COM364 Automata Theory
Lecture Note*2 - Nondeterminism

Kurtuluş Küllü

March 2018

The FA we saw until now were deterministic FA (DFA) in the sense that for each state and input symbol there was exactly one possible transition. In a nondeterministic FA (NFA), this condition is relaxed. So, it is possible to not to have a transition from a state for an input symbol or there can be several choices for the same symbol. Nondeterminism is a more general case. So, every DFA can be considered as a NFA.

Example: Figure 1 shows our first example NFA. You should note three important differences from those until now. First, there are two transitions from q_1 for the input symbol 1. Second, for some state and input symbols, there is no transition. For example, there is no transition for symbol 1 and state q_2 or for symbol 0 and q_3. Lastly, there is a transition from q_2 to q_3 for symbol ε. The ε symbol here is used for the empty string as before and this transition means that the automaton can go from q_2 to q_3 without any input symbol.

When a NFA is processing a string, we consider all possible computation paths. If any of these possible paths leads to an accepting state at the end of the input, the NFA accepts that input.

This way of processing can be visualized by using a tree notation. 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. As an example look at Figure 2. The example shows the NFA N_1 above processing string 010110.

Example: Let A be the language consisting of all strings over $\{0, 1\}$ containing a 1 in the third position from the end. (For example, 000100 is in A but 0011 is not.) An NFA to recognize A is shown in Figure 3.

Note: Every NFA can be converted into an equivalent DFA. Most of the time, the DFA will have any more states.

Question: What changes if we add ε-transitions from q_2 to q_3 and q_3 to q_4 in N_2?

Formal Definition of a NFA

The formal definition of a NFA is similar to that of a DFA except some important differences in details.

A NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. (same as in DFA) Q is a finite set of states,

*Based on the book "Introduction to the Theory of Computation" by Michael Sipser.

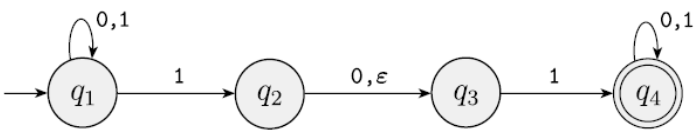

Figure 1: Our first example NFA, N_1.

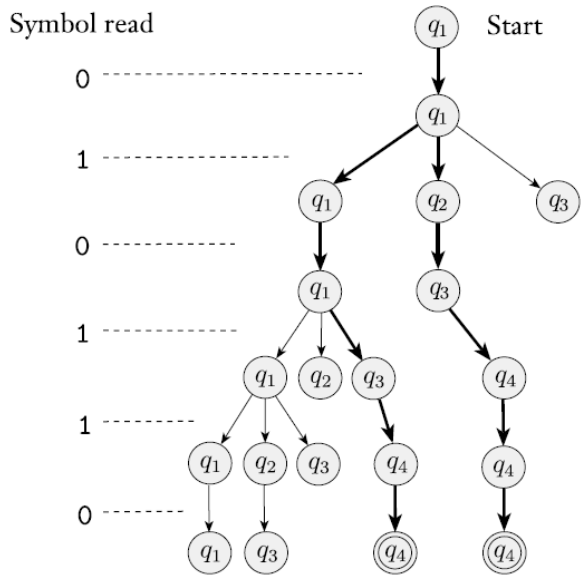

Figure 2: NFA N_1 processing string 010110 visualized as a tree.

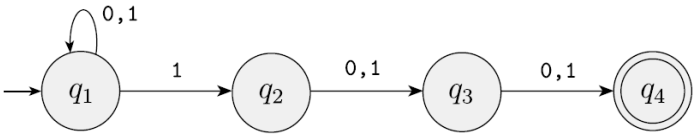

Figure 3: NFA N_2.

2. (same as in DFA) Σ is the input alphabet,
3. (different) $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function where Σ_ε stands for $\Sigma \cup \{\varepsilon\}$ and $P(Q)$ is the power set of Q,
4. (same as in DFA) $q_0 \in Q$ is the start state, and
5. (same as in DFA) $F \subseteq Q$ is the set of accepting (final) states.

Equivalance of NFA and DFA

Although NFA appear to be more powerful, the two types of automata are actually of equal power. In other words, the class of languages that can be recognized by NFA are the same as that of DFA and these are the regular languages.

Theorem: Every NFA has an equivalent DFA.

Proof Idea: We need to show that if a language is recognized by a NFA, there is also a DFA that recognizes the same language. We will convert the NFA to a DFA that simulates it.

