Theory of Computation
Regular Languages

Sets

- A set is a group of (possibly infinite) objects; its objects are called elements or members.
- The set without any element is called the empty set (written $\emptyset$).
- Let A, B be sets.
  - $A \cup B$ denotes the union of A and B.
  - $A \cap B$ denotes the intersection of A and B.
  - $\overline{A}$ denotes the complement of A (with respect to some universe U).
  - $A \subseteq B$ denotes that A is a subset of B.
  - $A \subsetneq B$ denotes that A is a proper subset of B.
- The power set of a set A (written 2^A) is the set consisting of all subsets of A.
- If the number of occurrences matters, we use multiset instead.

Sequences and Tuples

- A sequence is a (possibly infinite) list of ordered objects.
- A finite sequence of k elements is also called k-tuple; a 2-tuple is also called a pair.
- The Cartesian product of sets A and B (written $A \times B$) is defined by

$$A \times B = \{(a, b) : a \in A \text{ and } b \in B\}.$$

- We can take Cartesian products of k sets $A_1, A_2, \ldots, A_k$

$$A_1 \times A_2 \times \cdots \times A_k = \{(a_1, a_2, \ldots, a_k) : a_i \in A_i \text{ for every } 1 \leq i \leq k\}.$$

- Define

$$A^k = \overbrace{A \times A \times \cdots \times A}^{k}.$$

Functions and Relations

- A function $f : D \to R$ maps an element in the domain D to an element in the range R.
- Write $f(a) = b$ if f maps $a \in D$ to $b \in R$.
- When $f : A_1 \times A_2 \times \cdots \times A_k \to B$, we say f is a k-ary function and k is the arity of f.
  - When $k = 1$, f is a unary function.
  - When $k = 2$, f is a binary function.
- A predicate or property is a function whose range is $\{0, 1\}$.
- A property with domain $\overbrace{A \times A \times \cdots \times A}^{k}$ is a k-ary relation on A.
  - When $k = 2$, it is a binary relation.
- A binary relation R is an equivalence relation if
  - R is reflexive (for every x, xRx);
  - R is symmetric (for every x and y, xRy implies yRx; and
  - R is transitive (for every x, y, and z, xRy and yRz implies xRz.

More about Sets

A set A is countably infinite if there is a bijection $f : \mathbb{N} \to A$.

Theorem 1

Let $\mathbb{B}$ *be* $\{0, 1\}$. *Then* $A = \mathbb{B} \times \mathbb{B} \times \cdots \times \mathbb{B} \times \cdots$ *is uncountable.*

Proof.

s_1	=	0	0	0	0	0	0	0	0	0	0	0	...
s_2	=	1	1	1	1	1	1	1	1	1	1	1	...
s_3	=	0	1	0	1	0	1	0	1	0	1	0	...
s_4	=	1	0	1	0	1	0	1	0	1	0	1	...
s_5	=	1	1	0	1	0	1	1	0	1	0	1	...
s_6	=	0	0	1	1	0	1	1	0	1	1	0	...
s_7	=	1	0	0	0	1	0	0	0	1	0	0	...
s_8	=	0	0	1	1	0	0	1	1	0	0	1	...
s_9	=	1	1	0	0	1	1	0	0	1	1	0	...
s_{10}	=	1	1	0	1	1	1	0	0	1	0	1	...
s_{11}	=	1	1	0	1	0	1	0	0	1	0	0	...
$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

s	=	1	0	1	1	1	0	1	0	0	1	1	...

Induction Proof

- **Induction Principle**:
  $P(0) \wedge (\forall k, P(k) \Rightarrow P(k+1)) \Rightarrow (\forall n \in \mathbb{N}, P(n))$.
- **Well-founded Relation**:
  A binary R is called well-founded on a class X if every **non-empty subset** $S \subseteq X$ has a **minimal element** with respect to R. (E.g., $\mathbb{N}$ is well-founded; $\mathbb{Z}$ is not well-founded.)

Induction Principle $\Leftrightarrow$ $(\mathbb{N}, <)$ **is well-founded.**

To prove property $P(n)$ holds for all $n \in \mathbb{N}$,

- **(Induction Basis)**: Prove $P(0)$;
- **(Induction Step)**: Prove that if $P(k)$ holds, then $P(k+1)$ also holds.

Strings and Languages

- An alphabet is a nonempty finite set.
- Members of an alphabet are called symbols.
- A string over an alphabet is a finite sequence of symbols from the alphabet.
- If w is a string over an alphabet Σ, the length of w (written $|w|$) is the number of symbols in w.
- The string of length zero is the empty string.
- Let $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_m$ be strings of length n and m respectively. The concatenation of x and y (written xy) is the string $x_1 x_2 \cdots x_n y_1 y_2 \cdots y_m$ of length $n + m$.
- For any string x, $x^k = \overbrace{xx \cdots x}^{k}$.
- A language is a set of strings.

