Regular Languages

(Based on [Sipser 2006, 2013])

Yu-Fang Chen

Department of Information Management
National Taiwan University

Schematic of Finite Automata

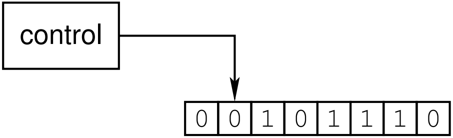


Figure: Schematic of Finite Automata

- A finite automaton has a finite set of control states.
- A finite automaton reads input symbols from left to right.
- A finite automaton accepts or rejects an input after reading the input.

Finite Automaton M_1

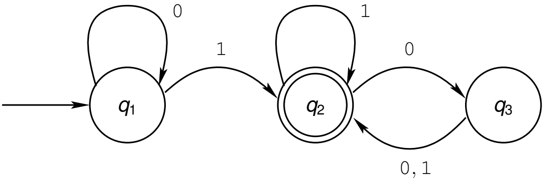
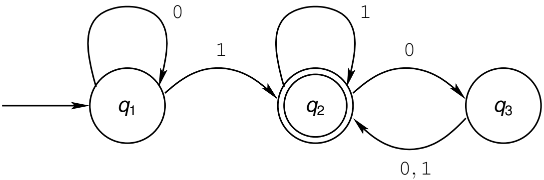


Figure: A Finite Automaton M_1

The *state diagram* of a finite automaton M_1. M_1 has

- 3 *states*: q_1, q_2, q_3;
- a *start state*: q_1;
- a *accept state*: q_2;
- 6 *transitions*: $q_1 \xrightarrow{0} q_1$, $q_1 \xrightarrow{1} q_2$, $q_2 \xrightarrow{1} q_2$, $q_2 \xrightarrow{0} q_3$, $q_3 \xrightarrow{0} q_2$, and $q_3 \xrightarrow{1} q_2$.

Accepted and Rejected String



- Consider an input string `1100`.
- M_1 processes the string from the start state q_1.
- It takes the transition labeled by the current symbol and moves to the next state.
- At the end of the string, there are two cases:
  - If M_1 is at an accept state, M_1 outputs *accept*;
  - Otherwise, M_1 outputs *reject*.
- Strings accepted by M_1: `1, 01, 11, 1100, 1101, ...`.
- Strings rejected by M_1: `0, 00, 10, 010, 1010, ...`.

Finite Automaton – Formal Definition

- A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
  - Q is a finite set of *states*;
  - Σ is a finite set called *alphabet*;
  - $\delta : Q \times \Sigma \to Q$ is the *transition function*;
  - $q_0 \in Q$ is the *start state*; and
  - $F \subseteq Q$ is the set of *accept states*.
- Accept states are also called *final states*.
- The set of all strings that M accepts is called the *language of machine M* (written $L(M)$).
  - Recall a *language* is a set of strings.
- We also say M *recognizes* (or *accepts*) $L(M)$.

M_1 – Formal Definition

- A finite automaton $M_1 = (Q, \Sigma, \delta, q_1, F)$ consists of
  - $Q = \{q_1, q_2, q_3\}$;
  - $\Sigma = \{\mathtt{0}, \mathtt{1}\}$;
  - $\delta : Q \times \Sigma \to Q$ is

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

  - q_1 is the start state; and
  - $F = \{q_2\}$.
- Moreover, we have

$$L(M_1) = \{w : \begin{array}{l} w \text{ contains at least one } \mathtt{1} \text{ and} \\ \text{an even number of } \mathtt{0}\text{'s follow the last } \mathtt{1}\} \end{array}$$

Finite Automaton M_2

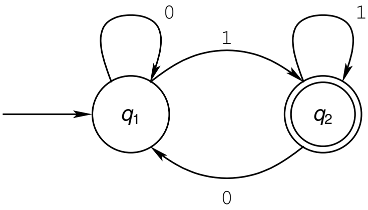


Figure: Finite Automaton M_2

- $M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$ where δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

- What is $L(M_2)$?