If the NFA has k states, the power set has 2^k elements. These will be the states for the DFA. Transitions, start and accepting states can all be arranged logically according to the NFA. For a full proof, let $N = (Q, \Sigma, \delta, q_0, F)$ be the initial NFA and try to define the components of a DFA $D = (Q', \Sigma', \delta', q_0', F')$ that will recognize the same language.

It is easy to first assume that there are no ε-transitions, and then try to extend including them. When extending, a new definition, called *ε-closure*, becomes useful. The ε-closure for a state q, written as ε-closure(q) is defined as the set of states that can be reached from q following 0 or more ε-transitions. ε-closure of a state always includes at least itself. For example, for the machine N_1 in Figure 1, ε-closure(q_2) = $\{q_2, q_3\}$.

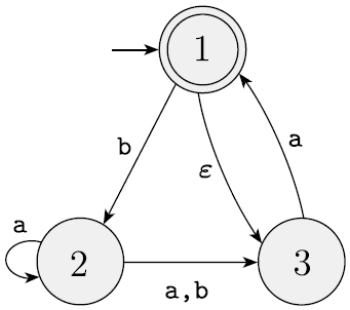

Figure 4: NFA N_4.

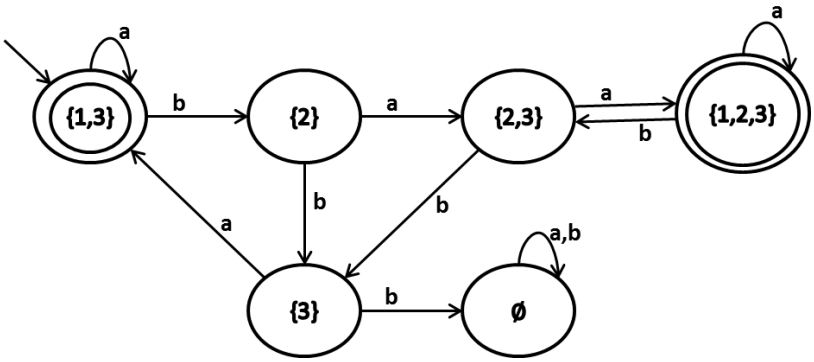

Figure 5: DFA D_4, which is equaivalent to the NFA N_4 in Figure 4.

Example: Figure 4 shows an example NFA called N_4 for which we will find the equivalent DFA, D_4. N_4 has three states, $\{1, 2, 3\}$, so we construct the DFA D_4 with eight states, one for each subset of N_4's states. Thus D_4's state set is $\{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$.

The start state of N_4 is 1, so the start state for D_4 should be ε-closure$(1) = \{1, 3\}$. The accepting states of D_4 should be the states that include an accepting state from N_4. The only accepting state of N_4 is 1, so the accepting states of D_4 are $\{\{1\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$. Lastly, the transitions can be found from the transitions of N_4 with careful analysis. We show the resulting D_4 in Figure 5. You should note that in the figure, there are six states although the power set of N_4's states has eight elements. In particular, states corresponding to $\{1\}$ and $\{1, 2\}$ are not shown. This is because there is no way to reach these states when we start from the start state. Hence, we omitted these states in the figure.

(Back to) Closure Properties

Before we started discussing nondeterminism, we proved that the regular languages are closed under union operation (by constructing a machine for $A \cup B$ from the machines for A and B). This proof would be much easier with nondeterminism as follows. If we have machines M_A and M_B that recognize languages A and B respectively, we can simply combine these two to create a new machine M by adding only a new start state and adding ε-transitions from this new state to the individual start states of M_A and M_B. This is shown graphically in Figure 6.

Closure Under Concatenation: If A and B are regular languages, prove that AB is also regular.

Because A and B are regular languages, there must be machines N_1 and N_2 that recognize them. We can construct a new machine N from N_1 and N_2 as shown in Figure 7 to recognize AB. This construction involves taking only the accepting states of N_1, making them nonaccepting, and adding ε-transitions from these states to the start state of N_2. Start state of N is the start state of N_1.