Schematic of Finite Automata

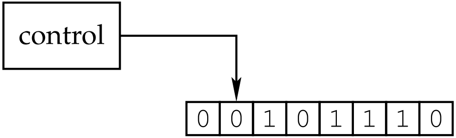


Figure: Schematic of Finite Automata

- 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

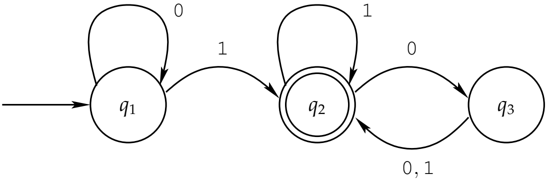
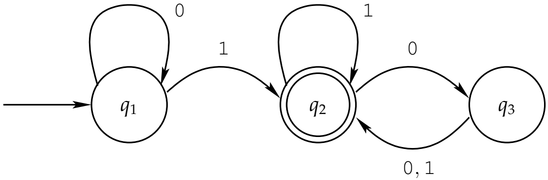


Figure: A Finite Automaton M_1

The above figure shows 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
  - $Q = \{q_1, q_2, q_3\}$;
  - $\Sigma = \{\texttt{0}, \texttt{1}\}$;
  - $\delta : Q \times \Sigma \to Q$ 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

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

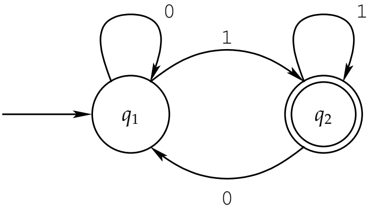


Figure: Finite Automaton M_2

- The above figure shows $M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$ 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_2

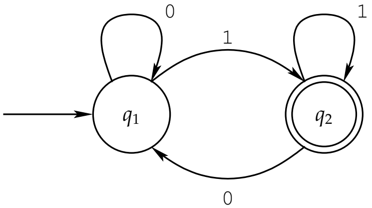


Figure: Finite Automaton M_2

- The above figure shows $M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$ 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

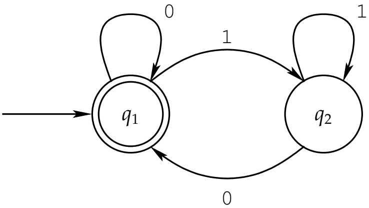


Figure: Finite Automaton M_3

- The above figure shows $M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$ 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_3

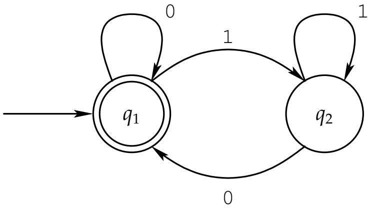


Figure: Finite Automaton M_3

- The above figure shows $M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$ 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 $0\}$.

Computation – Formal Definition

- 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

$$r_0 \xrightarrow{w_1} r_1 \xrightarrow{w_2} r_2 \cdots r_{n-1} \xrightarrow{w_n} r_n,$$

  - $r_0 = q_0$;
  - $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$; and
  - $r_n \in F$,
- M recognizes language A if $A = \{w : M \text{ accepts } w\}$.

Definition 2

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

Regular Operations

Definition 3

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: $A \cdot B = \{xy : x \in A \text{ and } y \in B\}$.
- 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

Theorem 4

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

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

- 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

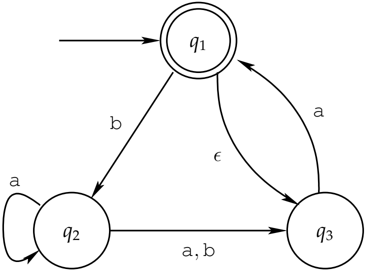


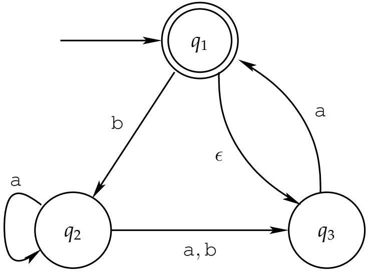
Figure: NFA N_4

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

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

$$r_0 \xrightarrow{y_1} r_1 \xrightarrow{y_2} r_2 \cdots r_{m-1} \xrightarrow{y_m} r_m,$$

  - $r_0 = q_0$;
  - $r_{i+1} \in \delta(r_i, y_{i+1})$ 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: $\texttt{a}, \texttt{baa}, \ldots$.