Finite Automaton M_2

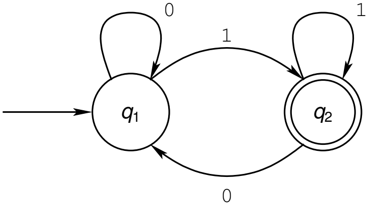


Figure: Finite Automaton M_2

- $M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$ where δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

- What is $L(M_2)$?
  - $L(M_2) = \{w : w \text{ ends in a } 1\}$.

Finite Automaton M_3

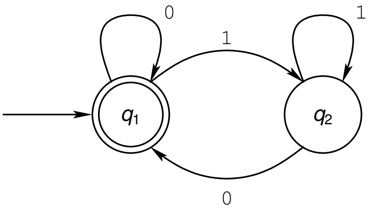


Figure: Finite Automaton M_3

- $M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$ where δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

- What is $L(M_3)$?

Finite Automaton M_3

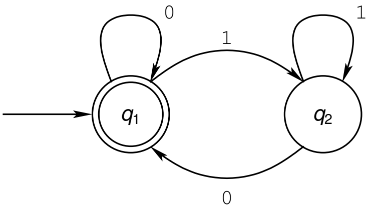


Figure: Finite Automaton M_3

- $M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$ where δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

- What is $L(M_3)$?
  - $L(M_3) = \{w : w$ is the empty string ϵ or ends in a $0\}$.

Finite Automaton M_5

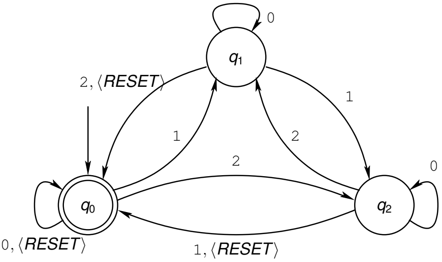


Figure: Finite Automaton M_5

- $M_5 = (\{q_0, q_1, q_2\}, \{0, 1, 2, \langle RESET \rangle\}, \delta, q_0, \{q_0\})$.
- Strings accepted by M_5: $0, 00, 12, 21, 012, 102, 120, 021, 201, 210, 111, 222, \ldots$.
- M_5 computes the sum of input symbols modulo 3. It resets upon the input symbol $\langle RESET \rangle$. M_5 accepts strings who sum is a multiple of 3.

Computation – Formal Definition

IM NTU

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and $w = w_1 w_2 \cdots w_n$ a string where $w_i \in \Sigma$ for every $i = 1, \ldots, n$.
- We say M *accepts* w if there is a sequence of states $r_0, r_1, \ldots, r_n$ such that
  - $r_0 = q_0$;
  - $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$; and
  - $r_n \in F$.
- *M recognizes language A* if $A = \{w : M \text{ accepts } w\}$.

Definition

A language is called a *regular language* if some finite automaton recognizes it.

Regular Operations

Definition

Let A and B be languages. We define the following operations:

- *Union*: $A \cup B = \{x : x \in A \text{ or } x \in B\}$.
- *Concatenation*: $A \circ B = \{xy : x \in A \text{ and } y \in B\}$.
- *Star*: $A^* = \{x_1 x_2 \cdots x_k : k \geq 0 \text{ and every } x_i \in A\}$.

- Note that $\epsilon \in A^*$ for every language A.

Closure Property – Union

IM NTU

Theorem

The class of regular languages is closed under the union operation. That is, $A_1 \cup A_2$ is regular if A_1 and A_2 are.

Proof.

Let $M_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ recognize A_i for $i = 1, 2$. Construct $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\}$;
- $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$;
- $q_0 = (q_1, q_2)$;
- $F = (F_1 \times Q_2) \cup (Q_1 \times F_2) = \{(r_1, r_2) : r_1 \in F_1 \text{ or } r_2 \in F_2\}$. $\square$

- Why is $L(M) = A_1 \cup A_2$?

