CSE 135: Introduction to Theory of Computation
Regular Expressions

Sungjin Im

University of California, Merced

02-10-2014
Operations on Languages

▶ Recall: A language is a set of strings

▶ A simple but powerful collection of operations:
  - Union, Concatenation and Kleene Closure
Operations on Languages

- Recall: A language is a set of strings
- We can consider new languages derived from operations on given languages
Operations on Languages

- Recall: A language is a set of strings
- We can consider new languages derived from operations on given languages
  - e.g., $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1^*$, ...
Recall: A language is a set of strings
We can consider new languages derived from operations on given languages
e.g., $L_1 \cup L_2$, $L_1 \cap L_2$, ...
Operations on Languages

- Recall: A language is a set of strings
- We can consider new languages derived from operations on given languages
  - e.g., \( L_1 \cup L_2 \), \( L_1 \cap L_2 \), \( \frac{1}{2} L \), ...
Operations on Languages

- Recall: A language is a set of strings
- We can consider new languages derived from operations on given languages
  - e.g., $L_1 \cup L_2$, $L_1 \cap L_2$, $\frac{1}{2}L$, ... 
- A simple but powerful collection of operations:
Operations on Languages

- Recall: A language is a set of strings
- We can consider new languages derived from operations on given languages
  - e.g., $L_1 \cup L_2$, $L_1 \cap L_2$, $\frac{1}{2}L$, …
- A simple but powerful collection of operations:
  - Union, Concatenation and Kleene Closure
Concatenation of Languages

Definition
Given languages $L_1$ and $L_2$, we define their concatenation to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, \ y \in L_2\}$
Concatenation of Languages

Definition
Given languages $L_1$ and $L_2$, we define their concatenation to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$

Example

- $L_1 = \{\text{hello}\}$ and $L_2 = \{\text{world}\}$ then $L_1 \circ L_2 =$

  $\{\text{helloworld}\}$
Concatenation of Languages

Definition
Given languages $L_1$ and $L_2$, we define their concatenation to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$

Example
- $L_1 = \{\text{hello}\}$ and $L_2 = \{\text{world}\}$ then $L_1 \circ L_2 = \{\text{helloworld}\}$
Definition
Given languages $L_1$ and $L_2$, we define their *concatenation* to be the language $L_1 \circ L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$

Example

- $L_1 = \{ \text{hello} \}$ and $L_2 = \{ \text{world} \}$ then $L_1 \circ L_2 = \{ \text{helloworld} \}$
- $L_1 = \{ 00, 10 \}; L_2 = \{ 0, 1 \}$. $L_1 \circ L_2 = \{ \}$
Concatenation of Languages

**Definition**
Given languages \( L_1 \) and \( L_2 \), we define their *concatenation* to be the language \( L_1 \circ L_2 = \{ xy \mid x \in L_1, \ y \in L_2 \} \)

**Example**
- \( L_1 = \{ \text{hello} \} \) and \( L_2 = \{ \text{world} \} \) then \( L_1 \circ L_2 = \{ \text{helloworld} \} \)
- \( L_1 = \{ 00, 10 \}; \ L_2 = \{ 0, 1 \} \). \( L_1 \circ L_2 = \{ 000, 001, 100, 101 \} \)
Definition
Given languages $L_1$ and $L_2$, we define their *concatenation* to be the language $L_1 \circ L_2 = \{ xy \mid x \in L_1, \ y \in L_2 \}$

Example
- $L_1 = \{ \text{hello} \}$ and $L_2 = \{ \text{world} \}$ then $L_1 \circ L_2 = \{ \text{helloworld} \}$
- $L_1 = \{00, 10\}; \ L_2 = \{0, 1\}$. $L_1 \circ L_2 = \{000, 001, 100, 101\}$
- $L_1 =$ set of strings ending in 0; $L_2 =$ set of strings beginning with 01. $L_1 \circ L_2 =$
Concatenation of Languages

Definition
Given languages $L_1$ and $L_2$, we define their concatenation to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$

Example

- $L_1 = \{\text{hello}\}$ and $L_2 = \{\text{world}\}$ then $L_1 \circ L_2 = \{\text{helloworld}\}$
- $L_1 = \{00, 10\}; L_2 = \{0, 1\}$. $L_1 \circ L_2 = \{000, 001, 100, 101\}$
- $L_1 = \text{set of strings ending in 0}; L_2 = \text{set of strings beginning with 01}$. $L_1 \circ L_2 = \text{set of strings containing 001 as a substring}$
Definition
Given languages $L_1$ and $L_2$, we define their \textit{concatenation} to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$

Example
- $L_1 = \{\text{hello}\}$ and $L_2 = \{\text{world}\}$ then $L_1 \circ L_2 = \{\text{helloworld}\}$
- $L_1 = \{00, 10\}; L_2 = \{0, 1\}$. $L_1 \circ L_2 = \{000, 001, 100, 101\}$
- $L_1 = \text{set of strings ending in } 0; L_2 = \text{set of strings beginning with } 01$. $L_1 \circ L_2 = \text{set of strings containing } 001 \text{ as a substring}$
- $L \circ \{\epsilon\} =$
Concatenation of Languages

Definition
Given languages $L_1$ and $L_2$, we define their concatenation to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$

Example
- $L_1 = \{\text{hello}\}$ and $L_2 = \{\text{world}\}$ then $L_1 \circ L_2 = \{\text{helloworld}\}$
- $L_1 = \{00, 10\}; \ L_2 = \{0, 1\}$. $L_1 \circ L_2 = \{000, 001, 100, 101\}$
- $L_1 =$ set of strings ending in 0; $L_2 =$ set of strings beginning with 01. $L_1 \circ L_2 =$ set of strings containing 001 as a substring
- $L \circ \{\varepsilon\} = L$. 
**Concatenation of Languages**

**Definition**
Given languages $L_1$ and $L_2$, we define their *concatenation* to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, \ y \in L_2\}$

