CS304 - Formal Languages and Automata Theory 4^{th} **Semester**

Lecture 2 : Finite State Automata

Instructor: Prof. Prateek Vishnoi *Indian Institute of Technology, Mandi*

We will study the set membership of the formal languages. First we define the meaning of the computability of a formal language. We call a language to be computable if for all strings we can determine correctly whether it belongs to language or not.

Before we proceed, we need to define a model of computation.

Deterministic Finite Automata(DFA)

Definition 1. A *deterministic finite automaton* is a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, F),$$

where

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_0 \in Q$ is the initial state, and
- $F \subseteq Q$ is the set of accepting states.

An input is a string stored on a tape (sequence of cells where each cell contains a symbol). A DFA reads the input one cell at a time and moves only to the right along the tape until the input is exhausted.

A language L is computable (acceptable) by a DFA M if, after reading the string $w \in L$, the automaton M halts(ends) in a final (accepting/final) state and for every $w \notin L$, M halts(ends) in a non-final state.

Example 1. Consider the finite automaton

$$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\}),$$

where the transition function δ is defined as follows:

$$\delta(q_0, 0) = q_0, \quad \delta(q_0, 1) = q_1, \quad \delta(q_1, 0) = q_0, \quad \delta(q_1, 1) = q_1.$$

This automaton accepts all binary strings that end with the symbol 1.

Example 2. Let

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\}),$$

where the transition function δ is defined by

$$\begin{aligned} \delta(q_0, a) &= q_1, & \delta(q_0, b) &= q_0, \\ \delta(q_1, a) &= q_1, & \delta(q_1, b) &= q_2, \\ \delta(q_2, a) &= q_1, & \delta(q_2, b) &= q_0. \end{aligned}$$

This DFA accepts all strings over $\{a, b\}$ that end with the string ab.

Regular Language

Definition 2. A language L is called *regular* if there exists a deterministic finite automaton M such that

$$L = L(M).$$

where $L(M)$ denotes the language accepted by the automaton M.

Closure properties of regular language

A class of languages is closed under an operation if applying the operation to languages in the class results in a language still in the class.

Theorem 3. If $L \subseteq \Sigma^*$ is regular, then $\overline{L} = \Sigma^* \setminus L$ is regular.

Proof. We can easily prove this by "Proof by construction". If a language L is regular then there exists a DFA M which accepts it. Simply change the final state to non-final states and vice-versa. This little modification actually creates a M' which accepts $\overline{L}$. Detailed explanation can be found in [1].

□

Theorem 4. If L_1 and L_2 are regular languages, then

$$L_1 \cup L_2$$

is also a regular language.

Proof. We can't think like this that we will run the input string on both the DFA's of L_1 and L_2 and if the string is accepted by any one of DFA then it accepts. But why so? Because our model of computation is such that we are only allowed to parse the input from left to right one at a time, and we must give the decision whether it accepts or reject the input as it ends. Now we give one DFA that keep tracks of both DFAs of L_1 and L_2.

Since L_1 and L_2 are regular, there exist DFAs that recognize them. We construct a new DFA using the **product automaton** that simulates both DFAs simultaneously and accepts if *either* machine accepts.

Let

$$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$

be a DFA recognizing L_1.

Let

$$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

be a DFA recognizing L_2.

Construction (Product Automaton)

We construct DFA

$$M = (Q, \Sigma, \delta, q_0, F)$$

recognizing $L_1 \cup L_2$ as follows:

- States:
$$Q = Q_1 \times Q_2$$
Each state represents the current state of both DFAs.
- Alphabet:
$$\Sigma \text{ (same alphabet)}$$
- Start state:
$$q_0 = (q_1, q_2)$$
- Transition function:
$$\delta((p, r), a) = (\delta_1(p, a), \delta_2(r, a))$$
- Accepting states:
$$F = \{(p, r) \in Q_1 \times Q_2 \mid p \in F_1 \text{ OR } r \in F_2\}$$

□

Why This Works

The new DFA processes the input string on both machines in parallel.

After reading a string w:

$$\delta^*((q_1, q_2), w) = (\delta_1^*(q_1, w), \delta_2^*(q_2, w))$$

The string is accepted if:

- M_1 accepts w, OR
- M_2 accepts w

Thus,

$$L(M) = L_1 \cup L_2$$

Theorem 5. If L_1 and L_2 are regular languages, then

$$L_1 \cap L_2$$

is also a regular language.

Proof. This is just a little variation of the above automaton. I expect that you complete it from yourself. □

References

[1] Michael Sipser, *Introduction to the Theory of Computation*, 3rd Edition, Cengage Learning, 2012.