Equivalence of NFA's and DFA's

Theorem 5

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

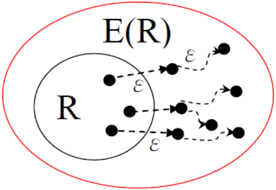
- $Q' = \mathcal{P}(Q)$;
- $\delta'(R, a) = \{q \in Q : q \in E(\delta(r, a))$ for some $r \in R\}$;
- $q'_0 = E(\{q_0\})$;
- $F' = \{R \in Q' : R \cap F \neq \emptyset\}$.

□

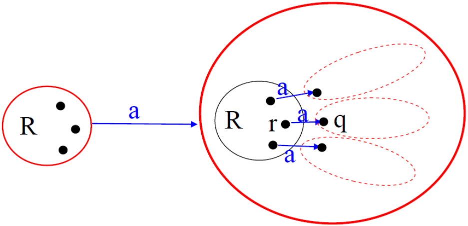
- Why is $L(M) = L(N)$?

Equivalence of NFA's and DFA's

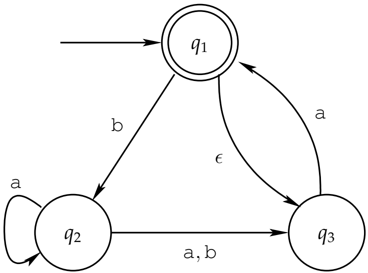
- ϵ-closure $E(R)$:

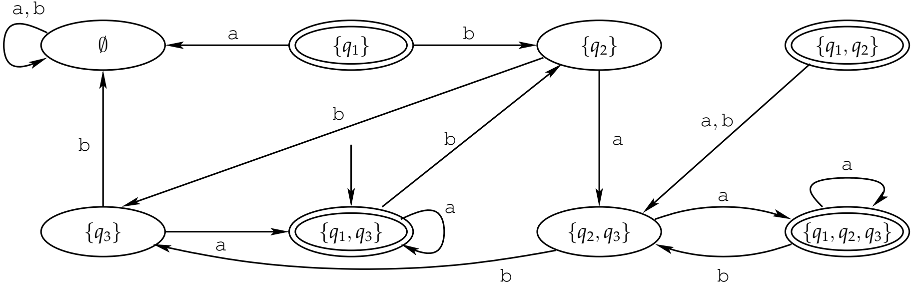


- Transition $\delta'(R, a) = \{q \in Q : q \in E(\delta(r, a))\}$



A DFA Equivalent to N_4





Closure Properties – Revisited

Theorem 6

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

□

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

Closure Properties – Revisited

Theorem 7

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

□

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

Closure Properties – Revisited

Theorem 8

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

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

Closure Properties – Revisited

Theorem 9

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. □

Regular Expressions (Syntax)

Definition 10

R is a regular expression if R is

- a for some $a \in \Sigma$;
- ϵ;
- $\emptyset$;
- $(R_1 + R_2)$ where R_1 and R_2 are regular expressions;
- $(R_1 \cdot R_2)$ where R_1 and R_2 are regular expressions; or
- (R_1^*) where R_1 is a regular expression.

- We write R^+ for $R \cdot R^*$. Hence $R^* = R^+ + \epsilon$.
- Moreover, write R^k for $\overbrace{R \cdot R \cdot \cdots \cdot R}^{k}$.
  - Define $R^0 = \epsilon$. We have $R^* = R^0 + R^1 + \cdots + R^n + \cdots$.
- $L(R)$ denotes the language described by the regular expression R.
- Note that $\emptyset \neq \{\epsilon\}$. $+$ is also written as "$\cup$" is many textbooks

Regular Expressions (Semantics)

Definition 11

The language associated with a regular expression R, written as $L(R)$, is defined recursively as

- $L(a) = \{a\}, a \in \Sigma$;
- $L(\epsilon) = \{\epsilon\}$;
- $L(\emptyset) = \emptyset$;
- $L(R_1 + R_2) = L(R_1) \cup L(R_2)$
- $L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2)$
- $L(R_1^*) = (L(R_1))^*$

Examples of Regular Expressions

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

Examples of Regular Expressions

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

Examples of Regular Expressions

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

Examples of Regular Expressions

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

Examples of Regular Expressions

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

Examples of Regular Expressions

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

Examples of Regular Expressions

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

Regular Expressions and Finite Automata

Lemma 12

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

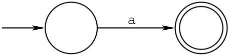
Proof.

We prove by induction on the regular expression R.