Nondeterminism

- When a machine is at a given state and reads an input symbol, there is precisely one choice of its next state.
- This is call *deterministic* computation.
- In *nondeterministic* machines, multiple choices may exist for the next state.
- A deterministic finite automaton is abbreviated as *DFA*; a nondeterministic finite automaton is abbreviated as *NFA*.
- A DFA is also an NFA.
- Since NFA allow more general computation, they can be much smaller than DFA.

NFA N_4

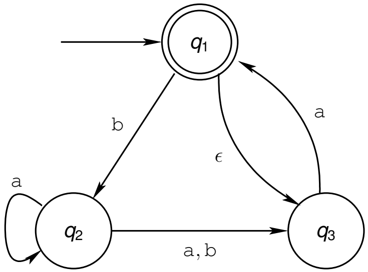


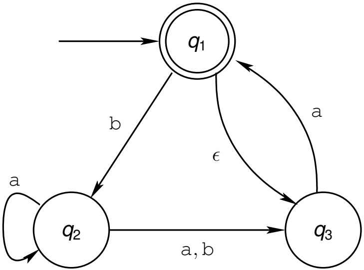
Figure: NFA N_4

- On input string baa, N_4 has several possible computation:
  - $q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_2 \xrightarrow{a} q_2$;
  - $q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_2 \xrightarrow{a} q_3$; or
  - $q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{a} q_1$.

Nondeterministic Finite Automaton – Formal Definition

- For any set Q, $\mathcal{P}(Q) = \{R : R \subseteq Q\}$ denotes the *power set* of Q.
- For any alphabet Σ, define Σ_ϵ to be $\Sigma \cup \{\epsilon\}$.
- A *nondeterministic finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
  - Q is a finite set of states;
  - Σ is a finite alphabet;
  - $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition function;
  - $q_0 \in Q$ is the start state; and
  - $F \subseteq Q$ is the accept states.
- Note that the transition function accepts the empty string as an input symbol.

NFA N_4 – Formal Definition



- $N_4 = (Q, \Sigma, \delta, q_1, \{q_1\})$ is a nondeterministic finite automaton where
  - $Q = \{q_1, q_2, q_3\}$;
  - Its transition function δ is

	ϵ	a	b
q_1	$\{q_3\}$	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\{q_2, q_3\}$	$\{q_3\}$
q_3	$\emptyset$	$\{q_1\}$	$\emptyset$

Nondeterministic Computation – Formal Definition

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over Σ. We say N *accepts* w if w can be rewritten as $w = y_1 y_2 \cdots y_m$ with $y_i \in \Sigma_\epsilon$ and there is a sequence of states $r_0, r_1, \ldots, r_m$ such that
  - $r_0 = q_0$;
  - $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, \ldots, m-1$; and
  - $r_m \in F$.
- Note that finitely many empty strings can be inserted in w.
- Also note that one sequence satisfying the conditions suffices to show the acceptance of an input string.
- Strings accepted by N_4: $\mathtt{a}, \mathtt{baa}, \ldots$.

Equivalence of NFA's and DFA's

Theorem

Every nondeterministic finite automaton has an equivalent deterministic finite automaton. That is, for every NFA N, there is a DFA M such that $L(M) = L(N)$.

Proof.

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA. For $R \subseteq Q$, define $E(R) = \{q : q$ can be reached from R along 0 or more ϵ transitions $\}$. Construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ where

- $Q' = \mathcal{P}(Q)$;
- $\delta'(R, a) = \{q \in Q : q \in E(\delta(r, a))$ for some $r \in R\}$;
- $q'_0 = E(\{q_0\})$;
- $F' = \{R \in Q' : R \cap F \neq \emptyset\}$. □

- Why is $L(M) = L(N)$?