**Example**

- $L_1 = \{\text{hello}\}$ and $L_2 = \{\text{world}\}$ then $L_1 \circ L_2 = \{\text{helloworld}\}$
- $L_1 = \{00, 10\}; \ L_2 = \{0, 1\}. \ L_1 \circ L_2 = \{000, 001, 100, 101\}$
- $L_1 = \text{set of strings ending in 0}; \ L_2 = \text{set of strings beginning with 01}. \ L_1 \circ L_2 = \text{set of strings containing 001 as a substring}$
- $L \circ \{\epsilon\} = L. \ L \circ \emptyset = \emptyset$
Definition
Given languages $L_1$ and $L_2$, we define their *concatenation* to be the language $L_1 \circ L_2 = \{xy \mid x \in L_1, \ y \in L_2\}$

Example
- $L_1 = \{\text{hello}\}$ and $L_2 = \{\text{world}\}$ then $L_1 \circ L_2 = \{\text{helloworld}\}$
- $L_1 = \{00, 10\}; \ L_2 = \{0, 1\}$. $L_1 \circ L_2 = \{000, 001, 100, 101\}$
- $L_1$ = set of strings ending in 0; $L_2$ = set of strings beginning with 01. $L_1 \circ L_2$ = set of strings containing 001 as a substring
- $L \circ \{\epsilon\} = L$. $L \circ \emptyset = \emptyset$. 

Concatenation of Languages
Kleene Closure

Definition

\[ L^n = \begin{cases} \\{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]
Kleene Closure

Definition

\[ L^n = \begin{cases} \{ \epsilon \} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i \geq 0 \).
Kleene Closure

Definition

\[
L^n = \begin{cases} 
\{\varepsilon\} & \text{if } n = 0 \\
L^{n-1} \circ L & \text{otherwise}
\end{cases}
\]

i.e., \(L^i\) is \(L \circ L \circ \cdots \circ L\) (concatenation of \(i\) copies of \(L\)), for \(i > 0\).

- If \(L = \{0, 1\}\), then \(L^0 = \)
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \)
Kleene Closure

Definition

\[ L^n = \begin{cases} \{ \epsilon \} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i \geq 0 \).

▶ If \( L = \{0, 1\} \), then \( L^0 = \{ \epsilon \}, L^2 = \)
Kleene Closure

Definition

$$L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases}$$

i.e., $L^i$ is $L \circ L \circ \cdots \circ L$ (concatenation of $i$ copies of $L$), for $i > 0$.

- If $L = \{0, 1\}$, then $L^0 = \{\epsilon\}$, $L^2 = \{00, 01, 10, 11\}$.
Kleene Closure

Definition

\[ L^n = \begin{cases} \{ \epsilon \} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

- If \( L = \{0, 1\} \), then \( L^0 = \{ \epsilon \} \), \( L^2 = \{00, 01, 10, 11\} \).
- \( \emptyset^{0} = \)
Kleene Closure

Definition

\[ L^n = \begin{cases} \{ \epsilon \} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

- If \( L = \{0, 1\} \), then \( L^0 = \{ \epsilon \} \), \( L^2 = \{00, 01, 10, 11\} \).
- \( \emptyset^0 = \{ \epsilon \} \).
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \).

- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \)
Kleene Closure

Definition

\[ L^n = \begin{cases} \{ \epsilon \} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \).
- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \).
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \quad \text{for } L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \).

- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \).
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \quad L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

\( L^* \), the Kleene Closure of \( L \): set of strings formed by taking any number of strings (possibly none) from \( L \), possibly with repetitions and concatenating all of them.

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \).

- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \).
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

\[ L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

\( L^* \), the Kleene Closure of \( L \): set of strings formed by taking any number of strings (possibly none) from \( L \), possibly with repetitions and concatenating all of them.

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \). \( L^* = \)

- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \).
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\varepsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

\[ L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

\( L^* \), the Kleene Closure of \( L \): set of strings formed by taking any number of strings (possibly none) from \( L \), possibly with repetitions and concatenating all of them.

- If \( L = \{0, 1\} \), then \( L^0 = \{\varepsilon\} \), \( L^2 = \{00, 01, 10, 11\} \). \( L^* = \) set of all binary strings (including \( \varepsilon \)).
- \( \emptyset^0 = \{\varepsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \).
Kleene Closure

Definition

\[ L^n = \begin{cases} 
\{\epsilon\} & \text{if } n = 0 \\
L^{n-1} \circ L & \text{otherwise} 
\end{cases} \quad L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

\( L^* \), the Kleene Closure of \( L \): set of strings formed by taking any number of strings (possibly none) from \( L \), possibly with repetitions and concatenating all of them.

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\}, \quad L^2 = \{00, 01, 10, 11\}. \quad L^* = \text{set of all binary strings (including } \epsilon \). \)
- \( \emptyset^0 = \{\epsilon\}. \quad \text{For } i > 0, \quad \emptyset^i = \emptyset. \quad \emptyset^* = \)
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

\[ L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

\( L^* \), the Kleene Closure of \( L \): set of strings formed by taking any number of strings (possibly none) from \( L \), possibly with repetitions and concatenating all of them.

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \). \( L^* \) = set of all binary strings (including \( \epsilon \)).
- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \). \( \emptyset^* = \{\epsilon\} \).
Kleene Closure

Definition

\[ L^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

\[ L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

\( L^* \), the Kleene Closure of \( L \): set of strings formed by taking any number of strings (possibly none) from \( L \), possibly with repetitions and concatenating all of them.

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \). \( L^* = \) set of all binary strings (including \( \epsilon \)).
- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \). \( \emptyset^* = \{\epsilon\} \)
- \( \emptyset \) is one of only two languages whose Kleene closure is finite. Which is the other?
Kleene Closure

Definition

\[ L^n = \begin{cases} \{ \epsilon \} & \text{if } n = 0 \\ L^{n-1} \circ L & \text{otherwise} \end{cases} \]

\[ L^* = \bigcup_{i \geq 0} L^i \]

i.e., \( L^i \) is \( L \circ L \circ \cdots \circ L \) (concatenation of \( i \) copies of \( L \)), for \( i > 0 \).

\( L^* \), the Kleene Closure of \( L \): set of strings formed by taking any number of strings (possibly none) from \( L \), possibly with repetitions and concatenating all of them.

