:

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

(Note that $^{+}$ and $^{?}$ have same precedence as *.)

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

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

Extended Regular Expressions

More operators can be added to regular expressions:

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

(Note that $^{+}$ and $^{?}$ have same precedence as *.)

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

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

For examples,

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

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

$$\epsilon \,|\, \mathtt{a} \,|\, \mathtt{b} \qquad \text{or} \qquad (\mathtt{a} \,|\, \mathtt{b})^?$$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

  $\epsilon\,|\,\mathtt{a}\,|\,\mathtt{b}$ or $(\mathtt{a}\,|\,\mathtt{b})^?$

- $L = \{w \in \{0, 1\}^* \mid w \text{ contains exactly two } 0's\}$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

$$\epsilon \,|\, \mathtt{a} \,|\, \mathtt{b} \qquad \text{or} \qquad (\mathtt{a} \,|\, \mathtt{b})^?$$

- $L = \{w \in \{0, 1\}^* \mid w \text{ contains exactly two } 0's\}$

$$1^*01^*01^*$$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

$$\epsilon\,|\,\mathtt{a}\,|\,\mathtt{b} \qquad \text{or} \qquad (\mathtt{a}\,|\,\mathtt{b})^?$$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains exactly two } 0's\}$

$$1^*01^*01^*$$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains at least two } 0's\}$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

  $\epsilon\,|\,\mathtt{a}\,|\,\mathtt{b}$ or $(\mathtt{a}\,|\,\mathtt{b})^?$

- $L = \{w \in \{0, 1\}^* \mid w \text{ contains exactly two } 0's\}$

  $1^*01^*01^*$

- $L = \{w \in \{0, 1\}^* \mid w \text{ contains at least two } 0's\}$

  $(0\,|\,1)^*0(0\,|\,1)^*0(0\,|\,1)^*$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

$$\epsilon \,|\, \mathtt{a} \,|\, \mathtt{b} \qquad \text{or} \qquad (\mathtt{a} \,|\, \mathtt{b})^?$$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains exactly two } 0's\}$

$$1^*01^*01^*$$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains at least two } 0's\}$

$$(0|1)^*0(0|1)^*0(0|1)^*$$

- $L = \{w \in \{0,1\}^* \mid w \text{ has three consecutive } 0's\}$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

  $\epsilon\,|\,\mathtt{a}\,|\,\mathtt{b}$ or $(\mathtt{a}\,|\,\mathtt{b})^?$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains exactly two } 0's\}$

  $1^*01^*01^*$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains at least two } 0's\}$

  $(0\,|\,1)^*0(0\,|\,1)^*0(0\,|\,1)^*$

- $L = \{w \in \{0,1\}^* \mid w \text{ has three consecutive } 0's\}$

  $(0\,|\,1)^*000(0\,|\,1)^*$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

  $$\epsilon \,|\, \mathtt{a} \,|\, \mathtt{b} \qquad \text{or} \qquad (\mathtt{a}\,|\,\mathtt{b})^?$$

- $L = \{w \in \{0, 1\}^* \mid w \text{ contains exactly two } 0's\}$

  $$\mathtt{1^*01^*01^*}$$

- $L = \{w \in \{0, 1\}^* \mid w \text{ contains at least two } 0's\}$

  $$\mathtt{(0|1)^*0(0|1)^*0(0|1)^*}$$

- $L = \{w \in \{0, 1\}^* \mid w \text{ has three consecutive } 0's\}$

  $$\mathtt{(0|1)^*000(0|1)^*}$$

- $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \mathtt{a} \text{ and } \mathtt{b} \text{ alternate in } w\}$

Examples

- $L = \{\epsilon, \mathtt{a}, \mathtt{b}\}$

  $\epsilon\,|\,\mathtt{a}\,|\,\mathtt{b}$ or $(\mathtt{a}\,|\,\mathtt{b})^?$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains exactly two } 0's\}$

  $\mathtt{1}^*\mathtt{0}\mathtt{1}^*\mathtt{0}\mathtt{1}^*$

- $L = \{w \in \{0,1\}^* \mid w \text{ contains at least two } 0's\}$

  $(\mathtt{0}\,|\,\mathtt{1})^*\mathtt{0}(\mathtt{0}\,|\,\mathtt{1})^*\mathtt{0}(\mathtt{0}\,|\,\mathtt{1})^*$