A DFA Equivalent to N_4

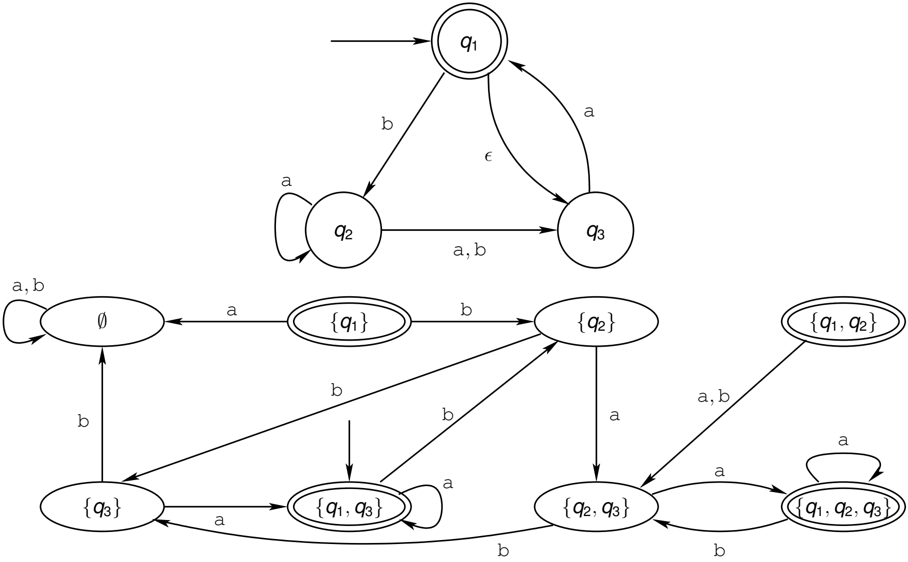


Figure: A DFA Equivalent to N_4

Closure Properties – Revisited

Theorem

The class of regular languages is closed under the union operation.

Proof.

Let $N_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ recognize A_i for $i = 1, 2$. Construct $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{q_0\} \cup Q_1 \cup Q_2$; $F = F_1 \cup F_2$; and
- $$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

$\square$

- Why is $L(N) = L(N_1) \cup L(N_2)$?

Closure Properties – Revisited

IM NTU

Theorem

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ recognize A_i for $i = 1, 2$. Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ where

- $Q = Q_1 \cup Q_2$; and
- $\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a) & q \in Q_2 \end{cases}$

$\square$

- Why is $L(N) = L(N_1) \circ L(N_2)$?

Closure Properties – Revisited

Theorem

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1. Construct $N = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{q_0\} \cup Q_1$; $F = \{q_0\} \cup F_1$; and
- $$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

$\square$

- Why is $L(N) = [L(N_1)]^*$?

Closure Properties – Revisited

IM NTU

Theorem

The class of regular languages is closed under complementation.

Proof.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing A. Consider $\overline{M} = (Q, \Sigma, \delta, q_0, Q \setminus F)$. We have $w \in L(M)$ if and only if $w \notin L(\overline{M})$. That is, $L(\overline{M}) = \overline{A}$ as required. $\square$

Regular Expressions

Definition

R is a *regular expression* if R is

- a for some $a \in \Sigma$;
- ϵ;
- $\emptyset$;
- $(R_1 \cup R_2)$ where R_i's are regular expressions;
- $(R_1 \circ R_2)$ where R_i's are regular expressions; or
- (R_1^*) where R_1 is a regular expression.

- We write R^+ for $R \circ R^*$. Hence $R^* = R^+ \cup \epsilon$.
- Moreover, write R^k for $\overbrace{R \circ R \circ \cdots \circ R}^{k}$.
  - Define $R^0 = \epsilon$. We have $R^* = R^0 \cup R^1 \cup \cdots \cup R^n \cup \cdots$.
- $L(R)$ denotes the language described by the regular expression R.