- If \( L = \{0, 1\} \), then \( L^0 = \{\epsilon\} \), \( L^2 = \{00, 01, 10, 11\} \). \( L^* = \) set of all binary strings (including \( \epsilon \)).
- \( \emptyset^0 = \{\epsilon\} \). For \( i > 0 \), \( \emptyset^i = \emptyset \). \( \emptyset^* = \{\epsilon\} \)
- \( \emptyset \) is one of only two languages whose Kleene closure is finite. Which is the other? \( \{\epsilon\}^* = \{\epsilon\} \).
A regular expression is a formula for representing a (complex) language in terms of “elementary” languages combined using the three operations union, concatenation and Kleene closure.
Syntax and Semantics

A regular expression over an alphabet $\Sigma$ is of one of the following forms:
Syntax and Semantics

A regular expression over an alphabet $\Sigma$ is of one of the following forms:

$$\emptyset$$
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet $\Sigma$ is of one of the following forms:

- $\emptyset$
- $\epsilon$
- $R_1 \cup R_2$
- $R_1 \circ R_2$
- $R_1^*$
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet \( \Sigma \) is of one of the following forms:

- \( \emptyset \)
- \( \epsilon \)
- \( a \)
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet \( \Sigma \) is of one of the following forms:

\[
\emptyset \\
\epsilon \\
a
\]

Basis
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet \( \Sigma \) is of one of the following forms:

- \( \emptyset \)
- \( \epsilon \)
- \( a \)
- \( (R_1 \cup R_2) \)
Syntax and Semantics

A regular expression over an alphabet \( \Sigma \) is of one of the following forms:

- \( \emptyset \)
- \( \epsilon \)
- \( a \)
- \( (R_1 \cup R_2) \)
- \( (R_1 \circ R_2) \)
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet $\Sigma$ is of one of the following forms:

- $\emptyset$
- $\varepsilon$
- $a$
- $(R_1 \cup R_2)$
- $(R_1 \circ R_2)$
- $(R_1^*)$
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet $\Sigma$ is of one of the following forms:

$\emptyset$

$\epsilon$

$a$

$(R_1 \cup R_2)$

$(R_1 \circ R_2)$

$(R_1^*)$
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet $\Sigma$ is of one of the following forms:

- **Syntax**
  - $\emptyset$
  - $\varepsilon$
  - $a$

- **Basis**
  - $(R_1 \cup R_2)$
  - $(R_1 \circ R_2)$
  - $(R_1^*)$
## Regular Expressions
### Formal Inductive Definition

### Syntax and Semantics

A regular expression over an alphabet $\Sigma$ is of one of the following forms:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$L(\emptyset) = {}$</td>
</tr>
<tr>
<td>Basis</td>
<td></td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>$L(\epsilon) = {\epsilon}$</td>
</tr>
<tr>
<td>$a$</td>
<td>$L(a) = {a}$</td>
</tr>
<tr>
<td>Induction</td>
<td></td>
</tr>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td>$L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$</td>
</tr>
<tr>
<td>$(R_1 \circ R_2)$</td>
<td>$L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$</td>
</tr>
<tr>
<td>$(R_1^*)$</td>
<td>$L(R_1^<em>) = L(R_1)^</em>$</td>
</tr>
</tbody>
</table>
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet $\Sigma$ is of one of the following forms:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$L(\emptyset) = {}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td></td>
</tr>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td></td>
</tr>
<tr>
<td>$(R_1 \circ R_2)$</td>
<td></td>
</tr>
<tr>
<td>$(R_1^*)$</td>
<td></td>
</tr>
</tbody>
</table>

Basis

Induction
## Syntax and Semantics

A regular expression over an alphabet \( \Sigma \) is of one of the following forms:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( L(\emptyset) = {} )</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>( L(\epsilon) = { \epsilon } )</td>
</tr>
<tr>
<td>( a )</td>
<td></td>
</tr>
<tr>
<td>( R_1 \cup R_2 )</td>
<td></td>
</tr>
<tr>
<td>( R_1 \circ R_2 )</td>
<td></td>
</tr>
<tr>
<td>( R_1^* )</td>
<td></td>
</tr>
</tbody>
</table>
## Syntax and Semantics

A regular expression over an alphabet $\Sigma$ is of one of the following forms:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$L(\emptyset) = {}$</td>
</tr>
<tr>
<td>Basis</td>
<td></td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>$L(\epsilon) = {\epsilon}$</td>
</tr>
<tr>
<td>$a$</td>
<td>$L(a) = {a}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Induction</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td></td>
</tr>
<tr>
<td>$(R_1 \circ R_2)$</td>
<td></td>
</tr>
<tr>
<td>$(R_1^*)$</td>
<td></td>
</tr>
</tbody>
</table>
Regular Expressions
Formal Inductive Definition

Syntax and Semantics
A regular expression over an alphabet $\Sigma$ is of one of the following forms:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$L(\emptyset) = {}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>$L(\epsilon) = {\epsilon}$</td>
</tr>
<tr>
<td>$a$</td>
<td>$L(a) = {a}$</td>
</tr>
</tbody>
</table>

Basis

Induction

$(R_1 \cup R_2)$
$L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$

$(R_1 \circ R_2)$

$(R_1^*)$
**Regular Expressions**

**Formal Inductive Definition**

**Syntax and Semantics**

A regular expression over an alphabet $\Sigma$ is of one of the following forms:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$L(\emptyset) = {} \quad \text{(Basis)}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>$L(\epsilon) = {\epsilon}$</td>
</tr>
<tr>
<td>$a$</td>
<td>$L(a) = {a}$</td>
</tr>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td>$L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$ (Induction)</td>
</tr>
<tr>
<td>$(R_1 \circ R_2)$</td>
<td>$L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$</td>
</tr>
<tr>
<td>$(R_1^*)$</td>
<td>$L((R_1^<em>)) = L(R_1)^</em>$</td>
</tr>
</tbody>
</table>
## Syntax and Semantics

A regular expression over an alphabet $\Sigma$ is of one of the following forms:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$L(\emptyset) = {}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>$L(\epsilon) = {\epsilon}$</td>
</tr>
<tr>
<td>$a$</td>
<td>$L(a) = {a}$</td>
</tr>
</tbody>
</table>

### Basis

- $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
- $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$
- $L((R_1^*)) = L(R_1)^*$
Notational Conventions
Removing the brackets

To avoid cluttering of parenthesis, we adopt the following conventions.
Notational Conventions
Removing the brackets

To avoid cluttering of parenthesis, we adopt the following conventions.

