# Lecture 1: Finite State Automaton

Instructor: Ketan Mulmuley
Scriber: Yuan Li

January 6, 2015

## 1 Deterministic Finite Automaton

Informally, a *deterministic finite automaton* (DFA) has finite number of states, and input symbols come from a finite alphabet $\Sigma$. Among all states, there is one designated initial state and one or more final states. Given the current state and the input symbol, there is exactly one transition (from old state to new state). When it is stuck or does not reach a final state, the input is rejected.

For example, following diagram is a DFA which accepts 01 strings with even number of 0's and even number of 1's.

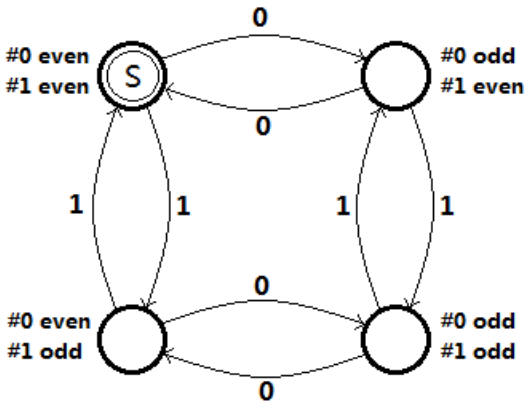


Figure 1: DFA accepting even number of 0s and 1s

**Definition 1.1.** *A deterministic finite automaton (DFA) is a 5-tuple* $M = (Q, \Sigma, \delta, q_0, F \subseteq Q)$*, where* $Q$ *is the set of states,* $\Sigma$ *is a (finite) set of alphabets,* $\delta : Q \times \Sigma \to Q$ *is the transition function,* $q_0$ *is the initial state, and* $F$ *is the set of final states.*

In order to *formally* define the language accepted by a DFA, let us recursively define the *transition closure* of $\delta$, denoted by $\hat{\delta} : Q \times \Sigma^* \to Q$, as follows. Let $\epsilon$ denote the empty string.

- $\hat{\delta}(q, \epsilon) = q$.
- $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$, where $w \in \Sigma^*$ and $a \in \Sigma$.

A string $w \in \Sigma^*$ is *accepted* iff $\hat{\delta}(q_0, w) \in F$. The *language accepted by* $M$ is

$$L(M) := \{w : \hat{\delta}(q_0, w) \in F\}.$$

A language $L \subset \Sigma^*$ is called *regular* iff $L = L(M)$ for some DFA $M$.

**Exercise 1.2** (*)**.** *Prove the set of all prime numbers (encoded as a binary number) is not regular.*

# 2 Nondeterministic Finite Automaton

Informally, *Nondeterministic finite automaton* (NFA) is a DFA with nose, that is, the transition, given current state and the next input symbol, is not unique, and the NFA can choose the right way to go. For example, following is a NFA that accepts Boolean strings ending with three consecutive 0's or two consecutive 1's.

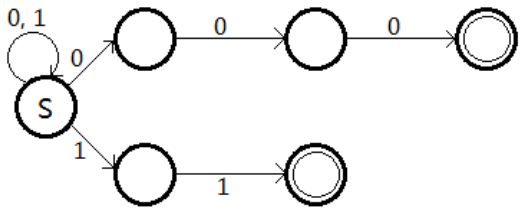


Figure 2: NFA accepting strings ending with 000 or 11

**Definition 2.1.** *A* Nondeterministic finite automaton *(NFA) is a 5-tuple* $M = (Q, \Sigma, \delta, q_0, F \subseteq Q)$*, where* $Q$ *is the set of states,* $\Sigma$ *is a (finite) set of alphabets,* $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ *is the transition function,* $q_0$ *is the initial state, and* $F$ *is the set of final states. Here,* $\mathcal{P}(Q)$ *denotes the power set of* $Q$*.*

**Remark 2.2.** *The difference between DFA and NFA is that, for DFA, the transition function* $\delta$ *is a function from* $Q \times \Sigma$ *to* $Q$*; for NFA, the transition function* $\delta$ *is a function from* $Q \times \Sigma$ *to* $\mathcal{P}(Q)$ *(the power set of* $Q$*). For example,* $\delta(q, a) = A \subseteq Q$ *means that, at state* $q$*, if the input symbol is* $a$*, the next state could be any state in* $A$*.*

Similarly, define transitive closure function $\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$ as follows.

