Lecture #3: Nondeterministic Finite Automata

Key Concepts

Definition 1. A ***nondeterministic finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite, nonempty, set of ***states***;
- Σ is an ***alphabet*** such that $Q \cap \Sigma = \emptyset$;
- $\delta : Q \times \Sigma_\lambda \to \mathcal{P}(Q)$ is a ***transition function***;
- $q_0 \in Q$ is the ***start state***; and
- $F \subseteq Q$ is the set of ***accept states***.

Here, $\Sigma_\lambda = \Sigma \cup \{\lambda\}$ and δ is a ***total*** function from $Q \times \Sigma_\lambda$ to $\mathcal{P}(Q)$.

Definition 2. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic finite automaton. The, for $q \in Q$, $\boldsymbol{Cl_\lambda(q)}$ is the ***set of states*** that reachable from q by following ***zero or more*** λ-transitions.

$Cl_\lambda(q)$ is sometimes called the ***λ-closure*** of the state q.

Definition 3. Let $M = (Q, \Sigma, \delta, q_0, F)$. The ***extended transition function*** of M is the total function

$$\delta^\star : Q \times \Sigma^\star \to \mathcal{P}(Q)$$

such that, for $q \in Q$ and $\omega \in \Sigma^\star$,

$$\delta^\star(q, \omega) = \begin{cases} Cl_\lambda(q) & \text{if } \omega = \lambda, \\ \displaystyle\bigcup_{r \in \delta^\star(q,\mu)} \left(\bigcup_{s \in \delta(r,\sigma)} Cl_\lambda(s) \right) & \text{if } \omega = \mu \cdot \sigma \text{ for } \mu \in \Sigma^\star \text{ and } \sigma \in \Sigma. \end{cases}$$

Definition 4. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic finite automaton. Then, for every string $\omega \in \Sigma^\star$, M ***accepts*** ω if

$$\delta^\star(q_0, \omega) \cap F \neq \emptyset$$

and M ***rejects*** ω otherwise.

Definition 5. Let $M = (Q, \Sigma, \delta, q_0, F)$. Then the ***language*** of M, $\boldsymbol{L(M)}$, is the set of strings $\omega \in \Sigma^\star$ such that M accepts ω.

The lecture also presented a proof of the following.

Claim 6. *For every alphabet Σ and for every language $L \subseteq \Sigma^\star$, the following are equivalent:*

*(a) L is a **regular language**. That is, L is the language of a **deterministic finite automaton**.*

*(b) L is the language of a **nondeterministic finite automaton**.*

Sketch of Proof. To prove that $(a) \Rightarrow (b)$, one should consider an arbitrarily chosen alphabet Σ and language $L \subseteq \Sigma^\star$. If L is a regular language then there exists a ***deterministic finite automaton***

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

such that $L = L(M)$. One can define a ***nondeterministic finite automaton***

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

—- with the same set Q of states, start state q_0, set of accepting states F and alphabet Σ — such that $L(\widehat{M}) = L(M) = L$, by defining the transition function $\widehat{\delta} : Q \times \Sigma_\lambda \to \mathcal{P}(Q)$ as follows: For every state $q \in Q$ and for all $\sigma \in \Sigma_\lambda$,

$$\widehat{\delta}(q, \sigma) = \begin{cases} \{\delta(q, \sigma)\} & \text{if } \sigma \in \Sigma, \\ \emptyset & \text{if } \sigma = \lambda. \end{cases}$$

Then the state diagrams for M and $\widehat{M}$.

To prove that $(b) \Rightarrow (a)$, one should consider an arbitrarily chosen alphabet Σ and language $L \subseteq \Sigma^\star$, once again — and suppose that L is the language of a ***nondeterministic finite automaton***

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

One can proceed by designing a ***deterministic finite automaton***

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

— with the same alphabet Σ but generally, with a different set $\widehat{Q}$ of states, accept state and set of final states — using the design process from earlier lectures: When processing symbols in a string, $\widehat{M}$ should remember *the set of states in Q that can be reached when the same symbols have been processed by M* — so that states in $\widehat{M}$ correspond to *sets of states* in M.

Details of the construction of $\widehat{M}$ from M, and a proof of the correctness of this construction, are given in a supplement for this lecture. □

A ***simulation*** is something that can be presented to relate the power of two models of computation. In order to show that the machines described by a ***second*** model of computation are (in some sense) at least as powerful or efficient as the machines described by a ***first*** model of computation, we generally do the following:

(a) Consider an arbitrary machine M, of the type described by the ***first*** model of computation.

(b) Use M to define another machine $\widehat{M}$, of the type described by the ***second*** model of computation.

(c) Prove that $\widehat{M}$ solves the same problem as M.

The above claim was (arguably) proved by giving two simulations. Simulations will be used again, later on in this course.