▶ Precedence: *, ◦, ∪. For example, $R \cup S^* \circ T$ means
Notational Conventions
Removing the brackets

To avoid cluttering of parenthesis, we adopt the following conventions.

- **Precedence**: $\ast, \circ, \cup$. For example, $R \cup S^* \circ T$ means $(R \cup ((S^*) \circ T))$
To avoid cluttering of parenthesis, we adopt the following conventions.

- **Precedence**: $\ast, \circ, \cup$. For example, $R \cup S^* \circ T$ means $(R \cup ((S^*) \circ T))$

- **Associativity**: $(R \cup (S \cup T)) = ((R \cup S) \cup T) = R \cup S \cup T$
  and $(R \circ (S \circ T)) = ((R \circ S) \circ T) = R \circ S \circ T$. 
Notational Conventions

Removing the brackets

To avoid cluttering of parenthesis, we adopt the following conventions.

- **Precedence**: $\times, \circ, \cup$. For example, $R \cup S^* \circ T$ means $(R \cup ((S^*) \circ T))$

- **Associativity**: $(R \cup (S \cup T)) = ((R \cup S) \cup T) = R \cup S \cup T$
  and $(R \circ (S \circ T)) = ((R \circ S) \circ T) = R \circ S \circ T$.

Also will sometimes omit $\circ$: e.g. will write $RS$ instead of $R \circ S$
Regular Expression Examples

\[ R \]

\[ L(R) \]
Regular Expression Examples

\[ R \]

\[ (0 \cup 1)^* \]

\[ L(R) \]
Regular Expression Examples

\[ R \]

\[ (0 \cup 1)^* \]

\[ L(R) \]

\[ = (\{0\} \cup \{1\})^* = \{0, 1\}^* \]
Regular Expression Examples

\[ R \]

\[(0 \cup 1)^*\]

\[ L(R) \]

\[= (\{0\} \cup \{1\})^* = \{0, 1\}^*\]
Regular Expression Examples

\[ R \]

\[(0 \cup 1)^*\]

\[ L(R) \]

\[(\{0\} \cup \{1\})^* = \{0, 1\}^*\]
Regular Expression Examples

\[ R \]

\[ (0 \cup 1)^* \]

\[ 0\emptyset \]

\[ 0^* \cup (0^*10^*10^*10^*)^* \]

\[ L(R) \]

\[ = (\{0\} \cup \{1\})^* = \{0, 1\}^* \]

\[ \emptyset \]
Regular Expression Examples

\[ R \]

\[(0 \cup 1)^* \]

\[ 0\emptyset \]

\[ 0^* \cup (0^*10^*10^*10^*)^* \]

\[ L(R) \]

\[ = (\{0\} \cup \{1\})^* = \{0, 1\}^* \]

\[ \emptyset \]

Strings where the number of 1s is divisible by 3
Regular Expression Examples

\[ R \]
\[
(0 \cup 1)^* \\
0\emptyset \\
0^* \cup (0^*10^*10^*10^*)^* \\
(0 \cup 1)^*001(0 \cup 1)^* \\
\]

\[ L(R) \]
\[
= (\{0\} \cup \{1\})^* = \{0, 1\}^* \\
\emptyset \\
\text{Strings where the number of 1s is divisible by 3} \\
\]
Regular Expression Examples

\[ R \]

\((0 \cup 1)^*\)

\(0\emptyset\)

\(0^* \cup (0*10*10*10*)^*\)

\((0 \cup 1)^*001(0 \cup 1)^*\)

\[ L(R) \]

\(= (\{0\} \cup \{1\})^* = \{0, 1\}^*\)

\(\emptyset\)

Strings where the number of 1s is divisible by 3

Strings that have 001 as a substring
More Examples

$$R$$

$$L(R)$$
More Examples

\[ R \]

\[ L(R) \]

\[(10)^* \cup (01)^* \cup 0(10)^* \cup 1(01)^* \]
## More Examples

<table>
<thead>
<tr>
<th>$R$</th>
<th>$L(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(10)^* \cup (01)^* \cup 0(10)^* \cup 1(01)^*$</td>
<td>Strings that consist of alternating 0s and 1s</td>
</tr>
</tbody>
</table>
More Examples

<table>
<thead>
<tr>
<th>$R$</th>
<th>$L(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(10)^* \cup (01)^* \cup 0(10)^* \cup 1(01)^*$</td>
<td>Strings that consist of alternating 0s and 1s</td>
</tr>
<tr>
<td>$(\epsilon \cup 1)(01)^*(\epsilon \cup 0)$</td>
<td></td>
</tr>
</tbody>
</table>
More Examples

\[ R \]

\[(10)^* \cup (01)^* \cup 0(10)^* \cup 1(01)^* \]

\[(\epsilon \cup 1)(01)^*(\epsilon \cup 0)\]

\[ L(R) \]

Strings that consist of alternating 0s and 1s

Strings that consist of alternating 0s and 1s
More Examples

**R**

\[(10)^* \cup (01)^* \cup 0(10)^* \cup 1(01)^*\]

\[(\epsilon \cup 1)(01)^*(\epsilon \cup 0)\]

\[(0 \cup \epsilon)(1 \cup 10)^*\]

**L(R)**

Strings that consist of alternating 0s and 1s

Strings that consist of alternating 0s and 1s
### More Examples

<table>
<thead>
<tr>
<th>$R$</th>
<th>$L(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(10)^* \cup (01)^* \cup 0(10)^* \cup 1(01)^*$</td>
<td>Strings that consist of alternating 0s and 1s</td>
</tr>
<tr>
<td>$(\epsilon \cup 1)(01)^*(\epsilon \cup 0)$</td>
<td>Strings that consist of alternating 0s and 1s</td>
</tr>
<tr>
<td>$(0 \cup \epsilon)(1 \cup 10)^*$</td>
<td>Strings that do not have two consecutive 0s</td>
</tr>
</tbody>
</table>
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$. 

▶ Commutativity: $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)

▶ Associativity: $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$

▶ Distributivity: $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$

▶ Concatenating with $\epsilon$: $R \circ \epsilon = \epsilon \circ R = R$

▶ Concatenating with $\emptyset$: $R \circ \emptyset = \emptyset \circ R = \emptyset$ if $\epsilon \in L(R)$

▶ $(R \ast) \ast = R \ast$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$
Some Regular Expression Identities

We say \( R_1 = R_2 \) if \( L(R_1) = L(R_2) \).

