Lesson 2. Regular expressions

CSIE 3110 – Formal Languages and Automata Theory

Tony Tan
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Table of contents

Table of contents

Regular expressions

(Def.) Let Σ be an alphabet.

Regular expressions (over Σ) are expressions built inductively as follows.

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

Regular expressions

(Def.) Let Σ be an alphabet.

Regular expressions (over Σ) are expressions built inductively as follows.

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

Regular expression is usually abbreviated as *regex*.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$
- $\emptyset a$ is a regular expression.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$
- $\emptyset a$ is a regular expression.
- $(ab)a$ is a regular expression.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$
- $\emptyset a$ is a regular expression.
- $(ab)a$ is a regular expression. $\Longrightarrow$ usually written as aba

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$
- $\emptyset a$ is a regular expression.
- $(ab)a$ is a regular expression. $\Longrightarrow$ usually written as aba
- $(ab)^*$ is a regular expression.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$
- $\emptyset a$ is a regular expression.
- $(ab)a$ is a regular expression. $\Longrightarrow$ usually written as aba
- $(ab)^*$ is a regular expression.
- $((ab)^*)^*$ is a regular expression.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$
- $\emptyset a$ is a regular expression.
- $(ab)a$ is a regular expression. $\Longrightarrow$ usually written as aba
- $(ab)^*$ is a regular expression.
- $((ab)^*)^*$ is a regular expression.
- ab^* is a regular expression.

Some examples of regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $\emptyset$ is a regular expression.
- $\emptyset \cdot \emptyset$ is a regular expression. $\Longrightarrow$ usually written as $\emptyset\emptyset$
- $\emptyset a$ is a regular expression.
- $(ab)a$ is a regular expression. $\Longrightarrow$ usually written as aba
- $(ab)^*$ is a regular expression.
- $((ab)^*)^*$ is a regular expression.
- ab^* is a regular expression.
- $(ab)^* \cup ab^*$ is a regular expression.

Examples that are not regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

Examples that are not regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

Examples that are not regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $a \cap b$ is not a regular expression, because $\cap$ is not allowed.

Examples that are not regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $a \cap b$ is not a regular expression, because $\cap$ is not allowed.
- c is not a regular expression over Σ, because $c \notin \Sigma$.

Examples that are not regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $a \cap b$ is not a regular expression, because $\cap$ is not allowed.
- c is not a regular expression over Σ, because $c \notin \Sigma$.
- $*$ is not a regular expression.

Examples that are not regular expressions

- $\emptyset$ is a regular expression.
- a is a regular expression, for every symbol $a \in \Sigma$.
- If e_1, e_2 are regular expressions, so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If e is a regular expression, so is $(e)^*$.

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

- $a \cap b$ is not a regular expression, because $\cap$ is not allowed.
- c is not a regular expression over Σ, because $c \notin \Sigma$.
- $*$ is not a regular expression.
- (NOT a) is not a regular expression.

The meaning of a regular expression

Each regular expression e represents/defines a language $L(e)$.

The meaning of a regular expression

Each regular expression e represents/defines a language $L(e)$.

As analogy, a C++ program:

Boolean `myprog`(String w)

The meaning of a regular expression

Each regular expression e represents/defines a language $L(e)$.

As analogy, a C++ program:

$$\text{Boolean}\;\; \texttt{myprog}(\text{String } w)$$

We can view `myprog` as defining the language:

$$L(\texttt{myprog}) \quad := \quad \{w \mid \texttt{myprog} \text{ outputs true on } w\}$$

The meaning of a regular expression

Each regular expression e represents/defines a language $L(e)$.

As analogy, a C++ program:

$$\text{Boolean} \;\; \texttt{myprog}(\text{String } w)$$

We can view `myprog` as defining the language:

$$L(\texttt{myprog}) \quad := \quad \{w \mid \texttt{myprog} \text{ outputs true on } w\}$$

So we can say that `myprog` is a finite representation of a (possibly infinite) language $L(\texttt{myprog})$.

The meaning of a regular expression

Each regular expression e represents/defines a language $L(e)$.