Examples of Regular Expressions

- For convenience, we write RS for $R \circ S$.
- We may also write the regular expression R to denote its language $L(R)$.
- $L(0^*10^*) =$
- $L(\Sigma^*1\Sigma^*) =$
- $L((\Sigma\Sigma)^*) =$
- $(0 \cup \epsilon)(1 \cup \epsilon) =$
- $1^*\emptyset =$
- $\emptyset^* =$
- For any regular expression R, we have $R \cup \emptyset = R$ and $R \circ \epsilon = R$.

Examples of Regular Expressions

- For convenience, we write RS for $R \circ S$.
- We may also write the regular expression R to denote its language $L(R)$.
- $L(\mathtt{0}^*\mathtt{1}\mathtt{0}^*) = \{w : w \text{ contains a single } \mathtt{1}\}$.
- $L(\Sigma^*\mathtt{1}\Sigma^*) =$
- $L((\Sigma\Sigma)^*) =$
- $(\mathtt{0} \cup \epsilon)(1 \cup \epsilon) =$
- $\mathtt{1}^*\emptyset =$
- $\emptyset^* =$
- For any regular expression R, we have $R \cup \emptyset = R$ and $R \circ \epsilon = R$.

Examples of Regular Expressions

- For convenience, we write RS for $R \circ S$.
- We may also write the regular expression R to denote its language $L(R)$.
- $L(0^*10^*) = \{w : w \text{ contains a single } 1\}$.
- $L(\Sigma^*1\Sigma^*) = \{w : w \text{ has at least one } 1\}$.
- $L((\Sigma\Sigma)^*) =$
- $(0 \cup \epsilon)(1 \cup \epsilon) =$
- $1^*\emptyset =$
- $\emptyset^* =$
- For any regular expression R, we have $R \cup \emptyset = R$ and $R \circ \epsilon = R$.

Examples of Regular Expressions

- For convenience, we write RS for $R \circ S$.
- We may also write the regular expression R to denote its language $L(R)$.
- $L(\mathtt{0}^*\mathtt{1}\mathtt{0}^*) = \{w : w \text{ contains a single } \mathtt{1}\}$.
- $L(\Sigma^*\mathtt{1}\Sigma^*) = \{w : w \text{ has at least one } \mathtt{1}\}$.
- $L((\Sigma\Sigma)^*) = \{w : w \text{ is a string of even length }\}$.
- $(\mathtt{0} \cup \epsilon)(1 \cup \epsilon) =$
- $\mathtt{1}^*\emptyset =$
- $\emptyset^* =$
- For any regular expression R, we have $R \cup \emptyset = R$ and $R \circ \epsilon = R$.

Examples of Regular Expressions

- For convenience, we write RS for $R \circ S$.
- We may also write the regular expression R to denote its language $L(R)$.
- $L(0^*10^*) = \{w : w \text{ contains a single } 1\}$.
- $L(\Sigma^*1\Sigma^*) = \{w : w \text{ has at least one } 1\}$.
- $L((\Sigma\Sigma)^*) = \{w : w \text{ is a string of even length }\}$.
- $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$.
- $1^*\emptyset =$
- $\emptyset^* =$
- For any regular expression R, we have $R \cup \emptyset = R$ and $R \circ \epsilon = R$.

Examples of Regular Expressions

- For convenience, we write RS for $R \circ S$.
- We may also write the regular expression R to denote its language $L(R)$.
- $L(\mathtt{0}^*\mathtt{1}\mathtt{0}^*) = \{w : w \text{ contains a single } \mathtt{1}\}$.
- $L(\Sigma^*\mathtt{1}\Sigma^*) = \{w : w \text{ has at least one } \mathtt{1}\}$.
- $L((\Sigma\Sigma)^*) = \{w : w \text{ is a string of even length }\}$.
- $(\mathtt{0} \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, \mathtt{0}, \mathtt{1}, \mathtt{01}\}$.
- $\mathtt{1}^*\emptyset = \emptyset$.
- $\emptyset^* =$
- For any regular expression R, we have $R \cup \emptyset = R$ and $R \circ \epsilon = R$.

