Formal Languages and Automata Theory

Rana Barua

Visiting Scientist
IAI, TCG CRES, Kolkata

1 Finite State Machines

A finite state machine also known as **finite state automaton** is a model for computers with very low memory. It consists of a finite tape with cells with a tape-head and a finite control. The input string $w = a_1 \ldots a_n$ is placed onto the tape with the ith letter occupying the ith cell.The automaton starts with the finite control at the initial state q_0 with the tape-head scanning the content of the first cell.

| a_1 | a_2 |......| a_n |
− − − − − − − − − − − − −
. ↑
− − −
| q_0 |
− − −

Depending on the letter being scanned by the tape-head and the state of the finite control, the automaton moves the tape-head one cell to the right and enters, perhaps, a new state. On scanning the entire input string if the automaton enters an **accepting** or **final** state, then the input string is accepted by the automaton.

Formally, a deterministic finite automaton(DFA) $\mathcal{M}$ is a 5-tuple $(\Sigma, Q, q_0, \delta, F)$ where

1. Σ: a finite *alphabet*,
2. Q: a finite set of *states*,
3. $q_0 \in Q$: the *initia*l or *start* state,
4. $\delta : Q \times \Sigma \to Q$; the *transition function*,
5. $F \subseteq Q$: the set of *final* or *accepting* states.

Extension to Σ^*: The transition function δ is extended to Σ^* as follows.

$$\delta^* : Q \times \Sigma^* \longmapsto Q$$

$$\delta^*(q, \lambda) = q,$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a).$$

It is clear that the equation

$$\delta^*(q, w) = \hat{q}$$

means that "the automaton $\mathcal{M}$ in state q after reading the entire string w enters the state $\hat{q}$". Thus we have

$$\delta^*(p, w_1 w_2) = \delta^*(\delta^*(p, w_1), w_2).$$

The **language accepted** by $\mathcal{M}$ is

$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}.$$

A language $\mathcal{L} \subseteq \Sigma^*$ is said to be **regular** if for some DFA $\mathcal{M}, \mathcal{L} = \mathcal{L}(\mathcal{M})$ i.e. it is accepted by some DFA.

State Transition Diagram: The state transition diagram of an automaton $\mathcal{M}$ gives the entire information about $\mathcal{M}$ and is defined as follows.

It is a labelled directed graph whose nodes, represented by circles, are labelled with the states of $\mathcal{M}$. There is a directed edge labelled a from a node with label p to a node with label q if $\delta(p, a) = q$. The initial state is designated with an arrow while the accepting states are denoted with two circles.
Examples of DFA:

1. Construct a DFA that accepts all binary strings which are binary representation of non-negative integers that are congruent to 0 mod 5
   e.g. $\delta(q_2, 0) = q_4; \delta(q_2, 1) = q_0$.
2. Construct a DFA that accepts all binary strings containing an even numbers of 0's and 1's
3. Construct a DFA that accepts all binary strings containing 1101 as a substring.

1.1 Non-deterministic Finite Automaton(NFA)

To prove closure under concatenation and Kleene closure, we need the notion of non-determinism in which the automaton has the choice of entering one of several states. Thus in the definition of DFA we just need to change the state transition function δ to the following.

$$\delta : Q \times \Sigma \longmapsto \mathcal{P}(Q).$$

Thus the equation

$$\delta(p, a) = \{q_1, \ldots, q_k\}$$

means that "the automaton in state p reading the letter a has the choice of entering any one of the states $q_1, q_2, \ldots, q_k$.
Extension to Σ^*:

$$\delta^*(q, \lambda) = \phi,$$

$$\delta^*(q, wa) = \bigcup_{p \in \delta^*(q.w)} \delta(p, a).$$

Definition 1. *A string $w \in \Sigma^*$ is accepted by $\mathcal{M}$ if $\delta^*(q_0, w) \bigcap F \neq \phi$. The language accepted by $\mathcal{M}$ is*