As analogy, a C++ program:

$$\text{Boolean } \ \texttt{myprog}(\text{String } w)$$

We can view `myprog` as defining the language:

$$L(\texttt{myprog}) \quad := \quad \{w \mid \texttt{myprog} \text{ outputs true on } w\}$$

So we can say that `myprog` is a finite representation of a (possibly infinite) language $L(\texttt{myprog})$.

Similarly, we can say that a regular expression e is a finite representation of the language $L(e)$.

The formal definition of the language $L(e)$

(Def.) A regular expression e over Σ defines the language $L(e)$ over the same alphabet Σ as follows.

The formal definition of the language $L(e)$

(Def.) A regular expression e over Σ defines the language $L(e)$ over the same alphabet Σ as follows.

- If e is $\emptyset$, then $L(e) = \emptyset$.

The formal definition of the language $L(e)$

(Def.) A regular expression e over Σ defines the language $L(e)$ over the same alphabet Σ as follows.

- If e is $\emptyset$, then $L(e) = \emptyset$.
- If e is a, where a is a symbol in Σ, then $L(e) = \{a\}$.

The formal definition of the language $L(e)$

(Def.) A regular expression e over Σ defines the language $L(e)$ over the same alphabet Σ as follows.

- If e is $\emptyset$, then $L(e) = \emptyset$.
- If e is a, where a is a symbol in Σ, then $L(e) = \{a\}$.
- If e is of the form $e_1 e_2$, then $L(e) = L(e_1) \cdot L(e_2)$.

The formal definition of the language $L(e)$

(Def.) A regular expression e over Σ defines the language $L(e)$ over the same alphabet Σ as follows.

- If e is $\emptyset$, then $L(e) = \emptyset$.
- If e is a, where a is a symbol in Σ, then $L(e) = \{a\}$.
- If e is of the form $e_1 e_2$, then $L(e) = L(e_1) \cdot L(e_2)$.
- If e is of the form $e_1 \cup e_2$, then $L(e) = L(e_1) \cup L(e_2)$.

The formal definition of the language $L(e)$

(Def.) A regular expression e over Σ defines the language $L(e)$ over the same alphabet Σ as follows.

- If e is $\emptyset$, then $L(e) = \emptyset$.
- If e is a, where a is a symbol in Σ, then $L(e) = \{a\}$.
- If e is of the form $e_1 e_2$, then $L(e) = L(e_1) \cdot L(e_2)$.
- If e is of the form $e_1 \cup e_2$, then $L(e) = L(e_1) \cup L(e_2)$.
- If e is of the form $(e_1)^*$, then $L(e) = L(e_1)^*$.

Some examples

Some examples

- $L(\emptyset) = \emptyset$.

Some examples

- $L(\emptyset) = \emptyset$.
- $L(a) = \{a\}$.

Some examples

- $L(\emptyset) = \emptyset$.
- $L(a) = \{a\}$.
- $L(ab) = L(a)L(b) = \{a\} \cdot \{b\} = \{ab\}$.

Some examples

- $L(\emptyset) = \emptyset$.
- $L(a) = \{a\}$.
- $L(ab) = L(a)L(b) = \{a\} \cdot \{b\} = \{ab\}$.
- $L((a \cup b)^*) = (L(a \cup b))^* = (L(a) \cup L(b))^* = (\{a\} \cup \{b\})^* = (\{a, b\})^* = \{a, b\}^*$.

Some examples

- $L(\emptyset) = \emptyset$.
- $L(a) = \{a\}$.
- $L(ab) = L(a)L(b) = \{a\} \cdot \{b\} = \{ab\}$.
- $L((a \cup b)^*) = (L(a \cup b))^* = (L(a) \cup L(b))^* = (\{a\} \cup \{b\})^* = (\{a, b\})^* = \{a, b\}^*$.
- $L(a\emptyset) = L(a)L(\emptyset) = \{a\} \cdot \emptyset = \emptyset$.