- **Commutativity:** \( R_1 \cup R_2 = R_2 \cup R_1 \) (but \( R_1 \circ R_2 \neq R_2 \circ R_1 \) typically)

- **Distributivity:** \( R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2 \) and \( (R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R \)

- **Concatenating with \( \epsilon \):** \( R \circ \epsilon = \epsilon \circ R = R \)

- **Concatenating with \( \emptyset \):** \( R \circ \emptyset = \emptyset \circ R = \emptyset \)

- **\( R \cup \emptyset = R, \; R \cup \epsilon = R \) if \( \epsilon \in L(R) \)**
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)

- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)
- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)

- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$

- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$

- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)
- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$
- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$
- **Concatenating with $\emptyset$:** $R \circ \emptyset = \emptyset \circ R =$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)

- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$

- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$

- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$

- **Concatenating with $\emptyset$:** $R \circ \emptyset = \emptyset \circ R = \emptyset$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)

- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$

- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$

- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$

- **Concatenating with $\emptyset$:** $R \circ \emptyset = \emptyset \circ R = \emptyset$

- $R \cup \emptyset = R$. 
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)
- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$
- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$
- **Concatenating with $\emptyset$:** $R \circ \emptyset = \emptyset \circ R = \emptyset$
- $R \cup \emptyset = R$. $R \cup \epsilon = R$ iff $\epsilon \in L(R)$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity**: $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)
- **Associativity**: $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- **Distributivity**: $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$
- **Concatenating with $\epsilon$**: $R \circ \epsilon = \epsilon \circ R = R$
- **Concatenating with $\emptyset$**: $R \circ \emptyset = \emptyset \circ R = \emptyset$
- $R \cup \emptyset = R$. $R \cup \epsilon = R$ iff $\epsilon \in L(R)$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)
- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$
- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$
- **Concatenating with $\emptyset$:** $R \circ \emptyset = \emptyset \circ R = \emptyset$
- $R \cup \emptyset = R$. $R \cup \epsilon = R$ iff $\epsilon \in L(R)$
- $(R^*)^* = \epsilon$
Some Regular Expression Identities

We say \( R_1 = R_2 \) if \( L(R_1) = L(R_2) \).

- **Commutativity**: \( R_1 \cup R_2 = R_2 \cup R_1 \) (but \( R_1 \circ R_2 \neq R_2 \circ R_1 \) typically)

- **Associativity**: \( (R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3) \) and \( (R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3) \)

- **Distributivity**: \( R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2 \) and \( (R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R \)

- **Concatenating with \( \epsilon \)**: \( R \circ \epsilon = \epsilon \circ R = R \)

- **Concatenating with \( \emptyset \)**: \( R \circ \emptyset = \emptyset \circ R = \emptyset \)

- \( R \cup \emptyset = R \). \( R \cup \epsilon = R \) iff \( \epsilon \in L(R) \)

- \((R^*)^* = R^*\)
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)
- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$
- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$
- **Concatenating with $\emptyset$:** $R \circ \emptyset = \emptyset \circ R = \emptyset$
- $R \cup \emptyset = R$. $R \cup \epsilon = R$ iff $\epsilon \in L(R)$
- $(R^*)^* = R^*$
- $\emptyset^* =$
Some Regular Expression Identities

We say $R_1 = R_2$ if $L(R_1) = L(R_2)$.

- **Commutativity:** $R_1 \cup R_2 = R_2 \cup R_1$ (but $R_1 \circ R_2 \neq R_2 \circ R_1$ typically)
- **Associativity:** $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$ and $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$
- **Distributivity:** $R \circ (R_1 \cup R_2) = R \circ R_1 \cup R \circ R_2$ and $(R_1 \cup R_2) \circ R = R_1 \circ R \cup R_2 \circ R$
- **Concatenating with $\epsilon$:** $R \circ \epsilon = \epsilon \circ R = R$
- **Concatenating with $\emptyset$:** $R \circ \emptyset = \emptyset \circ R = \emptyset$
- $R \cup \emptyset = R$. $R \cup \epsilon = R$ iff $\epsilon \in L(R)$
- $(R^*)^* = R^*$
- $\emptyset^* = \epsilon$
Useful Notation

Definition
Define $R^+ = RR^*$. 
Useful Notation

Definition
Define $R^+ = RR^*$. Thus, $R^* = R^+ \cup \epsilon$. 
Definition
Define \( R^+ = RR^* \). Thus, \( R^* = R^+ \cup \epsilon \). In addition, \( R^+ = R^* \) iff \( \epsilon \in L(R) \).
Regular Expressions and Regular Languages

Why do they have such similar names?

Theorem

\[ L \text{ is a regular language if and only if there is a regular expression } R \]

such that

\[ L(R) = L \]

i.e., Regular expressions have the same “expressive power” as finite automata.

Proof.

\[ \begin{align*}
\text{Given regular expression } & R, \text{ will construct NFA } N \text{ such that } L(N) = L(R) \\
\text{Given DFA } & M, \text{ will construct regular expression } R \text{ such that } L(M) = L(R)
\end{align*} \]

\[ \square \]
Regular Expressions and Regular Languages

Why do they have such similar names?

Theorem

$L$ is a regular language if and only if there is a regular expression $R$ such that $L(R) = L$
Theorem

$L$ is a regular language if and only if there is a regular expression $R$ such that $L(R) = L$

i.e., Regular expressions have the same “expressive power” as finite automata.
Regular Expressions and Regular Languages

Why do they have such similar names?

Theorem

$L$ is a regular language if and only if there is a regular expression $R$ such that $L(R) = L$

i.e., Regular expressions have the same “expressive power” as finite automata.

Proof.
Regular Expressions and Regular Languages
Why do they have such similar names?

Theorem
$L$ is a regular language if and only if there is a regular expression $R$ such that $L(R) = L$
i.e., Regular expressions have the same “expressive power” as finite automata.

Proof.
- Given regular expression $R$, will construct NFA $N$ such that $L(N) = L(R)$
Regular Expressions and Regular Languages

Why do they have such similar names?

Theorem

$L$ is a regular language if and only if there is a regular expression $R$ such that $L(R) = L$

i.e., Regular expressions have the same “expressive power” as finite automata.

