CSC236H
Introduction to the Theory of Computation

Alphabets and Strings

- **Alphabet:** a finite, non-empty set of atomic symbols. (denote by Σ)
  Example: a, b, c, 0, 1, $+$
- To prevent ambiguity, compound symbols like ab are not allowed.
- **String:** A finite sequence of symbols is called a string.
- Empty sequence is also allowed and denoted by ϵ (called **empty** or **null** string).
  To avoid confusion with empty string, ϵ is not allowed as symbols in an alphabet.
  Example:
  - $a, ab, cccc$ are strings over $\{a, b, c\}$.
  - $a + 00$ is not a string over $\{a, b, c\}$ but it is over $\{0, 1, +, a, b, c\}$.
- The set of all strings over alphabet Σ is denoted by Σ^*.
- A string on an alphabet Σ is a member of Σ^*.

- **Language:** A set of strings.
  A language can be empty, finite or infinite.
  Example:
  - $\{bab, bbabb, bbbabbb, \cdots\}$ is a language over $\{a, b, c\}$.
  - $\{\epsilon\}$ is a language over any alphabet.
  - $\{\}$ is a language over any alphabet.
- $\{\}$ is different from $\{\epsilon\}$.
  $\{\}$ contains NO string, but $\{\epsilon\}$ contains ONE string (i.e., the empty string ϵ).

Operations Over Strings

- **Length** of string s: the number of symbols in s. Denoted by $|s|$.
  Example:
  - $|bba| = 3$
  - $|a| = 1$
  - $|\epsilon| = 0$
- Strings s and t are **equal** iff $|s| = |t|$ and $s_i = t_i$, for all $1 \leq i \leq n$ where $n = |s|$ and v_i denotes $i-$th symbol in string v.
- **Reversal** of string s: a string obtained by reversing the order of symbols in s. Denoted by s^R.
  Example:
  - $1011^R = 1101$
  - $aaa^R = aaa$
  - $\epsilon^R = \epsilon$

- **Concatenation** of strings s and t: a string consists of every symbol of s followed by every symbol of t. Denoted by st or $s \circ t$.
  Example:
  - $bba \circ bb = bbabb$
  - $\epsilon \circ abc = abc$
- For string s, and natural number k, s^k denotes $\underline{k \text{ times concatenation of } s}$ with itself.
  Example:
  - $aba^2 = abaaba$
  - $aaa^0 = \epsilon$
  - $\epsilon^3 = \epsilon$.

Operations Over Strings

- For alphabet Σ, Σ^n denotes set of all strings of length n over Σ, and Σ^* denotes the set of all strings over Σ.
  Example:
  - $\{a, b, c\}^0 = \{\epsilon\}$
  - $\{0, 1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
  - $\{1\}^* = \{\epsilon, 1, 11, 111, ...\} = \{1^k : k \in \mathbb{N}\}$

Operations Over Strings

- **Prefix:** A string x is a prefix of string y if there exist a string x' (possibly ϵ) such that $xx' = y$.

- **Suffix:** A string x is a suffix of string y if there exist a string x' (possibly ϵ) such that $x'x = y$.

Let L, L' be languages over alphabet Σ:

- **Complementation:** $\overline{L} = \Sigma^* - L$.
  Example:
  - If $L = \{0x : x \in \{0,1\}^*\} = \{0, 00, 01, 000, 001, \cdots\}$, then $\overline{L} = \{\epsilon\} \cup \{1x : x \in \{0,1\}^*\}$
- **Union:** $L \cup L' = \{x : x \in L \text{ or } x \in L'\}$

Operations Over Languages

Let L, L' be languages over alphabet Σ:

- **Intersection:** $L \cap L' = \{x : x \in L \text{ and } x \in L'\}$

- **Reversal:** $Rev(L) = \{x^R : x \in L\}$
  - $Rev(\{a, ab, abb\}) = \{a, ba, bba\}$

Let L, L' be languages over alphabet Σ:

- **Concatenation:** $L \circ L' = \{s \in \Sigma^* : s = r \circ t$ for $r \in L, t \in L'\}$.
  Example:
  - $\{a, bc\} \circ \{bb, c\} = \{abb, ac, bcbb, bcc\}$
  - $\{a, aa, aaa, \cdots\} \circ \{b, bb, bbb, \cdots\} = \{ab, abb, abbb, \cdots, aab, aabb, \cdots\} =$ $\{s \in (a)^*(b)^* : s$ contains some number of a's followed by some number of b's, with at least one of each$\}$.
  - For all L, $L \circ \{\epsilon\} = L = \{\epsilon\} \circ L$.
  - For all L, $L \circ \{\} = \{\} = \{\} \circ L$.

