Formal Languages

Discrete Math, Spring 2025

Konstantin Chukharev

Formal Languages

Basic Terminology

Definition 1: *Alphabet* Σ is a finite non-empty set of symbols.

Examples: $\Sigma_1 = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, $\Sigma_2 = \{0, 1\}$, $\Sigma_3 = \{$🦀, 🐈, 🦭, 🦁$\}$.

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

Examples: "abacaba", "10110001", "i am a word", "" (empty word ε).

Definition 3: The set of *all* finite words over the alphabet Σ 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: $L_1 = \{0, 001, 0001, \ldots\}$, $L_2 = \{\mathrm{a}, \mathrm{aba}, \mathrm{ababa}, \mathrm{abababa}, \ldots\}$, $L_3 = \varnothing$, $L_4 = \{\varepsilon, \mathrm{ricercar}\}$.

Operations of Languages

- A formal language, $L \subseteq \Sigma^*$, can be defined by:
  - a *enumeration* of words, *e.g.* $L = \{w_1, w_2, \ldots, w_n\}$
  - a *regular expression*, *e.g.* $L \triangleq \texttt{01*}$
  - a *formal grammar*, *e.g.* $L \cong G$
- *Set-theoretic* operations:
  - $L_1 \cup L_2 = \{w \mid w \in L_1 \lor w \in L_2\}$, 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.
  - $L^k = \underbrace{L \cdot \ldots \cdot L}_{k \text{ times}} = \{\underbrace{ww\ldots w}_{k \text{ words}} \mid w \in L\}$
  - $L^0 = \{\varepsilon\}$
- *Kleene star*: $L^* = \bigcup_{k=0}^{\infty} L^k$

Regular Languages

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

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

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

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

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

$\square$

Regular Expressions

Language	Expression	Description
$\varnothing$		Empty language
$\{\varepsilon\}$	ε	Language with a single empty word
$\{\mathrm{a}\}$	`a`	Singleton language with a literal character "a"
A	α	Language A denoted by regex α
B	β	Language B denoted by regex β
$A \cup B$	$\alpha \mid \beta$	Union of languages A and B
$A \cdot B$	$\alpha\beta$	Concatenation of languages A and B
A^*	α^*	Kleene star of language A
A^+	α^+	Kleene plus of language A

Example: `(a|bc)*` $= \{\varepsilon, \mathrm{a}, \mathrm{aa}, \mathrm{aaa}, \ldots, \mathrm{bc}, \mathrm{bcbc}, \mathrm{bcbcbc}, \ldots, \mathrm{abc}, \mathrm{bca}, \mathrm{abca}, \mathrm{abcbc}, \mathrm{bcabc}, \ldots\}$

See also: PCRE ⧉

Automata

Deterministic Finite Automata

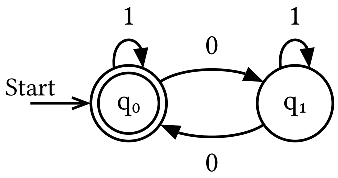
Definition 6: Deterministic Finite Automaton (DFA) is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where:

- Q is a *finite* set of states,
- Σ is an *alphabet* (finite set of input symbols),
- $\delta : Q \times \Sigma \to Q$ 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

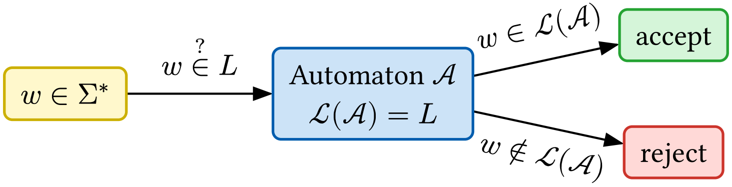
For each language below (over the alphabet $\Sigma = \{0, 1\}$), draw a DFA recognizing it:

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. $L_4 = \{110\}^* = \{\varepsilon, 110, 110110, 110110110, \ldots\}$
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 recieved 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 $\mathrm{SNAP} = Q \times \Sigma^*$.

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 ε 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.

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

Kleene's Theorem

Theorem 2: $\mathrm{REG} = \mathrm{AUT}$.

Proof: See the next lecture! $\square$

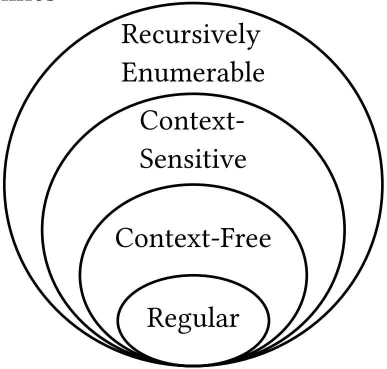
Extra slides

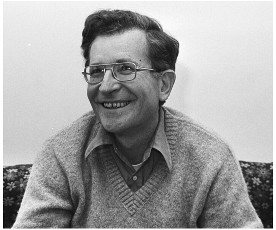
Chomsky Hierarchy

Definition 12 (Formal language): A set of strings over an alphabet Σ, 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





Noam Chomsky

Examples:

- $L = \{a^n \mid n \geq 0\}$ is regular.
- $L = \{a^n b^n \mid n \geq 0\}$ is context-free.
- $L = \{a^n b^n c^n \mid n \geq 0\}$ is context-sensitive.
- $L = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input } w\}$ is recursively enumerable.