Proof.

- Given regular expression $R$, will construct NFA $N$ such that $L(N) = L(R)$
- Given DFA $M$, will construct regular expression $R$ such that $L(M) = L(R)$
Lemma
For any regex $R$, there is an NFA $N_R$ s.t. $L(N_R) = L(R)$.

Proof Idea
We will build the NFA $N_R$ for $R$, inductively, based on the number of operators in $R$, $\#(R)$. 
Lemma

For any regex $R$, there is an NFA $N_R$ s.t. $L(N_R) = L(R)$.

Proof Idea

We will build the NFA $N_R$ for $R$, inductively, based on the number of operators in $R$, $\#(R)$.

- **Base Case:** $\#(R) = 0$ means that $R$ is $\emptyset$, $\epsilon$, or $a$ (from some $a \in \Sigma$). We will build NFAs for these cases.
Lemma
For any regex $R$, there is an NFA $N_R$ s.t. $L(N_R) = L(R)$.

Proof Idea
We will build the NFA $N_R$ for $R$, inductively, based on the number of operators in $R$, $\#(R)$.

- **Base Case:** $\#(R) = 0$ means that $R$ is $\emptyset$, $\epsilon$, or $a$ (from some $a \in \Sigma$). We will build NFAs for these cases.
- **Induction Hypothesis:** Assume that for regular expressions $R$, with $\#(R) \leq n$, there is an NFA $N_R$ s.t. $L(N_R) = L(R)$. 
Lemma
For any regex $R$, there is an NFA $N_R$ s.t. $L(N_R) = L(R)$.

Proof Idea
We will build the NFA $N_R$ for $R$, inductively, based on the number of operators in $R$, $\#(R)$.

- **Base Case:** $\#(R) = 0$ means that $R$ is $\emptyset$, $\epsilon$, or $a$ (from some $a \in \Sigma$). We will build NFAs for these cases.

- **Induction Hypothesis:** Assume that for regular expressions $R$, with $\#(R) \leq n$, there is an NFA $N_R$ s.t. $L(N_R) = L(R)$.

- **Induction Step:** Consider $R$ with $\#(R) = n + 1$. Based on the form of $R$, the NFA $N_R$ will be built using the induction hypothesis.
Regular Expression to NFA

Base Cases
If $R$ is an elementary regular expression, NFA $N_R$ is constructed as follows.
Base Cases
If $R$ is an elementary regular expression, NFA $N_R$ is constructed as follows.

$$R = \emptyset$$
Regular Expression to NFA

Base Cases
If $R$ is an elementary regular expression, NFA $N_R$ is constructed as follows.

$R = \emptyset$

\[ q_0 \]
Regular Expression to NFA

Base Cases
If $R$ is an elementary regular expression, NFA $N_R$ is constructed as follows.

\[ R = \emptyset \]

\[ R = \epsilon \]
Base Cases
If $R$ is an elementary regular expression, NFA $N_R$ is constructed as follows.

- $R = \emptyset$
- $R = \epsilon$
Base Cases
If $R$ is an elementary regular expression, NFA $N_R$ is constructed as follows.

- If $R = \emptyset$

  ![Diagram for $R = \emptyset$]

- If $R = \epsilon$

  ![Diagram for $R = \epsilon$]

- If $R = a$

  ![Diagram for $R = a$]
Regular Expression to NFA

Base Cases
If $R$ is an elementary regular expression, NFA $N_R$ is constructed as follows.

- $R = \emptyset$
  - $q_0$ as start state

- $R = \epsilon$
  - $q_0$ as start state

- $R = a$
  - Transition from $q_0$ to $q_1$ on input $a$
Induction Step: Union

Case $R = R_1 \cup R_2$
Induction Step: Union

Case $R = R_1 \cup R_2$

By induction hypothesis, there are $N_1, N_2$ s.t.
$L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$.
Induction Step: Union

Case $R = R_1 \cup R_2$

By induction hypothesis, there are $N_1, N_2$ s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$. Build NFA $N$ s.t. $L(N) = L(N_1) \cup L(N_2)$
Induction Step: Union

Case $R = R_1 \cup R_2$

By induction hypothesis, there are $N_1, N_2$ s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$. Build NFA $N$ s.t. $L(N) = L(N_1) \cup L(N_2)$.

![Diagram](image-url)
Induction Step: Union

Formal Definition

Case $R = R_1 \cup R_2$

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ (with $Q_1 \cap Q_2 = \emptyset$) such that $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$. The NFA $N = (Q, \Sigma, \delta, q_0, F)$ is given by

- $Q = Q_1 \cup Q_2 \cup \{q_0\}$, where $q_0 \notin Q_1 \cup Q_2$
- $F = F_1 \cup F_2$
- $\delta$ is defined as follows

$$
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \in Q_1 \\
\delta_2(q, a) & \text{if } q \in Q_2 \\
\{q_1, q_2\} & \text{if } q = q_0 \text{ and } a = \epsilon \\
\emptyset & \text{otherwise}
\end{cases}
$$
Induction Step: Union
Correctness Proof

Need to show that $w \in L(N)$ iff $w \in L(N_1) \cup L(N_2)$.

$\Rightarrow$ $w \in L(N)$ implies $q_0 \xrightarrow{w} N q$ for some $q \in F$. 
Induction Step: Union

Correctness Proof

Need to show that $w \in L(N)$ iff $w \in L(N_1) \cup L(N_2)$.

$\Rightarrow$ $w \in L(N)$ implies $q_0 \xrightarrow{w} N q$ for some $q \in F$. Based on the transitions out of $q_0$, $q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q$ or $q_0 \xrightarrow{\epsilon} N q_2 \xrightarrow{w} N q$. 

