CSE 105: Introduction to the Theory of Computation	Winter 2015
Lecture Notes: Nondeterministic Finite Automata	
Instructor: *Daniele Micciancio*	UCSD CSE

1 NFAs

A Nondeterministic Finite Automaton (NFA) is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$ consisting of

1. A finite set of states Q
2. Finite set of input symbols Σ
3. A transition function $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$
4. A start state $s \in Q$
5. A set of accepting states $F \subseteq Q$

Notice that the only difference between a DFA and an NFA is in the transition function δ. This is exactly the same as the definition of NFA given in the textbook. We proceed to define its computations using the same style as for DFAs.

1.1 Modeling computation as a transition system

The set of configurations $C = Q \times \Sigma^*$, initial configurations $I(w) = (s, w)$, final configurations $H = Q \times \epsilon$, and output function $O(q, \epsilon) =$ if $(q \in F)$ then 1 else 0" are the same as for DFAs. The transition relation is a simple extension of the definition for DFAs to take into account the new type of the transition function. Formally, the transition relation contains all transitions of the form

- $(q, xu) \to_R (q', u)$ where $q \in Q$, $x \in \Sigma \cup \{\epsilon\}$, $u \in \Sigma^*$ and $q' \in \delta(q, x)$.

Notice that when $x = \epsilon$, we have $xu = u$ and the transition does not modify the second component of the configuration.

The transition system associated to an NFA is nondeterministic: for each (final or nonfinal) configuration $c \in C$, there may be 0, 1 or more "next" configurations c'such that $c \to c'$. So, for each input string, there are in general several possible computations, and several corresponding output values. Also, some computations can be infinite, e.g., if there is a loop consisting of ϵ-transitions. Also, a computation does not necessarily stops upon reaching a final configuration.

By convention, the set of strings accepted by an NFA is defined as the set of strings w such that there is a computation $c_1 = I(w) \to c_1 \to c_2 \to \cdots \to c_n$ that outputs $O(c_n) = 1$. As discussed in the textbook, this is not a realistic model of computation, but it can be useful in studying the power and properties of (more realistic) deterministic finite automata.

1.2 Modeling computation as a function

As for the DFA, computations can also be simply defined by a function mapping inputs to outputs. As before, we start by extending the transition function δ of an NFA, to a function of type $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$ which takes a string as its second argument. The function δ^* is defined by induction on the length of the second argument according to the rules

1. $\delta^*(q,\epsilon)$ is the set of states $q' \in Q$ that can be reached from q following a sequence of 0 or more ϵ-transitions, i.e., there is a sequence of $n \geq 0$ states $q_1, \ldots, q_n$ where $q = q_1$, $q' = q_n$ and $q_{i+1} \in \delta(q_i, \epsilon)$ for all i.

2. $\delta^*(q, au) = \bigcup_{p \in \delta^*(q,\epsilon)} \bigcup_{r \in \delta(p,a)} \delta^*(r, u)$, i.e., all states that can be reached from q by first following a sequence of ϵ-transitions, then reading a from the input, and finally reading the rest of the input u.

Given an input string w, the function $\delta^*(s, w)$ returns not just one state, but the set of all possible states that can be reached starting from q and reading w from the input. The NFA accepts w if $\delta^*(s, w) \cap F \neq \emptyset$, i.e., if it is possible to reach a state in F, starting from s and reading w from the input. Formally, the function computed by an NFA M is defined as $f_M(w) = \text{if } (\delta^*(s, w) \cap F) = \emptyset) \text{ then } 0 \text{ else } 1$.