$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* : \delta^*(q_0, w) \bigcap F \neq \phi\}.$$

Observe that a string $a_1 \ldots . a_n$ is accepted by $\mathcal{M}$ if there is a sequence of states $q_1, \ldots, q_n$ such that $q_1 \in \delta(q_0, a_1), q_2 \in \delta(q_1, a_2), \ldots, q_n \in \delta(q_{n-1}, a_n)$ and $q_n \in F$.
Also $\delta^*(q, w)$ denotes all the possible states reached by $\mathcal{M}$ starting from state q and reading the string w.
Equivalence:

Theorem 1. *There is an algorithm that converts a given NFA $\mathcal{M}$ into an equivalent DFA $\hat{\mathcal{M}}$. Consequently, $\mathcal{L}$ is regular iff it is accepted by an NFA.*

Proof idea: Given an NFA $\mathcal{M} = (\Sigma, Q, q_0, \delta, F)$ first observe that $\delta^*(q_0, w)$ gives the set of all possible states that can be reached by $\mathcal{M}$ from the initial state on reading w. This set of states will be the state of the equivalent automaton $\hat{\mathcal{M}}$. Also $\mathcal{M}$ accepts w if this set of states contains an accepting state of $\mathcal{M}$. Thus an accepting state of $\hat{\mathcal{M}}$ will be those sets of states that contain an accepting state of $\mathcal{M}$. Thus we construct $\hat{\mathcal{M}} = (\Sigma, \hat{Q}, \hat{q}_0, \hat{\delta}, \hat{F})$ as follows.

1. $\hat{Q} = \mathcal{P}(Q)$
2. $\hat{q}_0 = \{q_0\}$

3. For $P \subseteq Q$, and $a \in \Sigma$,

$$\hat{\delta}(P, a) = \bigcup_{p \in P} \delta(p, a)$$

4. $\hat{F} = \{P \subseteq Q : P \bigcap F \neq \phi\}$.

Claim:

$$\hat{\delta}^*(\hat{q}_0, w) = \delta^*(q_0, w).$$

Thus

$$w \in \mathcal{L}(\hat{\mathcal{M}}) \leftrightarrow \hat{\delta}^*(\hat{q}_0, w) \in \hat{F}$$

$$\leftrightarrow \delta^*(q_0, w) \bigcap F \neq \phi \leftrightarrow w \in \mathcal{L}(\mathcal{M}).$$

□

Example: Construct an NFA accepting all binary string containing 101 as a substring.

Closure Properties I: Closure under finite $\bigcup, \bigcap$ and complementation

Theorem 2. *The class of regular languages is closed under finite $\bigcup$, finite $\bigcap$ and complementation. Consequently, the regular languages form a Boolean algebra*

Proof idea: Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two DFAs accepting $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. We shall construct a DFA $\hat{\mathcal{M}}$ that accepts $\mathcal{L}_1 \bigcap \mathcal{L}_2$. On input a string w, $\hat{\mathcal{M}}$ runs both $\mathcal{M}_1$ and $\mathcal{M}_2$ on w simultaneously i.e. in parallel. $\hat{\mathcal{M}}$ accepts w iff both $\mathcal{M}_1$ and $\mathcal{M}_2$ enter accepting states.

Formally, let $\mathcal{M}_1 = (\Sigma, Q_1, q_0^1, \delta_1, F_1)$ and $\mathcal{M}_2 = (\Sigma, Q_2, q_0^2, \delta_2, F_2)$. Define $\hat{\mathcal{M}} = (\Sigma, \hat{Q}, \hat{q}_0, \hat{\delta}, \hat{F})$ as follows.

1. $\hat{Q} = Q_1 \times Q_2$
2. $\hat{q}_0 = (q_0^1, q_0^2)$
3. $\hat{\delta}((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$
4. $\hat{F} = F_1 \times F_2$.