- $L = \{w \in \{0,1\}^* \mid w \text{ has three consecutive } 0's\}$

  $(\mathtt{0}\,|\,\mathtt{1})^*\mathtt{000}(\mathtt{0}\,|\,\mathtt{1})^*$

- $L = \{w \in \{\mathtt{a},\mathtt{b}\}^* \mid \mathtt{a} \text{ and } \mathtt{b} \text{ alternate in } w\}$

  $\mathtt{a}^?(\mathtt{ba})^*\mathtt{b}^?$

Examples

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

Examples

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

$$\mathtt{aaa}^+(\mathtt{bb})^*$$

Examples

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

$$\mathtt{aaa}^+\mathtt{(bb)}^*$$

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

Examples

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

$$\mathtt{aaa}^{+}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(aa)}^{*}\mathtt{(ab)}^{?}\mathtt{(bb)}^{*}$$

Examples

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

$$\mathtt{aaa}^+\mathtt{(bb)}^*$$

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

$$\mathtt{(aa)}^*\mathtt{(ab)}^?\mathtt{(bb)}^*$$

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

Examples

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

$$\mathtt{aaa}^{+}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(aa)}^{*}\mathtt{(ab)}^{?}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(1^{*}01^{*}01^{*}01^{*})}^{*}$$

Examples

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

$$\mathtt{aaa}^+(\mathtt{bb})^*$$

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

$$(\mathtt{aa})^*(\mathtt{ab})^?(\mathtt{bb})^*$$

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

$$(\mathtt{1}^*\mathtt{01}^*\mathtt{01}^*\mathtt{01}^*)^*$$

- $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \ (\text{mod } 3)\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary

Examples

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

$$\mathtt{aaa}^{+}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(aa)}^{*}\mathtt{(ab)}^{?}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(1^{*}01^{*}01^{*}01^{*})}^{*}$$

- $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \ (\text{mod } 3)\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary

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

Examples

PLRG

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

$$\mathtt{aaa}^{+}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(aa)}^{*}\mathtt{(ab)}^{?}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(1}^{*}\mathtt{01}^{*}\mathtt{01}^{*}\mathtt{01}^{*}\mathtt{)}^{*}$$

- $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0\ (\mathrm{mod}\ 3)\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary

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

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

Examples

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

$$\mathtt{aaa}^{+}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(aa)}^{*}\mathtt{(ab)}^{?}\mathtt{(bb)}^{*}$$

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

$$\mathtt{(1^{*}01^{*}01^{*}01^{*})}^{*}$$

- $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \ (\mathrm{mod}\ 3)\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary

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

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

Equivalence of Regular Expressions

We say two regular expressions R_1 and R_2 are **equivalent** ($R_1 \equiv R_2$) if their languages are the same: $L(R_1) = L(R_2)$.

Equivalence of Regular Expressions

We say two regular expressions R_1 and R_2 are **equivalent** ($R_1 \equiv R_2$) if their languages are the same: $L(R_1) = L(R_2)$.

Regular expressions have following equivalence relations:

- **Associativity** for union and concatenation:

$$R_1 \mid (R_2 \mid R_3) \equiv (R_1 \mid R_2) \mid R_3 \qquad \text{and} \qquad R_1(R_2 R_3) \equiv (R_1 R_2) R_3$$

Equivalence of Regular Expressions

We say two regular expressions R_1 and R_2 are **equivalent** ($R_1 \equiv R_2$) if their languages are the same: $L(R_1) = L(R_2)$.

Regular expressions have following equivalence relations:

- **Associativity** for union and concatenation:

$$R_1 \mid (R_2 \mid R_3) \equiv (R_1 \mid R_2) \mid R_3 \qquad \text{and} \qquad R_1(R_2 R_3) \equiv (R_1 R_2) R_3$$

- **Commutativity** for union:

$$R_1 \mid R_2 \equiv R_2 \mid R_1$$

Equivalence of Regular Expressions

We say two regular expressions R_1 and R_2 are **equivalent** ($R_1 \equiv R_2$) if their languages are the same: $L(R_1) = L(R_2)$.

