

# Discrete Mathematics

**Formal Languages** — Spring 2025

Konstantin Chukharev

# §1 Formal Languages

# Basic Terminology

**Definition 1:** *Alphabet*  $\Sigma$  is a finite non-empty set of symbols.

*Examples:*  $\Sigma_1 = \{a, b, c\}$ ,  $\Sigma_2 = \{0, 1\}$ ,  $\Sigma_3 = \{\text{🕷️}, \text{🐱}, \text{🐼}, \text{🦁}\}$ .

**Definition 2:** A *word*, or a *string*, over  $\Sigma$  is a *finite* sequence of symbols from  $\Sigma$ .

*Examples:* “abacaba”, “10110001”, “i am a word”, “” (empty word  $\varepsilon$ ).

**Definition 3:** The set of *all* finite words over the alphabet  $\Sigma$  is called the *Kleene star*,  $\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$ .

**Definition 4:** A *formal language*  $L \subseteq \Sigma^*$  is a set of finite words over a finite alphabet.

*Examples:*  $L_1 = \{0, 001, 0001, \dots\}$ ,  $L_2 = \{a, aba, ababa, abababa, \dots\}$ ,  $L_3 = \emptyset$ ,  $L_4 = \{\varepsilon, \text{ricercar}\}$ .

# Operations of Languages

- A formal language,  $L \subseteq \Sigma^*$ , can be defined by:
  - ▶ a *enumeration* of words, e.g.  $L = \{w_1, w_2, \dots, w_n\}$
  - ▶ a *regular expression*, e.g.  $L \triangleq 01^*$
  - ▶ a *formal grammar*, e.g.  $L \cong G$
- *Set-theoretic* operations:
  - ▶  $L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\}$ , the *union* of  $L_1$  and  $L_2$
  - ▶  $\overline{L} = \{w \mid w \notin L\} = \Sigma^* \setminus L$ , the *complement* of  $L$
  - ▶  $|L|$  is the *cardinality* of  $L$
- *Concatenation*:
  - ▶  $L_1 \cdot L_2 = \{ab \mid a \in L_1, b \in L_2\}$ , where  $ab$  is the concatenation of words  $a$  and  $b$ .
  - ▶  $L^k = \underbrace{L \cdot \dots \cdot L}_{k \text{ times}} = \underbrace{\{ww\dots w \mid w \in L\}}_{k \text{ words}}$
  - ▶  $L^0 = \{\varepsilon\}$
- *Kleene star*.  $L^* = \bigcup_{k=0}^{\infty} L^k$

# Regular Languages

**Definition 5:** A class of regular languages REG is defined inductively:

- $\text{Reg}_0 = \{\emptyset, \{\varepsilon\}\} \cup \{\{a\} \mid a \in \Sigma\}$ , the *empty* and *singleton* languages.
- $\text{Reg}_{i+1} = \text{Reg}_i \cup \{A \cup B \mid A, B \in \text{Reg}_i\} \cup \{A \cdot B \mid A, B \in \text{Reg}_i\} \cup \{A^* \mid A \in \text{Reg}_i\}$ , the inductively extended  $(i+1)$ -th *generation* of regular languages.
- $\text{REG} = \bigcup_{k=0}^{\infty} \text{Reg}_k$ , the *class* of all regular languages.

**Theorem 1:** REG is closed under union, concatenation, and Kleene star operations.

**Proof:** Let  $A \in \text{Reg}_i$ ,  $B \in \text{Reg}_j$ .

- $(A \cup B) \in (\text{Reg}_i \cup \text{Reg}_j) \in \text{Reg}_{\max(i,j)+1} \subseteq \text{REG}$
- $(A \cdot B) \in (\text{Reg}_i \cdot \text{Reg}_j) \in \text{Reg}_{\max(i,j)+1} \subseteq \text{REG}$
- $A^* \in \text{Reg}_{i+1} \subseteq \text{REG}$

□

## Regular Expressions

Language	Expression	Description
$\emptyset$		Empty language
$\{\varepsilon\}$	$\varepsilon$	Language with a single empty word
$\{a\}$	$a$	Singleton language with a literal character “a”
$A$	$\alpha$	Language $A$ denoted by regex $\alpha$
$B$	$\beta$	Language $B$ denoted by regex $\beta$
$A \cup B$	$\alpha \mid \beta$	Union of languages $A$ and $B$
$A \cdot B$	$\alpha\beta$	Concatenation of languages $A$ and $B$
$A^*$	$\alpha^*$	Kleene star of language $A$
$A^+$	$\alpha^+$	Kleene plus of language $A$

*Example:*  $(a \mid bc)^* = \{\varepsilon, a, aa, aaa, \dots, bc, bcbc, bcbcbc, \dots, abc, bca, abca, abcbc, bcabc, \dots\}$

See also: PCRE [☒](#)

## §2 Automata

# Deterministic Finite Automata

**Definition 6:** Deterministic Finite Automaton (DFA) is a 5-tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$  where:

- $Q$  is a *finite* set of states,
- $\Sigma$  is an *alphabet* (finite set of input symbols),
- $\delta : Q \times \Sigma \longrightarrow Q$  is a *transition function*,
- $q_0 \in Q$  is the *start* state,
- $F \subseteq Q$  is a set of *accepting* states.