Claim: $\hat{\delta}^*((p, q), w) = (\delta_1^*(p, w), \delta_2^*(q, w))$.
Hence

$$w \in \mathcal{L}(\hat{\mathcal{M}}) \leftrightarrow \hat{\delta}^*(\hat{q}_0, w) \in \hat{F}$$

$$\leftrightarrow \delta_1^*(q_0^1, w) \in F_1 \ \&\ \delta_2^*(q_0^2, w) \in F_2$$

$$\leftrightarrow w \in \mathcal{L}_1 \ \&\ w \in \mathcal{L}_2 \leftrightarrow w \in \mathcal{L}_1 \bigcap \mathcal{L}_2.$$

□

Closure Properties II:
Closure under concatenation:

Theorem 3. *If $\mathcal{L}_1$ and $\mathcal{L}_2$ are regular languages, then so is $\mathcal{L}_1.\mathcal{L}_2$.*

Proof idea: Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two DFAs accepting $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. The NFA $\mathcal{M}$ that accepts $\mathcal{L}_1.\mathcal{L}_2$ first runs $\mathcal{M}_1$ and on entering an accepting state has the choice of continuing in $\mathcal{M}_1$ or to enter the initial state of $\mathcal{M}_2$. This enables $\mathcal{M}$ to accept strings of the form $w_1.w_2$, where w_1 is accepted by $\mathcal{M}_1$ and w_2 is accepted by $\mathcal{M}_2$.

Formally, let $\mathcal{M}_1 = (\Sigma, Q_1, q_0^1, \delta_1, F_1)$ and $\mathcal{M}_2 = (\Sigma, Q_2, q_0^2, \delta_2, F_2)$. W.l.g. assume that $\lambda \notin \mathcal{L}_1$. Construct $\hat{\mathcal{M}} = (\Sigma, \hat{Q}, \hat{q}_0, \hat{\delta}, \hat{F})$ as follows

1. $\hat{Q} = Q_1 \bigcup Q_2$
2. $\hat{q}_0 = q_0^1$

3. $\hat{\delta}(q,a) = \begin{cases} \{\delta_1(q,a)\} & \text{if } \ q \in Q_1 - F_1 \\ \{\delta_1(q,a), \delta_2(q_0^2,a)\} & \text{if } \ q \in F_1 \\ \{\delta_2(q,a)\} & \text{if } \ q \in Q_2 \end{cases}$
4. $\hat{F} = F_2$.

Clearly $\mathcal{L}(\mathcal{M}) = \mathcal{L}_1.\mathcal{L}_2$.

If $\lambda \in \mathcal{L}_1$, then consider $\mathcal{L}_1' = \mathcal{L}_1 - \{\lambda\}$. Clearly, $\mathcal{L}_1'$ is regular and

$$\mathcal{L}_1.\mathcal{L}_2 = \mathcal{L}_1'.\mathcal{L}_2 \cup \mathcal{L}_2.$$

The first term of RHS is regular by above and $\mathcal{L}_2$ is given to be regular. So the union is regular and we are done. □

Closure under Kleene *:

Theorem 4. *Let $\mathcal{L}$ be a regular language. Then $\mathcal{L}^*$ is also regular.*

Proof idea: Let $\mathcal{M}$ be a DFA that accepts $\mathcal{L}$. W.l.g. assume that $\delta(q,a) \neq q_0$ for all $q \in Q$ and $a \in \Sigma$. (Such an automaton is called *non-restarting.*) We construct $\hat{\mathcal{M}}$ that runs $\mathcal{M}$ and on entering an accepting state has the option of either continuing as $\mathcal{M}$ or restart from the initial state of $\mathcal{M}$. Thus $\hat{\mathcal{M}} = (\Sigma, \hat{Q}, \hat{q}_0, \hat{\delta}, \hat{F})$ is defined as follows.