Let L, L' be languages over alphabet Σ:

- **Exponentiation:** $L^k = L \circ L \circ \cdots \circ L$.
  Example:
  - $\{1, 11, 111\}^0 = \{\epsilon\}$.
  - $\{\epsilon\}^5 = \{\epsilon\}$.
  - $\{\}^4 = \{\}$.
  - $\{\}^0 = \{\epsilon\}$.

Let L, L' be languages over alphabet Σ:

- **Kleene star:** $L^{\circledast} = L^0 \cup L^1 \cup L^2 \cup \cdots = \{s : s \in L^k, k \in \mathbb{N}\}$.
  Example:
  - $\{ab\}^{\circledast} = \{\epsilon, ab, abab, ababab, \cdots\}$.
  - $\{\epsilon\}^{\circledast} = \{\epsilon\}$.
  - $\{\}^{\circledast} = \{\epsilon\}$.

The set of **regular languages** over an alphabet Σ is defined recursively as follows:

- $\{\}$ is a regular language.
- $\{\epsilon\}$ is a regular language.
- For any symbol $a \in \Sigma$, $\{a\}$ is a regular language.
- If L, M are regular languages, then so are $L \cup M, L \circ M$, and $L^{\circledast}$.

Many Problems can be reduced to languages.
Examples:

- Logical Formulas.
- Program Compilation.
- Natural Language Processing.

Many Problems can be reduced to languages.
Examples:

- Logical Formulas.
- Program Compilation.
- Natural Language Processing.

Language Recognition Problem:

- Given language L and string s, does s belong to L?

Language Recognition

Many Problems can be reduced to languages.
Examples:

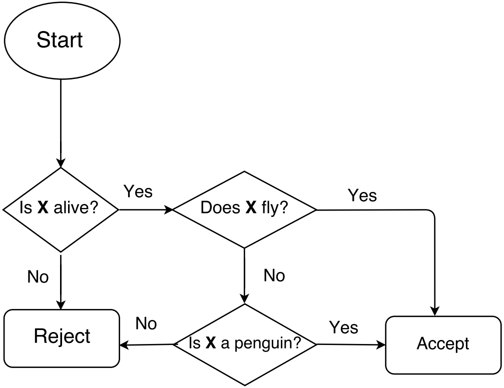
- Logical Formulas.
- Program Compilation.
- Natural Language Processing.

Language Recognition Problem:

- Given language L and string s, does s belong to L?

Regular Languages may be describe by
- Procedurally (E.g., **Finite State Automata**)
- Descriptive Generators (E.g., **Regular Expressions**)

Bird Recognition: Is the given object, X, a bird?



Language Recognition – Example

Let $L = \{s \in \{a, b, c\}^* : s$ includes at least one $a\}$.
Among the following strings, determine those that are members of L.

- $bbbccc$
- $bbbaccc$
- abb
- $b1aa$

Very informally, a **Finite State Automaton** consists of

- a set of **states**;
- a set of rules (called **transition rules**) for transition between states based on the input.
- A designated **initial state**.
- A set of designated **accepting states**.

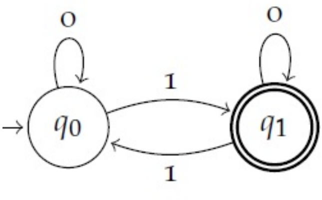
Let $L = \{0s : s \in \{0,1\}^*\}$.
Give a DFA which **only accepts** strings in L.

A **deterministic finite automaton (DFA)** $\mathcal{D}$ is a quintuple $\mathcal{D} = \langle Q, \Sigma, \delta, s, F \rangle$ where:

- Q is the **set of states** in $\mathcal{D}$;
- Σ is the **alphabet** of symbols used by $\mathcal{D}$;
- $\delta : Q \times \Sigma \to Q$ is the **transition function**;
- $s \in Q$ is the **initial state** of $\mathcal{D}$;
- $F \subseteq Q$ is the set of **accepting states** of $\mathcal{D}$.

Let $L = \{0s1 : s \in \{0,1\}^*\}$.
Give a DFA which **only accepts** strings in L.

Describe the language that the following DFA accepts



- DFAs read strings one letter at a time, from left to right.
- At a particular state, there is exactly one transition rule for each symbol in the alphabet.
- Inputs to DFAs can be any length.
- DFAs cannot go back and reread previous letters.
- DFAs have a finite amount of memory, since they have a finite number of states.
  In other words, DFAs have limited memory.

Some problem to think about (optional)

Design a DFA for controlling a vending machine that accepts only nickels (5¢), dimes (10¢) and quarters (25¢), and everything that it sells costs exactly 30¢.