Examples of Regular Expressions

- For convenience, we write RS for $R \circ S$.
- We may also write the regular expression R to denote its language $L(R)$.
- $L(\mathtt{0}^*\mathtt{1}\mathtt{0}^*) = \{w : w \text{ contains a single } \mathtt{1}\}$.
- $L(\Sigma^*\mathtt{1}\Sigma^*) = \{w : w \text{ has at least one } \mathtt{1}\}$.
- $L((\Sigma\Sigma)^*) = \{w : w \text{ is a string of even length }\}$.
- $(\mathtt{0} \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, \mathtt{0}, \mathtt{1}, \mathtt{01}\}$.
- $\mathtt{1}^*\emptyset = \emptyset$.
- $\emptyset^* = \{\epsilon\}$.
- For any regular expression R, we have $R \cup \emptyset = R$ and $R \circ \epsilon = R$.

Regular Expressions and Finite Automata

IM NTU

Lemma

If a language is described by a regular expression, it is regular.

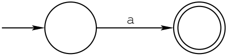
Proof.

We prove by induction on the regular expression R.

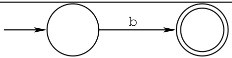
- $R = a$ for some $a \in \Sigma$. Consider the NFA $N_a = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$ where $\delta(r, y) = \begin{cases} \{q_2\} & r = q_1 \text{ and } y = a \\ \emptyset & \text{otherwise} \end{cases}$
- $R = \epsilon$. Consider the NFA $N_\epsilon = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$ where $\delta(r, y) = \emptyset$ for any r and y.
- $R = \emptyset$. Consider the NFA $N_\emptyset = (\{q_1\}, \Sigma, \delta, q_1, \emptyset)$ where $\delta(r, y) = \emptyset$ for any r and y.
- $R = R_1 \cup R_2$, $R = R_1 \circ R_2$, or $R = R_1^*$. By inductive hypothesis and the closure properties of finite automata. □

Regular Expressions and Finite Automata

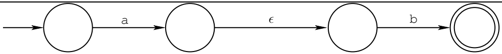
a



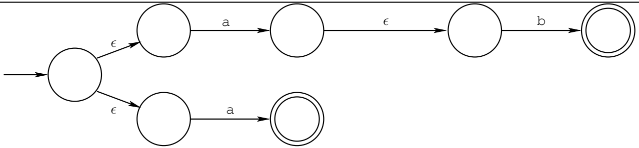
b



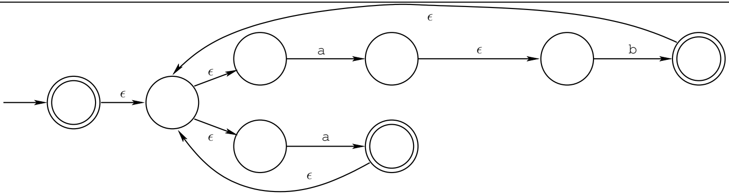
ab



ab ∪ a



$(ab \cup a)^*$



Regular Expressions and Finite Automata

Lemma

If a language is regular, it is described by a regular expression.

For the proof, we introduce a generalization of finite automata.

Generalized Nondeterministic Finite Automata

Definition

A *generalized nondeterministic finite automaton* is a 5-tuple $(Q, \Sigma, q_{\text{start}}, q_{\text{accept}})$ where

- Q is the finite set of states;
- Σ is the input alphabet;
- $\delta : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \to \mathcal{R}$ is the transition function, where $\mathcal{R}$ denotes the set of regular expressions;
- q_{start} is the start state; and
- q_{accept} is the accept state.

A GNFA *accepts* a string $w \in \Sigma^*$ if $w = w_1 w_2 \cdots w_k$ where $w_i \in \Sigma^*$ and there is a sequence of states $r_0, r_1, \ldots, r_k$ such that

- $r_0 = q_{\text{start}}$;
- $r_k = q_{\text{accept}}$; and
- for every i, $w_i \in L(R_i)$ where $R_i = \delta(q_{i-1}, q_i)$.