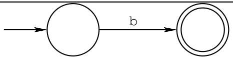
- $R = a$ for some $a \in \Sigma$. Consider the NFA $N_a = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$ where
  $$\delta(r, y) = \begin{cases} \{q_2\} & r = q_1 \text{ and } y = a \\ \emptyset & \text{otherwise} \end{cases}$$
- $R = \epsilon$. Consider the NFA $N_\epsilon = (\{q_1\}, \Sigma, \delta,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

Regular Expressions and Finite Automata

a

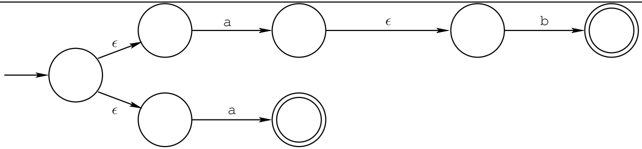


b

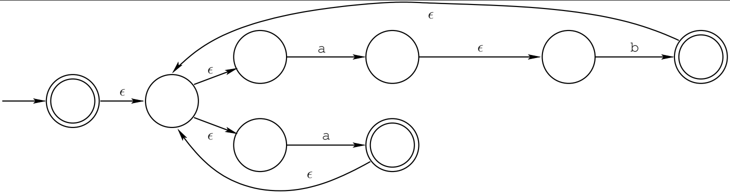


ab

ab + a



(ab + a)*



Lemma 13

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 14

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 $R_i = \delta(q_{i-1}, q_i)$.

Regular Expressions and Finite Automata

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

Lemma 15

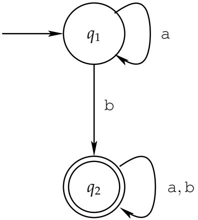
For any GNFA G, CONVERT (G) is equivalent to G.

Proof.

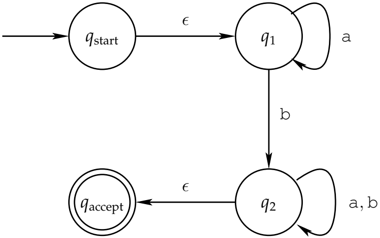
We prove by induction on the number k of states of G.

- $k = 2$. 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'. □

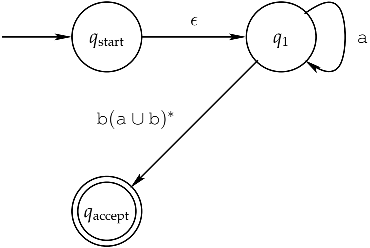
Regular Expressions and Finite Automata



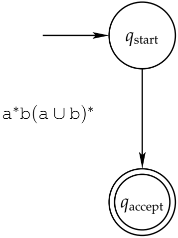
(a) DFA M



(b) GNFA G



(c) GNFA



(d) GNFA

Theorem 16

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

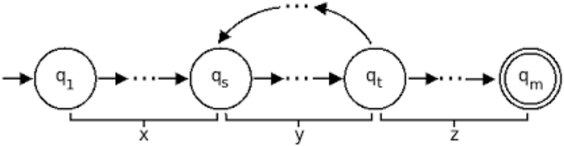
Pumping Lemma

Lemma 17

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

Proof Idea:



Pumping Lemma (Proof)

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and $p = |Q|$.
Consider any string $s = \sigma_1\sigma_2 \cdots \sigma_{m-1}$ of length $m - 1 \geq p$. Let $q_1, \ldots, q_m$ be the sequence of states such that $q_{i+1} = \delta(q_i, \sigma_i)$ for $1 \leq i \leq m - 1$.
Since $m \geq p + 1 = |Q| + 1$, there are $1 \leq s < t \leq p + 1$ such that $q_s = q_t$ (why?). Let $x = \sigma_1 \cdots \sigma_{s-1}$, $y = \sigma_s \cdots \sigma_{t-1}$, and $z = \sigma_t \cdots \sigma_{m-1}$.
Note that $q_1 \xrightarrow{x} q_s$, $q_s \xrightarrow{y} q_t$, and $q_t \xrightarrow{z} q_m \in F$. Thus M accepts xy^iz for $i \geq 0$. Since $t \neq s$, $|y| > 0$. Finally, $|xy| \leq p$ for $t \leq p + 1$. □

Applications of Pumping Lemma

Example 18

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

Proof.

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

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

Corollary 19

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

Proof.

Suppose C is regular. Then $B = C \cap 0^* 1^*$ is regular. □

Applications of Pumping Lemma

Example 20

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

Proof.

Suppose F is a regular language and p the pumping length. Choose $s = \mathtt{0}^p\mathtt{1}\mathtt{0}^p\mathtt{1}$. 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 $|xy| \leq p$, $y \in \mathtt{0}^+$. But then $xz \notin F$. A contradiction. □

Applications of Pumping Lemma

Example 21

$D = \{1^{n^2} : n \geq 0\}$ 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

Example 22

$E = \{0^i 1^j : i > j\}$ 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. □