1. $\hat{Q} = Q$
2. $\hat{q}_0 = q_0$
3. $\hat{\delta}(q,a) = \begin{cases} \{\delta(q,a)\} & \text{if } \ \delta(q,a) \in Q - F \\ \{\delta(q,a), q_0\} & \text{if } \ \delta(q,a) \in F \end{cases}$.
4. $\hat{F} = \{q_0\}$.

The NFA $\hat{\mathcal{M}}$ accepts $\mathcal{L}^*$ □

Exercise 1. Given a DFA $\mathcal{M}$, construct a non-restarting DFA $\mathcal{M}'$ that is equivalent to $\mathcal{M}$.

1.2 Regular Expressions

An important notion in Automata Theory is the concept of **regular expressions** which we now introduce.

Definition 2. *A regular expression over an alphabet Σ is defined by induction as follows.*

1. *For each $a \in \Sigma$, $\boldsymbol{a}$ is a regular expression and represents the language $\{a\}$,*
2. *λ and ϕ are regular expressions and represent $\{\lambda\}$ and ϕ respectively,*
3. *If r_1 and r_2 are two regular expressions representing $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively, then so is $(r_1 + r_2)$ and it represents $\mathcal{L}_1 \bigcup \mathcal{L}_2$,*
4. *If r_1 and r_2 are two regular expressions representing $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively, then so is $(r_1.r_2)$ and it represents $\mathcal{L}_1.\mathcal{L}_2$,*
5. *If r is a regular expression representing $\mathcal{L}$ the so is (r^*) and it represents $\mathcal{L}^*$.*

Notation: *$\mathcal{L}(\mathbf{r})$ denotes the language represented by* $\mathbf{r}$

The following can be derived by induction on the *length* of regular expressions and from the closure properties.

Lemma 1. *The language represented by a regular expression is regular.*

Properties of regular expressions:
For two regular expressions $\mathbf{r}$ and $\mathbf{s}$ we write $\mathbf{r} = \mathbf{s}$ if $\mathcal{L}(\mathbf{r}) = \mathcal{L}(\mathbf{s})$.
The following identities hold for regular expressions.

1. $\mathbf{r} + \mathbf{r} = \mathbf{r}$.
2. $\mathbf{r} + \mathbf{s} = \mathbf{s} + \mathbf{r}$.
3. $(\mathbf{r} + \mathbf{s}) + \mathbf{t} = \mathbf{r} + (\mathbf{s} + \mathbf{t})$.
4. $(\mathbf{r}.\mathbf{s}).\mathbf{t} = \mathbf{r}.(\mathbf{s}.\mathbf{t})$.
5. $\mathbf{r}.(\mathbf{s} + \mathbf{t}) = \mathbf{r}.\mathbf{s} + \mathbf{r}.\mathbf{t}$.
6. $(\mathbf{r} + \mathbf{s}).\mathbf{t} = \mathbf{r}.\mathbf{t} + \mathbf{s}.\mathbf{t}$.
7. $(\mathbf{r}^*)^* = \mathbf{r}^*$.
8. $(\lambda + \mathbf{r})^* = \mathbf{r}^*$.
9. $(\mathbf{r} + \mathbf{s})^* = (\mathbf{r}^*.\mathbf{s}^*)^* = (\mathbf{r}^* + \mathbf{s}^*)^*$.

Exercise 2. .

1. Let **r, s** be two regular expressions. Consider the following equation in the regular expression **X**.

$$\mathbf{X} = \mathbf{s} + \mathbf{X}.\mathbf{r}.$$

   Prove that this equation has a solution

$$\mathbf{X}{=}(\mathbf{s}.\mathbf{r}^*).$$

   Show that the solution is unique if $\lambda \notin \mathcal{L}(\mathbf{r})$.
2. Let $\mathcal{L} = \{x \in \{a, b\}^* : x \neq \lambda$ and bb is not a substring of x$\}$.
   (a) Show that $\mathcal{L}$ is regular by constructing a DFA $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = \mathcal{L}$.
   (b) Find a regular expression $\mathbf{r}$ such that $\mathcal{L}(\mathbf{r}) = \mathcal{L}$.