IM NTU

Proof of Lemma.

Let M be the DFA for the regular language. Construct an equivalent GNFA G by adding $q_{\text{start}}, q_{\text{accept}}$ and necessary ϵ-transitions.

CONVERT (G):

1. Let k be the number of states of G.
2. If $k = 2$, then return the regular expression R labeling the transition from q_{start} to q_{accept}.
3. If $k > 2$, select $q_{\text{rip}} \in Q \setminus \{q_{\text{start}}, q_{\text{accept}}\}$. Construct $G' = (Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}})$ where
   - $Q' = Q \setminus \{q_{\text{rip}}\}$;
   - for any $q_i \in Q' \setminus \{q_{\text{accept}}\}$ and $q_j \in Q' \setminus \{q_{\text{start}}\}$, define $\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup R_4$ where $R_1 = \delta(q_i, q_{\text{rip}})$, $R_2 = \delta(q_{\text{rip}}, q_{\text{rip}})$, $R_3 = \delta(q_{\text{rip}}, q_j)$, and $R_4 = \delta(q_i, q_j)$.
4. return CONVERT (G').

□

Regular Expressions and Finite Automata

IM NTU

Lemma

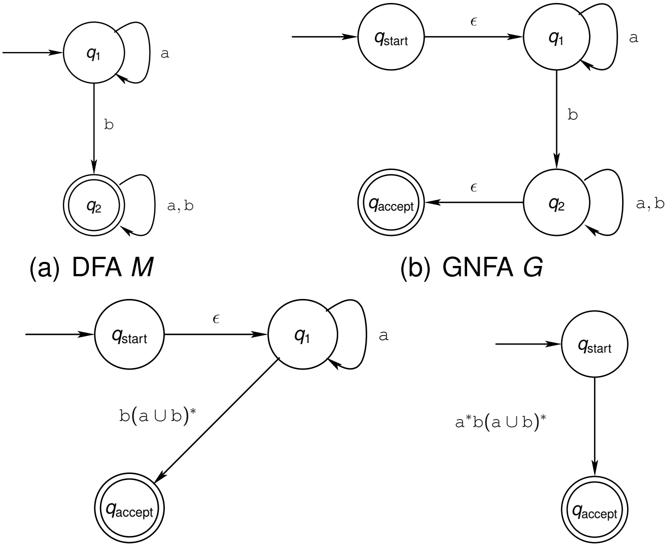
For any GNFA G, CONVERT (G) is equivalent to G.

Proof.

We prove by induction on the number k of states of G.

- $k = 2$. Trivial.
- Assume the lemma holds for $k-1$ states. We first show G' is equivalent to G. Suppose G accepts an input w. Let $q_{\text{start}}, q_1, q_2, \ldots, q_{\text{accept}}$ be an accepting computation of G. We have $q_{\text{start}} \xrightarrow{w_1} q_1 \cdots q_{i-1} \xrightarrow{w_i} q_i \xrightarrow{w_{i+1}} q_{\text{rip}} \cdots q_{\text{rip}} \xrightarrow{w_{j-1}} q_{\text{rip}} \xrightarrow{w_j} q_j \cdots q_{\text{accept}}$. Hence $q_{\text{start}} \xrightarrow{w_1} q_1 \cdots q_{i-1} \xrightarrow{w_i} q_i \xrightarrow{w_{i+1} \cdots w_j} q_j \cdots q_{\text{accept}}$ is a computation of G'. Conversely, any string accepted by G' is also accepted by G since the transition between q_i and q_j in G' describes the strings taking q_i to q_j in G. Hence G' is equivalent to G. By inductive hypothesis, CONVERT (G') is equivalent to G'. □

Regular Expressions and Finite Automata



(a) DFA M

(b) GNFA G

(c) GNFA

(d) GNFA

Figure: Finite Automaton to Regular Expression

Regular Expressions and Finite Automata

IM NTU

Theorem

A language is regular if and only if some regular expression describes it.

