Nondeterministic Finite Automata

S. E. Venegas-Andraca
Quantum Information Processing Group
Tecnológico de Monterrey
http://www.mindsofmexico.org/sva
svenegas@itesm.mx

August 23, 2017

The contents of this handout are mostly based on [1].

1 Nondeterministic Finite Automata

When every step of a computation follows in a unique way from the preceding step (as in a DFA), we say that we are doing deterministic computation. In a nondeterministic machine, several choices may exist for the next state at any point (Fig. (1).)

Let us explain how Nondeterministic Finite Automata (NFA) work by means of an example. Please see the NFA presented in (Fig. (2)). Two evident differences:
1. Every state of a DFA always has exactly one exiting transition arrow for each symbol in the alphabet. Nondeterministic finite automata violate this rule. For example, q_1 in Fig. (2) has one exiting arrow for 0 but *two* for 1. Moreover, q_3 has one exiting arrow for 1 but none for 0. **So, in general, an NFA state may have zero, one or many exiting arrows for each alphabet symbol.**
2. In a DFA, labels on the transition arrows are symbols from the alphabet. In contrast, NFAs may have arrows labels with members of the alphabet <u>or</u> ε. Zero, one or many arrows may exit from each state with the label ε.

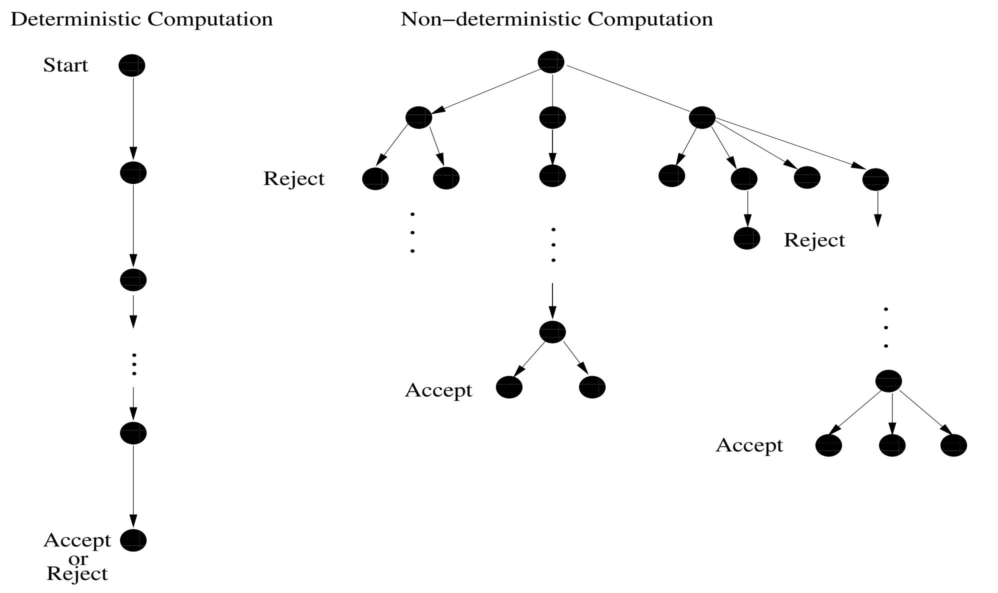

Figure 1: In a deterministic computing machine, every single step is fully determined by the previous step. In a nondeterministic computing machine, a step may be followed by n new steps or, equivalently, a nondeterministic computing machine makes n copies of itself, one for each possibility.

How does an NFA compute?
- Suppose that we are running an NFA on an input string and come to a state with multiple states to proceed (for example, say that we are in state q_1 in the NFA presented in Fig. (2) and that the next input symbol is 1). Then, after reading the corresponding input symbol, the machine splits into multiple copies of itself and follows all the possibilities in parallel. Each copy of the machine takes one of the possible ways to proceed and continues as before.
- If there are subsequent choices, the machine splits again.
- If the next input symbol does not appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of the computation associated with it.
- If a state with an ε symbol on an exiting arrow is encountered, then, *without reading any input*, the machine splits into multiple copies, each one following each of the exiting ε-labeled arrows and one staying at the current state. then, the machine proceeds nondeterministically as before.

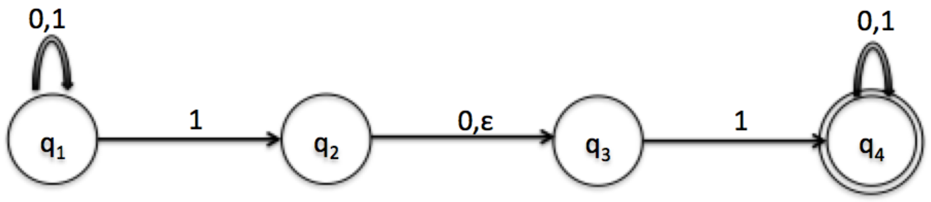

Figure 2: An example of a Nondeterministic Finite Automata.

- Finally, if any one of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input string.