Regular expressions have following equivalence relations:

- **Associativity** for union and concatenation:

$$R_1 \mid (R_2 \mid R_3) \equiv (R_1 \mid R_2) \mid R_3 \qquad \text{and} \qquad R_1(R_2 R_3) \equiv (R_1 R_2) R_3$$

- **Commutativity** for union:

$$R_1 \mid R_2 \equiv R_2 \mid R_1$$

- Left and right **distributive laws**:

$$(R_1 \mid R_2) R_3 \equiv R_1 R_3 \mid R_2 R_3 \qquad \text{and} \qquad R_1(R_2 \mid R_3) \equiv R_1 R_2 \mid R_1 R_3$$

Equivalence of Regular Expressions

- $\varnothing$ and ϵ are **identity** for union and concatenation:

$$R \mid \varnothing \equiv \varnothing \mid R \equiv R \qquad \text{and} \qquad R\epsilon \equiv \epsilon R \equiv R$$

Equivalence of Regular Expressions

- $\varnothing$ and ϵ are **identity** for union and concatenation:

$$R \,|\, \varnothing \equiv \varnothing \,|\, R \equiv R \qquad \text{and} \qquad R\epsilon \equiv \epsilon R \equiv R$$

- $\varnothing$ is **annihilator** for concatenation:

$$R\varnothing \equiv \varnothing R \equiv \varnothing$$

Equivalence of Regular Expressions

- $\varnothing$ and ϵ are **identity** for union and concatenation:

$$R \mid \varnothing \equiv \varnothing \mid R \equiv R \qquad \text{and} \qquad R\epsilon \equiv \epsilon R \equiv R$$

- $\varnothing$ is **annihilator** for concatenation:

$$R\varnothing \equiv \varnothing R \equiv \varnothing$$

- **Idempotent Law** for union:

$$R \mid R \equiv R$$

Equivalence of Regular Expressions

- $\varnothing$ and ϵ are **identity** for union and concatenation:

$$R \mid \varnothing \equiv \varnothing \mid R \equiv R \qquad \text{and} \qquad R\epsilon \equiv \epsilon R \equiv R$$

- $\varnothing$ is **annihilator** for concatenation:

$$R\varnothing \equiv \varnothing R \equiv \varnothing$$

- **Idempotent Law** for union:

$$R \mid R \equiv R$$

- Laws involving Kleene star:

$$(R^*)^* \equiv R^* \qquad \text{and} \qquad \varnothing^* \equiv \epsilon \qquad \text{and} \qquad \epsilon^* \equiv \epsilon$$

$$\epsilon \mid R^* \equiv R^* \mid \epsilon \equiv R^* \qquad \text{and} \qquad R \mid R^* \equiv R^* \mid R \equiv R^*$$

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

For example,

$((\mathtt{a}\varnothing)^*(\mathtt{b}|\varnothing|\mathtt{b}^*))^*$

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

For example,

$$\texttt{((a}\varnothing\texttt{)}^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* \quad \equiv \quad \texttt{(}\varnothing^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* \quad (\because R\varnothing \equiv \varnothing - \text{Annihilator})$$

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

For example,

$$
\begin{aligned}
\texttt{((a}\varnothing\texttt{)}^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* &\equiv \texttt{(}\varnothing^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* && (\because R\varnothing \equiv \varnothing - \text{Annihilator}) \\
&\equiv \texttt{(}\epsilon\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* && (\because \varnothing^* \equiv \epsilon)
\end{aligned}
$$

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

For example,

$$
\begin{aligned}
\texttt{((a}\varnothing\texttt{)}^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* &\equiv \texttt{(}\varnothing^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* && (\because R\varnothing \equiv \varnothing \text{ -- Annihilator}) \\
&\equiv \texttt{(}\epsilon\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* && (\because \varnothing^* \equiv \epsilon) \\
&\equiv \texttt{(b|}\varnothing\texttt{|b}^*\texttt{)}^* && (\because \epsilon R \equiv R \text{ -- Identity})
\end{aligned}
$$