Pumping Lemma

IM NTU

Lemma

If A is a regular language, then there is a number p such that for any $s \in A$ of length at least p, there is a partition $s = xyz$ with

1. *for each $i \geq 0$, $xy^iz \in A$; $|y| > 0$; and $|xy| \leq p$.*

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and $p = |Q|$. Consider any string $s = s_1 s_2 \cdots s_n$ of length $n \geq p$. Let $r_1 = q_1, \ldots, r_{n+1}$ be the sequence of states such that $r_{i+1} = \delta(r_i, s_i)$ for $1 \leq i \leq n$. Since $n + 1 \geq p + 1 = |Q| + 1$, there are $1 \leq j < l \leq p + 1$ such that $r_j = r_l$ (why?). Choose $x = s_1 \cdots s_{j-1}$, $y = s_j \cdots s_{l-1}$, and $z = s_l \cdots s_n$.

Note that $r_1 \xrightarrow{x} r_j$, $r_j \xrightarrow{y} r_l$, and $r_l \xrightarrow{z} r_{n+1} \in F$. Thus M accepts xy^iz for $i \geq 0$. Since $j \neq l$, $|y| > 0$. Finally, $|xy| \leq p$ for $l \leq p + 1$. □

Applications of Pumping Lemma

IM NTU

Example

$B = \{\texttt{0}^n\texttt{1}^n : n \geq 0\}$ is not a regular language.

Proof.

Suppose B is regular. Let p be the pumping length given by the pumping lemma. Choose $s = \texttt{0}^p\texttt{1}^p$. Then $s \in B$ and $|s| \geq p$, there is a partition $s = xyz$ such that $xy^iz \in B$ for $i \geq 0$.

- $y \in \texttt{0}^+$ or $y \in \texttt{1}^+$. $xz \notin B$. A contradiction.
- $y \in \texttt{0}^+\texttt{1}^+$. $xyyz \notin B$. A contradiction. □

Applications of Pumping Lemma

Example

$B = \{\texttt{0}^n\texttt{1}^n : n \geq 0\}$ is not a regular language.

Corollary

$C = \{w : w$ has an equal number of $\texttt{0}$'s and $\texttt{1}$'s$\}$ is not a regular language.

Proof.

Suppose C is regular. Then $B = C \cap \texttt{0}^*\texttt{1}^*$ is regular. $\square$

Applications of Pumping Lemma

IM NTU

Example

$F = \{ww : w \in \{0,1\}^*\}$ is not a regular language.

Proof.

Suppose F is a regular language and p the pumping length. Choose $s = 0^p10^p1$. By the pumping lemma, there is a partition $s = xyz$ such that $|xy| \leq p$ and $xy^iz \in F$ for $i \geq 0$. Since $|xy| \leq p$, $y \in 0^+$. But then $xz \notin F$. A contradiction. $\square$

Applications of Pumping Lemma

Example

$D = \{1^{n^2} : n \geq 0\}$ is not a regular language.

Proof.

Suppose D is a regular language and p the pumping length. Choose $s = 1^{p^2}$. By the pumping lemma, there is a partition $s = xyz$ such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in D$ for $i \geq 0$. Consider the strings xyz and xy^2z. We have $|xyz| = p^2$ and $|xy^2z| = p^2 + |y| \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2$. Since $|y| > 0$, we have $p^2 = |xyz| < |xy^2z| < (p+1)^2$. Thus $xy^2z \notin D$. A contradiction. □

Applications of Pumping Lemma

IM NTU

Example

$E = \{0^i 1^j : i > j\}$ is not a regular language.

Proof.

Suppose E is a regular language and p the pumping length. Choose $s = 0^{p+1}1^p$. By the pumping lemma, there is a partition $s = xyz$ such that $|y| > 0$, $|xy| \leq p$, and $xy^iz \in E$ for $i \geq 0$. Since $|xy| \leq p$, $y \in 0^+$. But then $xz \notin E$ for $|y| > 0$. A contradiction. $\square$