Some examples

- $L(\emptyset) = \emptyset$.
- $L(a) = \{a\}$.
- $L(ab) = L(a)L(b) = \{a\} \cdot \{b\} = \{ab\}$.
- $L((a \cup b)^*) = (L(a \cup b))^* = (L(a) \cup L(b))^* = (\{a\} \cup \{b\})^* = (\{a, b\})^* = \{a, b\}^*$.
- $L(a\emptyset) = L(a)L(\emptyset) = \{a\} \cdot \emptyset = \emptyset$.
- $L(\emptyset^*) = L(\emptyset)^* = \emptyset^* = \{\varepsilon\}$.

Some examples

- $L(\emptyset) = \emptyset$.
- $L(a) = \{a\}$.
- $L(ab) = L(a)L(b) = \{a\} \cdot \{b\} = \{ab\}$.
- $L((a \cup b)^*) = (L(a \cup b))^* = (L(a) \cup L(b))^* = (\{a\} \cup \{b\})^* = (\{a, b\})^* = \{a, b\}^*$.
- $L(a\emptyset) = L(a)L(\emptyset) = \{a\} \cdot \emptyset = \emptyset$.
- $L(\emptyset^*) = L(\emptyset)^* = \emptyset^* = \{\varepsilon\}$.
- $L((a \cup b)^* a) = (L(a \cup b)^* \cdot L(a) = \{a, b\}^* \cdot \{a\}$.
  That is, the language $\{w \mid w \text{ is a word that ends with } a\}$.

The main theorem in this lesson

Theorem 2.1

Regular expressions define precisely the class of regular languages.
More formally:

- *For every regular expression e over Σ, $L(e)$ is a regular language, i.e., there is an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(e)$.*
- *For every NFA $\mathcal{A}$, there is a regular expression e such that $L(e) = L(\mathcal{A})$.*

Table of contents

The main theorem in this lesson

Theorem 2.1

Regular expressions define precisely the class of regular languages.

More formally:

- *For every regular expression e over Σ, $L(e)$ is a regular language, i.e., there is an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(e)$.*
- *For every NFA $\mathcal{A}$, there is a regular expression e such that $L(e) = L(\mathcal{A})$.*

We will first prove the first item:

Theorem (The first item of Theorem 2.1)

- *For every regular expression e over Σ, $L(e)$ is a regular language, i.e., there is an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(e)$.*

(Proof) By induction on the regex e. The base case is when e is either $\emptyset$ or a symbol $a \in \Sigma$.

We will first prove the first item:

Theorem (The first item of Theorem 2.1)

- *For every regular expression e over Σ, $L(e)$ is a regular language, i.e., there is an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(e)$.*

(Proof) By induction on the regex e. The base case is when e is either $\emptyset$ or a symbol $a \in \Sigma$.

- When e is $\emptyset$, then $L(e) = \emptyset$.

We will first prove the first item:

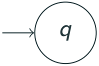
Theorem (The first item of Theorem 2.1)

- *For every regular expression e over Σ, $L(e)$ is a regular language, i.e., there is an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(e)$.*

(Proof) By induction on the regex e. The base case is when e is either $\emptyset$ or a symbol $a \in \Sigma$.

- When e is $\emptyset$, then $L(e) = \emptyset$.

  The NFA that accepts $\emptyset$ is:



We will first prove the first item:

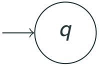
Theorem (The first item of Theorem 2.1)

- *For every regular expression e over Σ, $L(e)$ is a regular language, i.e., there is an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(e)$.*

(Proof) By induction on the regex e. The base case is when e is either $\emptyset$ or a symbol $a \in \Sigma$.

- When e is $\emptyset$, then $L(e) = \emptyset$.

  The NFA that accepts $\emptyset$ is:

- When e is a, for some symbol $a \in \Sigma$, then $L(e) = \{a\}$.

We will first prove the first item:

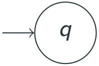
Theorem (The first item of Theorem 2.1)

- *For every regular expression e over Σ, $L(e)$ is a regular language, i.e., there is an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(e)$.*

(Proof) By induction on the regex e. The base case is when e is either $\emptyset$ or a symbol $a \in \Sigma$.

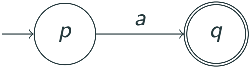
- When e is $\emptyset$, then $L(e) = \emptyset$.