Theorem 5 (Kleene). *A language$\mathcal{L}$ over Σ is regular iff there is a regular expression over Σ that represents $\mathcal{L}$.*

Proof idea: One direction follows from Lemma 1. For the other direction, suppose $\mathcal{L}$ is accepted by a DFA $\mathcal{M} = (\Sigma, Q, q_1, \delta, F)$. Suppose $Q = \{q_1, q_2, \ldots, q_n\}$. Define
$R^k_{i,j} = \{w \in \Sigma^* : \delta^*(q_i, w) = q_j$ & $\mathcal{M}$ does not pass through any intermediate state q_l with $l > k\}$.
We shall show by induction on k that each $R^k_{i,j}$ can be represented by a regular expression. Clearly,

$$R^0_{i,j} = \{a \in \Sigma : \delta(q_i, a) = q_j\}$$

and hence can be represented by a regular expression.
Claim:

$$R^{k+1}_{i,j} = R^k_{i,j} \bigcup R^k_{i,k+1}(R^k_{k+1,k+1})^*.R^k_{k+1,j} \qquad (1)$$

By induction hypothesis, each term in RHS of (1) can be represented by a regular expression. So the RHS can be represented by a regular expression..Hence the LHS term can also be represented by a regular expression. Consequently,

$$\mathcal{L} = \bigcup_{q_j \in F} R^n_{1,j}$$

can also be represented by a regular expression. □

Corollary 1. *A language $\mathcal{L}$ is regular iff it can be obtained from finite languages by finitely many applications of $\bigcup, \bigcap$ and Kleene *.*

1.3 The Pumping Lemma and Its Applications

Pumping Lemma: Let $\mathcal{L}$ be a regular language accepted by a DFA $\mathcal{M}$ with n states. Then for every $x \in \mathcal{L}$ with $|x| \geq n$, x can be written as $x = uvw$ where

1. $|v| > 0$,
2. $|uv| \leq n$,
3. for all $i = 0, 1, 2, \dots$ $uv^iw \in \mathcal{L}$.

Proof idea: Let $\mathcal{M} = (\Sigma, Q, q_0, \delta, F)$ with $|Q| = n$. Let $x = a_1 \dots a_k$, where $k \geq n$. Define for $1 \leq i \leq k, \delta^*(q_0, a_1 \dots a_i) = q_i$. Clearly, in the sequence $q_0, q_1, \dots, q_n$ there exist $i < j \leq n$ such that $q_i = q_j$. Set $u = a_1 \dots a_i, v = a_{i+1} \dots a_j, w = a_{j+1} \dots a_k$. It is easy to see that conditions (1)-(3) are satisfied. □

Use the Pumping Lemma to show that the following languages are not regular.

1. $\{0^n1^n : n \geq 1\}$.
2. $\{0^p : p \text{ is prime}\}$.
3. $\{0^{n^2} : n \geq 1\}$.
4. $\{0^{n^3} : n \geq 1\}$.
5. $\{0^n1^m : 0 < n \leq m\}$.
6. Binary strings with equal numbers of 0's and 1's.
7. $\{ww : w \in \{0,1\}^*\}$.
8. Set of all palindromes.
9. $\{0^n1^n2^n | n \geq 0\}$.
10. $\{w.w.w | w \in \{a,b\}^*$.
11. $\{0^{2^n} | n \geq 0\}$.

Proof idea: (2) Fix a DFA with n states accepting (2). Fix a prime $p \geq n+2$. By Pumping Lemma

1. $0^p = uvw$
2. $|v| > 0$
3. $|uv| \leq n$
4. $uv^iw \in \mathcal{L}$ for all i.