DFAs recognize *regular* languages (Type 3).

*Example:* Automaton  $\mathcal{A}$  recognizing strings with an even number of 0s,  $\mathcal{L}(\mathcal{A}) = \{0^n \mid n \text{ is even}\}$ .

	0	1
q0	q1	q0
q1	q0	q1



Here,  $q_0$  is the *start* (denoted by an arrow) and also the *accepting* (denoted by double circle) state.



## Exercises

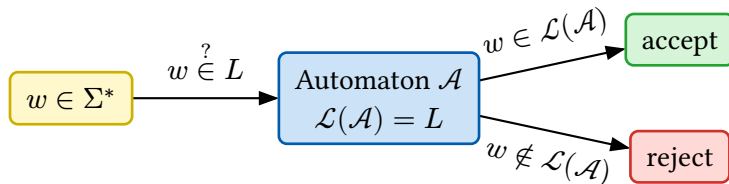
For each language below (over the alphabet  $\Sigma = \{0, 1\}$ ), draw a DFA recognizing it:

1.  $L_1 = \{101, 110\}$
2.  $L_2 = \Sigma^* \setminus \{101, 110\}$
3.  $L_3 = \{w \mid w \text{ starts and ends with the same bit}\}$
4.  $L_4 = \{110\}^* = \{\varepsilon, 110, 110110, 110110110, \dots\}$
5.  $L_5 = \{w \mid w \text{ contains } 110 \text{ as a substring}\}$

# Recognizers vs Transducers

There are two main types of finite-state machines:

1. *Acceptors* (or *recognizers*), automata that produce a binary *yes/no answer*, indicating whether or not the received input word  $w \in \Sigma^*$  is *accepted*, i.e., belongs to the language  $L$  recognized by the automaton.



1. *Transducers*, machines that produce an output action *for each* symbol of an input.
  - Moore machines (1956)
  - Mealy machines (1955)

# Computation

**Definition 7:** A process of *computation* by a finite-state machine  $\mathcal{A}$  is a finite sequence of *configurations*, or *snapshots*. A set of all possible configurations is denoted  $\text{SNAP} = Q \times \Sigma^*$ .

**Definition 8:** A *reachability relation*  $\vdash$  is a binary relation over configurations:

$$\langle q, \alpha \rangle \vdash \langle r, \beta \rangle \quad \text{iff} \quad \begin{cases} \alpha = c\beta & \text{where } c \in \Sigma \\ r = \delta(q, c) \end{cases}$$

- $c_1 \vdash c_2$  means “configuration  $c_2$  is reachable in *one step* from  $c_1$ ”.
- $\vdash^*$ , the reflexive-transitive closure of  $\vdash$ , denotes “reachable in *any* number of steps”.

## Automata Languages

**Definition 9:** A word  $w \in \Sigma^*$  is *accepted* by an automaton  $\mathcal{A}$  if the computation, starting in the initial configuration at state  $q_0$  with input  $w$ , *can reach the final configuration*  $\langle f, \varepsilon \rangle$ , where  $f \in F$  is any accepting state, and  $\varepsilon$  denotes that the input has been fully consumed.

Formally,  $\mathcal{A}$  *accepts*  $w \in \Sigma^*$  if  $\langle q_0, w \rangle \vdash^* \langle f, \varepsilon \rangle$  for some  $f \in F$ .

**Definition 10:** The language *recognized* by an automaton  $\mathcal{A}$  is a set of all words accepted by  $\mathcal{A}$ .

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \langle q_0, w \rangle \vdash^* \langle f, \varepsilon \rangle \text{ where } f \in F\}$$

**Definition 11:** The class of *automaton languages* recognized by DFAs is denoted AUT.

$$\text{AUT} = \{X \mid \exists \mathcal{A} \text{ such that } \mathcal{L}(\mathcal{A}) = X\}$$

## Kleene's Theorem

**Theorem 2:**  $\text{REG} = \text{AUT}$ .

**Proof:** See the next lecture!



## §3 Extra slides

# Chomsky Hierarchy

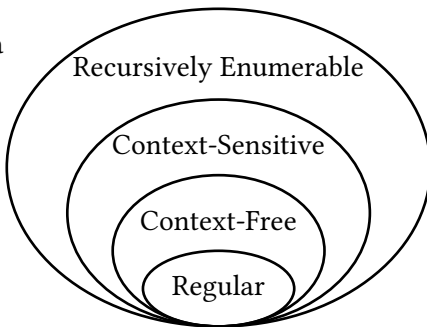
**Definition 12** (Formal language): A set of strings over an alphabet  $\Sigma$ , closed under concatenation.

Formal languages are classified by *Chomsky hierarchy*:

- Type 0: Recursively Enumerable – Turing Machines
- Type 1: Context-Sensitive – Linear TMs
- Type 2: Context-Free – Pushdown Automata
- Type 3: Regular – Finite Automata

*Examples:*

- $L = \{a^n \mid n \geq 0\}$
- $L = \{a^n b^n \mid n \geq 0\}$
- $L = \{a^n b^n c^n \mid n \geq 0\}$
- $L = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input } w\}$



Noam Chomsky