  The NFA that accepts $\emptyset$ is:



- When e is a, for some symbol $a \in \Sigma$, then $L(e) = \{a\}$.

  The NFA that accepts $\{a\}$ is:



(Proof continued)

For the induction step, suppose *e* is either of the form $\alpha \cdot \beta$, $\alpha \cup \beta$ or α^*.

(Proof continued)

For the induction step, suppose e is either of the form $\alpha \cdot \beta$, $\alpha \cup \beta$ or α^*.

By the induction hypothesis, there are NFA $\mathcal{A}_1$ and $\mathcal{A}_2$ that accept the languages $L(\alpha)$ and $L(\beta)$, respectively.

(Proof continued)

For the induction step, suppose e is either of the form $\alpha \cdot \beta$, $\alpha \cup \beta$ or α^*.

By the induction hypothesis, there are NFA $\mathcal{A}_1$ and $\mathcal{A}_2$ that accept the languages $L(\alpha)$ and $L(\beta)$, respectively.

Since regular languages are closed under concatenation, union and Kleene star, (See Remark 1.4 and Theorem 1.8 in Lesson 1), there are NFAs for all the languages $L(\alpha \cdot \beta)$, $L(\alpha \cup \beta)$ and $L(\alpha^*)$.

(Proof continued)

For the induction step, suppose e is either of the form $\alpha \cdot \beta$, $\alpha \cup \beta$ or α^*.

By the induction hypothesis, there are NFA $\mathcal{A}_1$ and $\mathcal{A}_2$ that accept the languages $L(\alpha)$ and $L(\beta)$, respectively.

Since regular languages are closed under concatenation, union and Kleene star, (See Remark 1.4 and Theorem 1.8 in Lesson 1), there are NFAs for all the languages $L(\alpha \cdot \beta)$, $L(\alpha \cup \beta)$ and $L(\alpha^*)$.

Recall that:

- $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$.
- $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$.
- $L(\alpha^*) = L(\alpha)^*$.

We now prove the second item:

Theorem (The second item of Theorem 2.1)

- *For every NFA $\mathcal{A}$, there is a regular expression e such that $L(e) = L(\mathcal{A})$.*

(Proof) Let $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ be an NFA, where $Q = \{1, \ldots, n\}$.

We now prove the second item:

Theorem (The second item of Theorem 2.1)

- *For every NFA $\mathcal{A}$, there is a regular expression e such that $L(e) = L(\mathcal{A})$.*

(Proof) Let $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ be an NFA, where $Q = \{1, \ldots, n\}$.

For every $1 \leqslant i, j \leqslant n$ and $0 \leqslant k \leqslant n$, define the language $L(i, j, k)$:

$$L(i,j,k) \quad := \quad \left\{ \; w \in \Sigma^* \; \middle| \; \begin{array}{l} \text{there is a run of } \mathcal{A} \text{ on } w \text{ from state } i \text{ to state } j \\ \textit{without} \text{ passing any states } \; \geqslant k+1 \end{array} \right\}$$

We now prove the second item:

Theorem (The second item of Theorem 2.1)

- *For every NFA $\mathcal{A}$, there is a regular expression e such that $L(e) = L(\mathcal{A})$.*

(Proof) Let $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ be an NFA, where $Q = \{1, \ldots, n\}$.

For every $1 \leqslant i, j \leqslant n$ and $0 \leqslant k \leqslant n$, define the language $L(i, j, k)$:

$$L(i,j,k) \quad := \quad \left\{ \; w \in \Sigma^* \; \middle| \; \begin{array}{l} \text{there is a run of } \mathcal{A} \text{ on } w \text{ from state } i \text{ to state } j \\ \textit{without} \text{ passing any states } \; \geqslant k+1 \end{array} \right\}$$

That is, if $w \in L(i, j, k)$, there is a run of $\mathcal{A}$ on w from state i to j *without* passing through the states $k+1, \ldots, n$.