$\Leftarrow$ $w \in L(N_1) \cup L(N_2)$. Consider $w \in L(N_1)$; case of $w \in L(N_2)$ is similar. Then, $q_1 \xrightarrow{w} N q$ for some $q \in F_1$. Thus, $q_0 \xrightarrow{w} N q$, and $q \in F$. This means that $w \in L(N)$. 
Induction Step: Union
Correctness Proof

Need to show that \( w \in L(N) \) iff \( w \in L(N_1) \cup L(N_2) \).

\[ \Rightarrow \] \( w \in L(N) \) implies \( q_0 \xrightarrow{w} q \) for some \( q \in F \). Based on the transitions out of \( q_0 \), \( q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{w} q \) or \( q_0 \xrightarrow{\epsilon} q_2 \xrightarrow{w} q \). Consider \( q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{w} q \). (Other case is similar)
Need to show that \( w \in L(N) \) iff \( w \in L(N_1) \cup L(N_2) \).

\[ w \in L(N) \Rightarrow q_0 \xrightarrow{w} N q \text{ for some } q \in F. \]

Based on the transitions out of \( q_0 \), \( q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q \) or \( q_0 \xrightarrow{\epsilon} N q_2 \xrightarrow{w} N q \). Consider \( q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q \). (Other case is similar) This means \( q_1 \xrightarrow{w} N_1 q \) (as \( N \) has the same transition as \( N_1 \) on the states in \( Q_1 \)) and \( q \in F_1 \). This means \( w \in L(N_1) \).
Induction Step: Union
Correctness Proof

Need to show that $w \in L(N)$ iff $w \in L(N_1) \cup L(N_2)$.

$\Rightarrow$ $w \in L(N)$ implies $q_0 \xrightarrow{w} N q$ for some $q \in F$. Based on the transitions out of $q_0$, $q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q$ or $q_0 \xrightarrow{\epsilon} N q_2 \xrightarrow{w} N q$. Consider $q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q$. (Other case is similar) This means $q_1 \xrightarrow{w} N_1 q$ (as $N$ has the same transition as $N_1$ on the states in $Q_1$) and $q \in F_1$. This means $w \in L(N_1)$.

$\Leftarrow$ $w \in L(N_1) \cup L(N_2)$. Consider $w \in L(N_1)$; case of $w \in L(N_2)$ is similar.
Induction Step: Union
Correctness Proof

Need to show that \( w \in L(N) \) iff \( w \in L(N_1) \cup L(N_2) \).

\[ \Rightarrow w \in L(N) \text{ implies } q_0 \xrightarrow{w}_N q \text{ for some } q \in F. \] Based on the transitions out of \( q_0 \), \( q_0 \xrightarrow{\epsilon}_N q_1 \xrightarrow{w}_N q \) or \( q_0 \xrightarrow{\epsilon}_N q_2 \xrightarrow{w}_N q \). Consider \( q_0 \xrightarrow{\epsilon}_N q_1 \xrightarrow{w}_N q \). (Other case is similar) This means \( q_1 \xrightarrow{w}_{N_1} q \) (as \( N \) has the same transition as \( N_1 \) on the states in \( Q_1 \)) and \( q \in F_1 \). This means \( w \in L(N_1) \).

\[ \Leftarrow w \in L(N_1) \cup L(N_2). \text{ Consider } w \in L(N_1); \text{ case of } w \in L(N_2) \text{ is similar. Then, } q_1 \xrightarrow{w}_{N_1} q \text{ for some } q \in F_1. \]
Induction Step: Union
Correctness Proof

Need to show that \( w \in L(N) \) iff \( w \in L(N_1) \cup L(N_2) \).

\( \Rightarrow \) \( w \in L(N) \) implies \( q_0 \xrightarrow{w} N q \) for some \( q \in F \). Based on the transitions out of \( q_0 \), \( q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q \) or \( q_0 \xrightarrow{\epsilon} N q_2 \xrightarrow{w} N q \). Consider \( q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q \). (Other case is similar) This means \( q_1 \xrightarrow{w} N_1 q \) (as \( N \) has the same transition as \( N_1 \) on the states in \( Q_1 \)) and \( q \in F_1 \). This means \( w \in L(N_1) \).

\( \Leftarrow \) \( w \in L(N_1) \cup L(N_2) \). Consider \( w \in L(N_1) \); case of \( w \in L(N_2) \) is similar. Then, \( q_1 \xrightarrow{w} N_1 q \) for some \( q \in F_1 \). Thus, \( q_0 \xrightarrow{\epsilon} N q_1 \xrightarrow{w} N q \), and \( q \in F \). This means that \( w \in L(N) \).
Induction Step: Concatenation

Case $R = R_1 \circ R_2$
Induction Step: Concatenation

Case $R = R_1 \circ R_2$

- By induction hypothesis, there are $N_1, N_2$ s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$
Induction Step: Concatenation

Case $R = R_1 \circ R_2$

- By induction hypothesis, there are $N_1, N_2$ s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$
- Build NFA $N$ s.t. $L(N) = L(N_1) \circ L(N_2)$
Induction Step: Concatenation

Case $R = R_1 \circ R_2$

- By induction hypothesis, there are $N_1, N_2$ s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$
- Build NFA $N$ s.t. $L(N) = L(N_1) \circ L(N_2)$
Induction Step: Concatenation

Case $R = R_1 \circ R_2$

- By induction hypothesis, there are $N_1, N_2$ s.t. $L(N_1) = L(R_1)$ and $L(N_2) = L(R_2)$
- Build NFA $N$ s.t. $L(N) = L(N_1) \circ L(N_2)$

Formal definition and proof of correctness left as exercise.
Induction Step: Kleene Closure

Case $R = R_1^*$
Induction Step: Kleene Closure

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
Induction Step: Kleene Closure

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$
Induction Step: Kleene Closure

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$
Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$

Problem: May not accept $\epsilon$! One can show that $L(N) = (L(N_1))^+$. 

\[
\begin{array}{c}
q_0 \\
\rightarrow \\
\epsilon \\
q_1 \\
\epsilon \\
q_2
\end{array}
\]
Induction Step: Kleene Closure

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$

![Diagram of NFA](image)
Induction Step: Kleene Closure

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$
Induction Step: Kleene Closure

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$

![Diagram of NFA](image-url)
Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$

Problem: May accept strings that are not in $(L(N_1))^*$!
Example demonstrating the problem

Example NFA $N$

Incorrect Kleene Closure of $L(N) = (0 \cup 1)^* 1 (0 \cup 1)^*$.
Thus, $(L(N))^* = \epsilon \cup (0 \cup 1)^* 1 (0 \cup 1)^*$.