Another way to think of a nondeterministic computation is as a tree of possibilities. The root of the tree corresponds to the start of the computation. Every branching point in the tree corresponds to a point in the computation at which the machine has multiple choices. The machine accepts if at least one of the computation branches ends in an accept state. A graphical illustration of a nondeterministic computation is given in Fig. (1).

Example 1.1. *Process 010110 on the NFA presented in Fig. (2).* **The answer can be found on Fig. (3)**.

An NFA is not a fully realistic model of computation as it assumes the capability of producing, in principle, an unlimited number of NFA instances to run in parallel (it would be like suddenly producing as many computers as instances for each computation step). However, nondeterminism may be viewed as a kind of parallel computation wherein multiple independent processes can be running concurrently, and this view does prepare the grounds for introducing the concept of probabilistic computation.

Thus, nondeterminism may be viewed as a kind of parallel computation wherein multiple independent processes can be running concurrently. When an NFA splits to follow several choices, that corresponds to a process forking into several children, each proceeding separately. If at least one of these processes accepts, then the entire computation accepts.

Definition 1.1. Nondeterministic Finite automaton. An NFA R_N is a 5-tuple $(Q, \Sigma, \delta, q_o, F)$, where 1) Q is a finite set of states; 2) Σ is a finite alphabet; 3) $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function; ($\mathcal{P}$ is the power set of set Q); 4) $q_0 \in Q$ is the start state, and 5) $F \subseteq Q$ is the set of accept states.

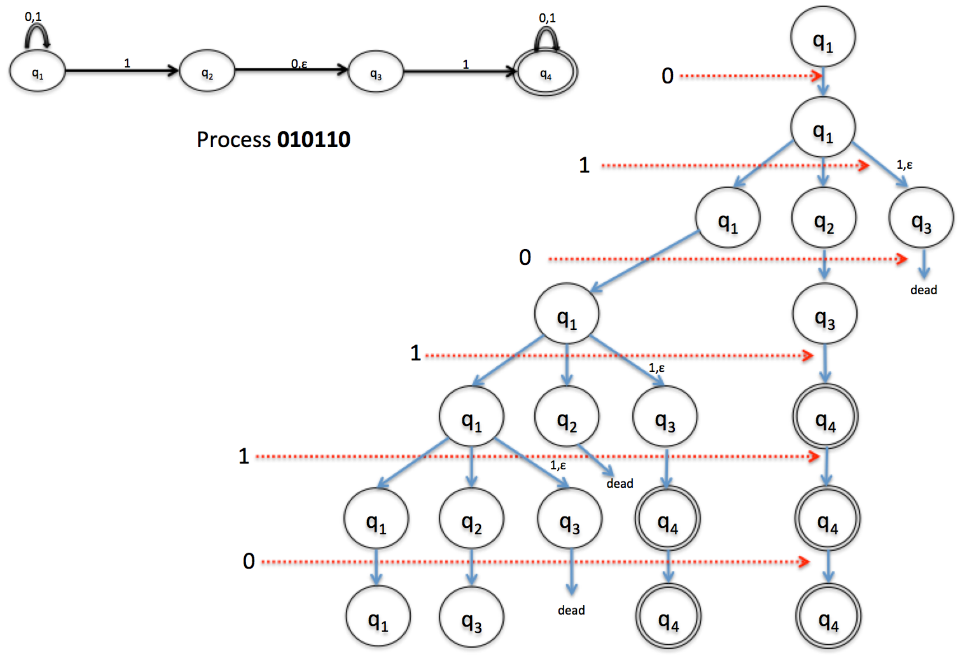

Figure 3: An example of a Nondeterministic Finite Automata.

Definition 1.2. Computation on a Nondeterministic Finite Automaton. Let $R_N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ. Then we say that R_N **accepts** w if we can write w as $w = y_1 y_2 \ldots y_m$, where each y_i is a member of Σ and a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions:
1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \ldots, m-1$, and
3. $r_m \in F$.

Condition 1 states that the machine starts out in the start state. Condition 2 means that state r_{i+1} is one of the allowable next states when R_N is in state r_i and reading y_{i+1}. Observe that $\delta(r_i, y_{i+1})$ is the *set* of allowable next states and so we say that r_{i+1} is a member of that set. Finally, condition 3 establishes that the machine accepts its input if the last state is an accept state. We say that R_N **accepts** language A if $A = \{w | R_N \text{ accepts } w\}$, i.e. A is the set of all strings accepted by R_N.

Example 1.2. Example of an NFA. *Let A be the language consisting of all strings over {0,1} containing a 1 in the third position from the end (e.g.* $000100 \in A$ *but* $0011 \notin A$*. Draw both a DFA and an NFA that recognize language A.*

It can be proved that DFAs and NFAs recognize the same class of languages. However, *the number of states of a deterministic counterpart of an NFA can be exponential in the number of branches of such an NFA* and this is a very important difference between these two models of computation.

References

[1] M. Sipser, Introduction to the Theory of Computation, PWS (2005).