We now prove the second item:

Theorem (The second item of Theorem 2.1)

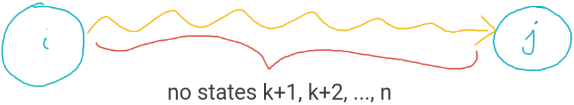
- *For every NFA $\mathcal{A}$, there is a regular expression e such that $L(e) = L(\mathcal{A})$.*

(Proof) Let $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ be an NFA, where $Q = \{1, \ldots, n\}$.

For every $1 \leqslant i, j \leqslant n$ and $0 \leqslant k \leqslant n$, define the language $L(i, j, k)$:

$$L(i,j,k) \quad := \quad \left\{ \; w \in \Sigma^* \;\middle|\; \begin{array}{l} \text{there is a run of } \mathcal{A} \text{ on } w \text{ from state } i \text{ to state } j \\ \textit{without} \text{ passing any states } \geqslant k+1 \end{array} \right\}$$

That is, if $w \in L(i, j, k)$, there is a run of $\mathcal{A}$ on w from state i to j *without* passing through the states $k + 1, \ldots, n$.



Claim 1

For every $1 \leqslant i, j \leqslant n$ *and* $0 \leqslant k \leqslant n$*, there is a regex* e *such that* $L(e) = L(i, j, k)$.

(Proof of Claim 1: By induction on k)

Claim 1

For every $1 \leqslant i, j \leqslant n$ and $0 \leqslant k \leqslant n$, there is a regex e such that $L(e) = L(i, j, k)$.

(Proof of Claim 1: By induction on k)

(Base case $k = 0$) For every $1 \leqslant i, j \leqslant n$, we consider $L(i, j, 0)$:

Claim 1

For every $1 \leqslant i, j \leqslant n$ and $0 \leqslant k \leqslant n$, there is a regex e such that $L(e) = L(i, j, k)$.

(Proof of Claim 1: By induction on k)

(Base case $k = 0$) For every $1 \leqslant i, j \leqslant n$, we consider $L(i, j, 0)$:

- If $i \neq j$ and there is no transition from i to j:

The language $L(i, j, 0) = \emptyset$, so the regex e is $\emptyset$.

Claim 1

For every $1 \leqslant i, j \leqslant n$ and $0 \leqslant k \leqslant n$, there is a regex e such that $L(e) = L(i, j, k)$.

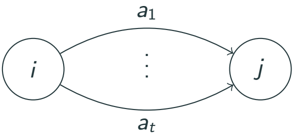
(Proof of Claim 1: By induction on k)

(Base case $k = 0$) For every $1 \leqslant i, j \leqslant n$, we consider $L(i, j, 0)$:

- If $i \neq j$ and there is no transition from i to j:

  The language $L(i, j, 0) = \emptyset$, so the regex e is $\emptyset$.

- If $i \neq j$ and there are some transitions from i to j:

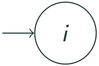


  The language $L(i, j, 0) = \{a_1, \ldots, a_t\}$, so the regex e is $a_1 \cup \cdots \cup a_t$.

(Base case $k = 0$ – continued)

(Base case $k = 0$ – continued)

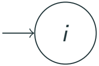
- If $i = j$ and there is no transition from i to i:



The language $L(i, j, 0) = \{\varepsilon\}$, so the regex e is $\emptyset^*$.

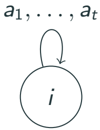
(Base case $k = 0$ – continued)

- If $i = j$ and there is no transition from i to i:



The language $L(i, j, 0) = \{\varepsilon\}$, so the regex e is $\emptyset^*$.

- If $i = j$ and there are some transitions from i to j:



The language $L(i, j, 0) = \{a_1, \ldots, a_t, \varepsilon\}$, so the regex e is $a_1 \cup \cdots \cup a_t \cup \emptyset^*$.