Let $|v| = m$. Then $|uv^iw| = p + (i-1)m$. Choose $i = p+1$. Then $|uv^{p+1}w| = p(m+1)$ which is not prime. Hence $uv^{p+1}w \notin \mathcal{L}$ contradicting (4). Hence $\mathcal{L}$ cannot be regular. □

1.4 Decision Properties

I.**Emptiness:**

Theorem 6. *Let $\mathcal{M}$ be a DFA with n states that accepts $\mathcal{L}$. Then $\mathcal{L}(\mathcal{M}) \neq \phi$ iff there is an $x \in \mathcal{L}(\mathcal{M})$ such that $|x| < n$.*
Consequently, there is an algorithm to test whether $\mathcal{L}(\mathcal{M})$ is empty or not.

Proof idea: If $\mathcal{L}(\mathcal{M}) \neq \phi$ then fix $x \in \mathcal{L}(\mathcal{M})$ with *smallest length.*
Claim: $|x| < n$.
Algo: Enumerate all strings of length less than n. If none is accepted by $\mathcal{M}$, then empty, else non-empty. □

II. **Finiteness:**

Theorem 7. *$\mathcal{L}(\mathcal{M})$ is infinite iff there is a string $x \in \mathcal{L}(\mathcal{M})$ such that $n \leq |x| < 2n$.*
Consequently, there is an algorithm to test whether $\mathcal{L}(\mathcal{M})$ is finite or infinite.

Proof idea: Let $\mathcal{M}$ be a DFA with n states accepting $\mathcal{L}$.
$\leftarrow$: If the condition holds, then $\mathcal{L}$ is infinite by the Pumping Lemma
$\rightarrow$: So let $\mathcal{L}$ be infinite. Fix a string $x \in \mathcal{L}$ such that $|x| \geq n$ and $|x|$ is as small as possible.
Claim: $n \leq |x| < 2n$.
By the Pumping Lemma, x can be written as $x = uvw$, where

1. $|v| > 0$
2. $|uv| < n$ and
3. $uv^iw \in \mathcal{L}$ for $i = 0, 1, 2 \ldots$

In particular, $uw \in \mathcal{L}$. Also $|uw| < |x|$. Hence by our choice of x, $|uw| < n$. Now

$$|x| = |uw| + |v| \leq |uw| + |uv| < n + n = 2n.$$

This complete the proof of the first part.
More decision problems:
Test whether

1. $\mathcal{L}(\mathcal{M}) = \Sigma^*$.
2. $\mathcal{L}(\mathcal{M}_1) \subseteq \mathcal{L}(\mathcal{M}_2)$.
   Hint: Construct a DFA $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}_1) \bigcap \mathcal{L}(\mathcal{M}_2)^C$.
3. $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$.

Exercise 3. .

1* Given an NFA $\mathcal{M}$ construct a regular expression that represents $\mathcal{L}(\mathcal{M})$.
(NB: You may take a look at the following notes
https://www.cs.unc.edu/ plaisted/comp455/slides/fare2.3.pdf)
2. **Myhill-Nerode Theorem:** Given a language $\mathcal{L} \subseteq \Sigma^*$, and strings $x, y \in \Sigma^*$, define $x \equiv_{\mathcal{L}} y$ if

$$\forall w \in \Sigma^*(xw \in \mathcal{L} \leftrightarrow yw \in \mathcal{L}).$$

(i) Show that $\equiv_{\mathcal{L}}$ is an equivalence relation.
(ii) Show that if $x \equiv_{\mathcal{L}} y$ then for any $w \in \Sigma^*$, $xw \in \mathcal{L} \leftrightarrow yw \in \mathcal{L}$.
(iii) Define the *index* of $\mathcal{L}$ to be the maximum number of inequivalent elements. Show that $\mathcal{L}$ is regular iff $\mathcal{L}$ is of finite index. Moreover, its index is the size of the smallest DFA accepting it. (Note: The index is the number of equivalence classes.)