The previous construction gives an NFA that accepts $0 \not\in (L(N))^*$!
Example demonstrating the problem

Example NFA $N$

$L(N) = (0 \cup 1)^*1(0 \cup 1)^*$.
Example demonstrating the problem

Example NFA $N$

$L(N) = (0 \cup 1)^*1(0 \cup 1)^*$. Thus, $(L(N))^* = \epsilon \cup (0 \cup 1)^*1(0 \cup 1)^*$. 
Example demonstrating the problem

Example NFA $N$

Incorrect Kleene Closure of $N$

$L(N) = (0 \cup 1)^*1(0 \cup 1)^*$. Thus, $(L(N))^* = \epsilon \cup (0 \cup 1)^*1(0 \cup 1)^*$. 
Example demonstrating the problem

Example NFA $N$

Incorrect Kleene Closure of $N$

$L(N) = (0 \cup 1)^*1(0 \cup 1)^*$. Thus, $(L(N))^* = \varepsilon \cup (0 \cup 1)^*1(0 \cup 1)^*$. The previous construction, gives an NFA that accepts $0 \not\in (L(N))^*$!
Induction Step: Kleene Closure
Correct Construction

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$

Formal definition and proof of correctness left as exercise.
Induction Step: Kleene Closure
Correct Construction

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$

![NFA Diagram]

Formal definition and proof of correctness left as exercise.
Induction Step: Kleene Closure
Correct Construction

Case $R = R_1^*$

- By induction hypothesis, there is $N_1$ s.t. $L(N_1) = L(R_1)$
- Build NFA $N$ s.t. $L(N) = (L(N_1))^*$

Formal definition and proof of correctness left as exercise.
Regular Expressions to NFA

To Summarize

We built an NFA $N_R$ for each regular expression $R$ inductively.
Regular Expressions to NFA

To Summarize

We built an NFA $N_R$ for each regular expression $R$ inductively

▶ When $R$ was an elementary regular expression, we gave an explicit construction of an NFA recognizing $L(R)$
Regular Expressions to NFA

To Summarize

We built an NFA $N_R$ for each regular expression $R$ inductively

- When $R$ was an elementary regular expression, we gave an explicit construction of an NFA recognizing $L(R)$
- When $R = R_1 \oplus R_2$ (or $R = \text{op}(R_1)$), we constructed an NFA $N$ for $R$, using the NFAs for $R_1$ and $R_2$. 
Regular Expressions to NFA

An Example

Build NFA for \((1 \cup 01)^*\)
Regular Expressions to NFA

An Example

Build NFA for \((1 \cup 01)^*\)

\(N_0\)
Regular Expressions to NFA

An Example

Build NFA for \((1 \cup 01)^*\)

\[ N_0 \quad \xrightarrow{0} \quad \text{loop} \]
Build NFA for \((1 \cup 01)^*\)
Regular Expressions to NFA
An Example

Build NFA for \((1 \cup 01)^*\)

\[ N_0 \quad \xrightarrow{0} \quad N_0 \quad \xrightarrow{1} \quad N_1 \]

\[ N_1 \]
Regular Expressions to NFA

An Example

Build NFA for \((1 \cup 01)^*\)

\[ \begin{align*}
N_0 & \quad \xrightarrow{0} \\
N_1 & \quad \xrightarrow{1}
\end{align*} \]
Regular Expressions to NFA

An Example

Build NFA for \((1 \cup 01)^*\)

\[ N_0 \quad \xrightarrow{0} \quad \text{circle} \quad \xrightarrow{1} \quad \text{circle} \]

\[ N_1 \quad \xrightarrow{0} \quad \text{circle} \quad \xrightarrow{\epsilon} \quad \text{circle} \quad \xrightarrow{1} \quad \text{circle} \]
Regular Expressions to NFA

An Example

Build NFA for \((1 \cup 01)^*\)

\(N_0\)

[Diagram]

\(N_1\)

[Diagram]

\(N_{01}\)

[Diagram]

\(N_{1\cup 01}\)
Regular Expressions to NFA

An Example

Build NFA for \((1 \cup 01)^*\)

\[ N_0 \]

\[ N_1 \]

\[ N_{01} \]

\[ N_{1\cup01} \]
Build NFA for \((1 \cup 01)^*\)
Example Continued

Build NFA for \((1 \cup 01)^*\)

\(N_{(1 \cup 01)^*}\)
Build NFA for \((1 \cup 01)^*\)

\[ N_{(1 \cup 01)^*} \]
Today

- Defined Regular Expressions

Syntax: what a regex is built out of — $\emptyset$, $\epsilon$, characters in $\Sigma$, and operators $\cup$, $\circ$, $\ast$.

Semantics: what language a regex stands for.

Expressive power of regular expressions: can express (any and only) regular languages

Today: Languages represented by regular expressions are regular (we showed how to build NFAs for them).

Coming up: Regular languages can be represented by regular expressions (by building regex for any given DFA).
Today

- Defined **Regular Expressions**
  - Syntax: what a regex is built out of — $\emptyset$, $\epsilon$, characters in $\Sigma$, and operators $\cup$, $\circ$, $\ast$.
Today

- Defined **Regular Expressions**
  - Syntax: what a regex is built out of — $\emptyset$, $\epsilon$, characters in $\Sigma$, and operators $\cup$, $\circ$, $\ast$.
  - Semantics: what language a regex stands for.
Today

- Defined Regular Expressions
  - Syntax: what a regex is built out of — $\emptyset$, $\epsilon$, characters in $\Sigma$, and operators $\cup$, $\circ$, $\ast$.
  - Semantics: what language a regex stands for.
- Expressive power of regular expressions: can express (any and only) regular languages
Today

- Defined Regular Expressions
  - Syntax: what a regex is built out of — $\emptyset$, $\epsilon$, characters in $\Sigma$, and operators $\cup$, $\circ$, $\ast$.
  - Semantics: what language a regex stands for.
- Expressive power of regular expressions: can express (any and only) regular languages
  - Today: Languages represented by regular expressions are regular (we showed how to build NFAs for them).
Today

- Defined Regular Expressions
  - Syntax: what a regex is built out of — $\emptyset$, $\epsilon$, characters in $\Sigma$, and operators $\cup$, $\circ$, $\ast$.
  - Semantics: what language a regex stands for.
- Expressive power of regular expressions: can express (any and only) regular languages
  - Today: Languages represented by regular expressions are regular (we showed how to build NFAs for them).
  - Coming up: Regular languages can be represented by regular expressions (by building regex for any given DFA).