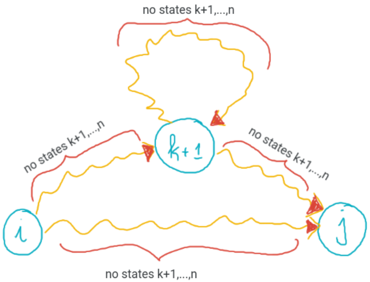
(Induction step – proof of Claim 1)

(Induction step – proof of Claim 1) We have the identity:

$$L(i,j,k+1) \quad = \quad L(i,j,k) \;\cup\; \Big(L(i,k+1,k)\cdot L(k+1,k+1,k)^{*}\cdot L(k+1,j,k)\Big)$$

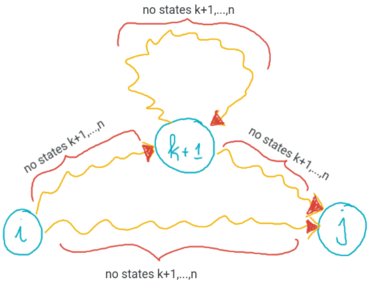
(Induction step – proof of Claim 1) We have the identity:

$$L(i,j,k+1) \quad = \quad L(i,j,k) \cup \Big(L(i,k+1,k) \cdot L(k+1,k+1,k)^* \cdot L(k+1,j,k)\Big)$$



(Induction step – proof of Claim 1) We have the identity:

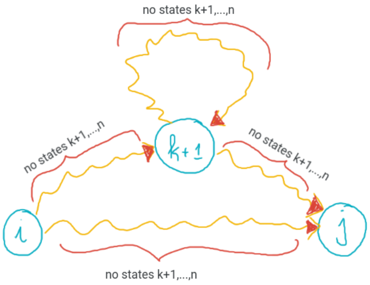
$$L(i,j,k+1) \quad = \quad L(i,j,k) \,\cup\, \Big(L(i,k+1,k)\cdot L(k+1,k+1,k)^{*}\cdot L(k+1,j,k)\Big)$$



By induction hypothesis, there is regex for each of $L(i,j,k)$, $L(i,k+1,k)$, $L(k+1,k+1,k)$, and $L(k+1,j,k)$.

(Induction step – proof of Claim 1) We have the identity:

$$L(i,j,k+1) \quad = \quad L(i,j,k) \;\cup\; \Big(L(i,k+1,k)\cdot L(k+1,k+1,k)^{*}\cdot L(k+1,j,k)\Big)$$



By induction hypothesis, there is regex for each of $L(i,j,k)$, $L(i,k+1,k)$, $L(k+1,k+1,k)$, and $L(k+1,j,k)$.

Thus, there is regex for $L(i,j,k+1)$.

(Finishing the proof of Claim 1)

The language $L(\mathcal{A})$ can be defined as:

$$L(\mathcal{A}) \quad = \quad \bigcup_{q_f \in F} L(q_0, q_f, n)$$

(Finishing the proof of Claim 1)

The language $L(\mathcal{A})$ can be defined as:

$$L(\mathcal{A}) \quad = \quad \bigcup_{q_f \in F} L(q_0, q_f, n)$$

By Claim 1, there is a regex that defines each $L(q_0, q_f, n)$.

(Finishing the proof of Claim 1)

The language $L(\mathcal{A})$ can be defined as:

$$L(\mathcal{A}) \quad = \quad \bigcup_{q_f \in F} L(q_0, q_f, n)$$

By Claim 1, there is a regex that defines each $L(q_0, q_f, n)$.

Taking the union over all $q_f \in F$, we have a regex for $L(\mathcal{A})$.

To conclude:

Corollary 2.2

Let L be a language. The following are equivalent.

- *L is accepted by a DFA.*
- *L is accepted by an NFA.*
- *L is defined by a regular expression.*

To conclude:

Corollary 2.2

Let L be a language. The following are equivalent.

- *L is accepted by a DFA.*
- *L is accepted by an NFA.*
- *L is defined by a regular expression.*

One nice implication of this corollary is that languages defined by regular expression are also closed under intersection and complement.

To conclude:

Corollary 2.2

Let L be a language. The following are equivalent.

- *L is accepted by a DFA.*
- *L is accepted by an NFA.*
- *L is defined by a regular expression.*

One nice implication of this corollary is that languages defined by regular expression are also closed under intersection and complement.

This is despite the fact that we are not allowed to use intersection or negation in regular expressions.

End of Lesson 2