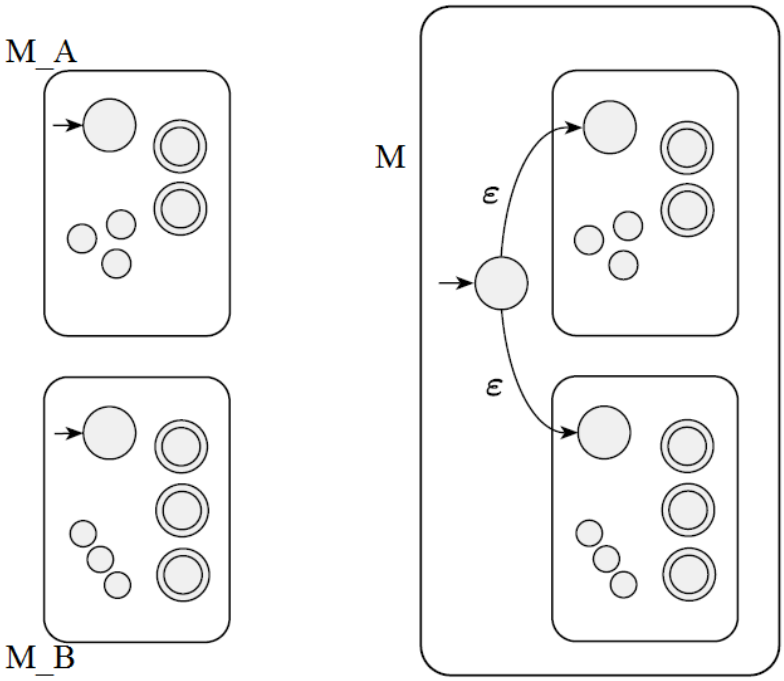

Figure 6: M_A and M_B can be combined in this way to create M, which will recognize $A \cup B$.

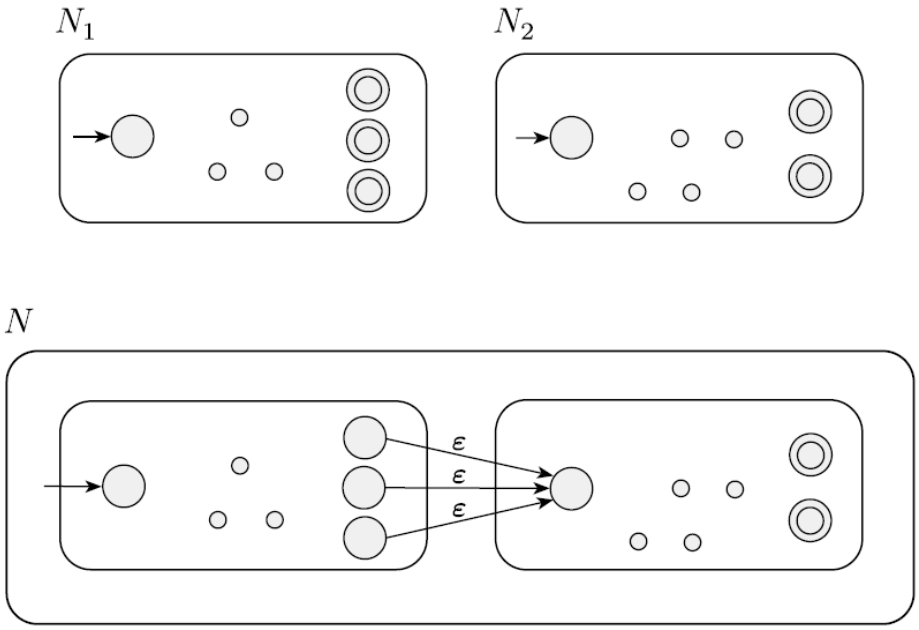

Figure 7: N_1 and N_2 can be combined in this way to create N, which will recognize AB.

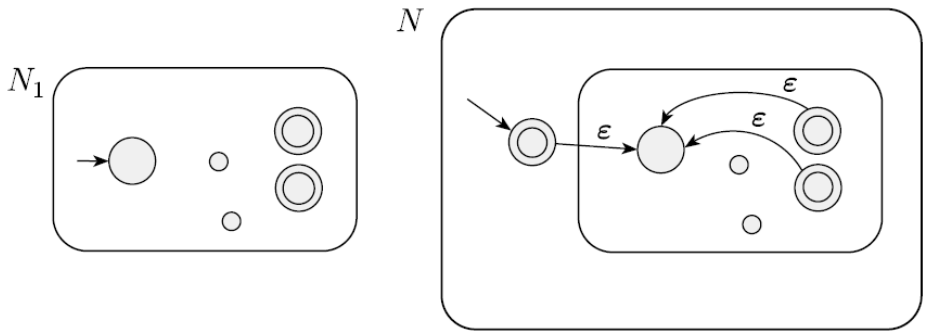

Figure 8: N_1 can be extended in this way to create N, which will recognize A^*.

Theorem: Regular languages are closed under the star operation.

Proof Idea: Let N_1 be the automaton that recognize language A. The procedure shown in Figure 8 can be used to construct N that can recognize A^*.