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

For example,

$$
\begin{aligned}
\texttt{((a}\varnothing\texttt{)}^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* &\equiv \texttt{(}\varnothing^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* && (\because R\varnothing \equiv \varnothing - \text{Annihilator}) \\
&\equiv \texttt{(}\epsilon\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* && (\because \varnothing^* \equiv \epsilon) \\
&\equiv \texttt{(b|}\varnothing\texttt{|b}^*\texttt{)}^* && (\because \epsilon R \equiv R - \text{Identity}) \\
&\equiv \texttt{(b|b}^*\texttt{)}^* && (\because R \,|\, \varnothing \equiv R - \text{Identity})
\end{aligned}
$$

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

For example,

$$
\begin{array}{rcll}
\texttt{((a}\varnothing\texttt{)}^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* & \equiv & \texttt{(}\varnothing^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* & (\because R\varnothing \equiv \varnothing - \text{Annihilator}) \\
& \equiv & \texttt{(}\epsilon\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* & (\because \varnothing^* \equiv \epsilon) \\
& \equiv & \texttt{(b|}\varnothing\texttt{|b}^*\texttt{)}^* & (\because \epsilon R \equiv R - \text{Identity}) \\
& \equiv & \texttt{(b|b}^*\texttt{)}^* & (\because R \mid \varnothing \equiv R - \text{Identity}) \\
& \equiv & \texttt{(b}^*\texttt{)}^* & (\because R \mid R^* \equiv R^*)
\end{array}
$$

Simplifying Regular Expressions

We can simplify regular expressions using the equivalence laws.

For example,

$$
\begin{array}{rcll}
\texttt{((a}\varnothing\texttt{)}^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* & \equiv & \texttt{(}\varnothing^*\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* & (\because R\varnothing \equiv \varnothing \text{ – Annihilator}) \\
& \equiv & \texttt{(}\epsilon\texttt{(b|}\varnothing\texttt{|b}^*\texttt{))}^* & (\because \varnothing^* \equiv \epsilon) \\
& \equiv & \texttt{(b|}\varnothing\texttt{|b}^*\texttt{)}^* & (\because \epsilon R \equiv R \text{ – Identity}) \\
& \equiv & \texttt{(b|b}^*\texttt{)}^* & (\because R \,|\, \varnothing \equiv R \text{ – Identity}) \\
& \equiv & \texttt{(b}^*\texttt{)}^* & (\because R \,|\, R^* \equiv R^*) \\
& \equiv & \texttt{b}^* & (\because (R^*)^* \equiv R^*)
\end{array}
$$

Contents

Regular Expressions in Practice

Most **programming languages** support **regular expressions**:

- **Scala** – `scala.util.matching.Regex` class
- **Python** – `re` module
- **JavaScript** – `RegExp` object
- **Rust** – `regex` crate
- ...

Regular Expressions in Practice

Most **programming languages** support **regular expressions**:

- **Scala** – `scala.util.matching.Regex` class
- **Python** – `re` module
- **JavaScript** – `RegExp` object
- **Rust** – `regex` crate
- ...

For example, we can convert a string to a regular expression (`Regex`) object by using the `r` method in Scala:

```scala
import scala.util.matching.Regex

val re: Regex = "(a|b)c*".r
re.matches("a")      // true
re.matches("b")      // true
re.matches("accc")   // true
re.matches("bccccc") // true
re.matches("ba")     // false
re.matches("cba")    // false
re.matches("aacc")   // false
re.matches("cccccc") // false
```

Regular Expressions in Practice

In practice, regular expressions support more syntactic sugar:

Syntax	Description
`^`	start of the line
`$`	start and end of the line
`.`	any character
`[]`	any character in the set
`[^ ]`	any character not in the set
`\d`	any digit
`\w`	any alphanumeric character

Regular Expressions in Practice

In practice, regular expressions support more syntactic sugar:

Syntax	Description
`^`	start of the line
`$`	start and end of the line
`.`	any character
`[]`	any character in the set
`[^ ]`	any character not in the set
`\d`	any digit
`\w`	any alphanumeric character

`"ci[dait]*".r` `"\\w+$".r` `"\\d+".r`

For example, above Scala regular expressions find patterns in each string:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut 53 et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation 42 laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate 129 esse cillum dolore eu
fugiat nulla 5323. Excepteur sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim id est laborum.
```

Summary

Next Lecture

PLRG

- Equivalence of Regular Expressions and Finite Automata

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