- $\hat{\delta}(q, \epsilon) = \{q\}$.
- $\hat{\delta}(q, wa) = \{p \in Q : \exists r \in \hat{\delta}(q, w) \text{ such that } p \in \delta(r, q)\}$.

The *language accepted by* $M$ is defined as

$$L(M) := \{w \in \Sigma^* : \hat{\delta}(q, w) \text{ contains a state in } F\}.$$

A DFA is clearly a NFA by definition. Here comes a question: can every NFA be simulated by some DFA?

**Theorem 2.3.** *Every NFA can be simulated by some DFA. In other words, DFA and NFA are equivalent.*

*Proof.* Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Define DFA

$$\widetilde{M} = (\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{q}_0, \widetilde{F}),$$

where $\widetilde{Q} = \mathcal{P}(Q)$, $\widetilde{q}_0 = \{q_0\}$, $\widetilde{F} = \{A \subseteq Q : A \cap F \neq \emptyset\}$, and

$$\widetilde{\delta}(\{p_1, p_2, \ldots, p_i\}, a) := \{p' : \exists p_i \text{ such that } p' \in \delta(p_i, a)\}.$$

It is not difficult to verify that $L(\widetilde{M}) = L(M)$. □

Note that in the proof, the number of states in the DFA is exponential in the number of states of NFA. For example, if the number of states of NFA is 200, the corresponding DFA would have $2^{200}$ states, which is more than the number of particles in the universe!

**Exercise 2.4** (*)**.** *Prove that there exists NFA with* $n$ *states which can* not *be simulated by any DFA with* $\leq n^{100}$ *states.*

# 3 NFA with $\epsilon$-moves

Recall that $\epsilon$ denotes the empty string – string of length 0. Again, NFA with $\epsilon$-moves is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q).$$

If $\delta(q, \epsilon) = A \subseteq Q$, then the next state of $q$ could be any state in $A$ on the empty string (= without any input symbol).

For example, we have constructed a DFA, and hence also an NFA, which accepts strings that have even number of 0's and even number of 1's, denoted by $M_1$; and we have constructed an NFA which accepts strings with either 3 consecutive 0's or 2 consecutive 1's, denoted by $M_2$. The following NFA with $\epsilon$-moves accepts strings which either have even number of 0's and even number of 1's, or contains 3 consecutive 0's or 2 consecutive 1's.

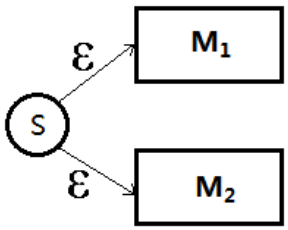


Figure 3: $\epsilon$-NFA accepting $L(M_1) \cup L(M_2)$

Define $\epsilon$-closure of $q$ as the set of states reachable from $q$ by $\epsilon$-moves, denoted by $\epsilon$-closure($q$). Let

$$\epsilon\text{-closure}(P) = \bigcup_{q \in P} \epsilon\text{-closure}(q).$$

- $\hat{\delta}(q, \epsilon) = \epsilon$-closure of $q$.
- $\hat{\delta}(q, wa) = \epsilon$-closure($P$), where

$$P = \{p \in Q : \exists r \in \widetilde{\delta}(q, w) \text{ such that } p \in \delta(r, a)\}.$$

The *language accepted by* $M$ is

$$L(M) := \{w \in \Sigma^* : \hat{\delta}(q, w) \text{ contains a state in } F\}.$$

The next result says NFA with $\epsilon$-moves is *not* stronger than the usual NFA.

**Theorem 3.1.** *Any NFA with $\epsilon$-moves can be simulated by some NFA (without $\epsilon$-moves).*

*Proof.* Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with $\epsilon$-moves. Define an NFA without $\epsilon$-moves $M' = (Q, \Sigma, \delta', q_0, F')$, where

$$F' = \begin{cases} F \cup \{q_0\}, & \text{if } \epsilon\text{-closure}(q_0) \text{ contains a state in } F, \\ F, & \text{otherwise} \end{cases} \tag{1}$$

and $\delta'(q, a) = \hat{\delta}(q, a)$. We leave it as an exercise to verify $L(M) = L(M')$. □

Putting everything together, we have shown DFA is equivalent to NFA, equivalent to NFA with $\epsilon$-moves (in the sense of computational power, regardless of the number of states).