CS 3719 (Theory of Computation and Algorithms) – Lecture 6

Antonina Kolokolova*

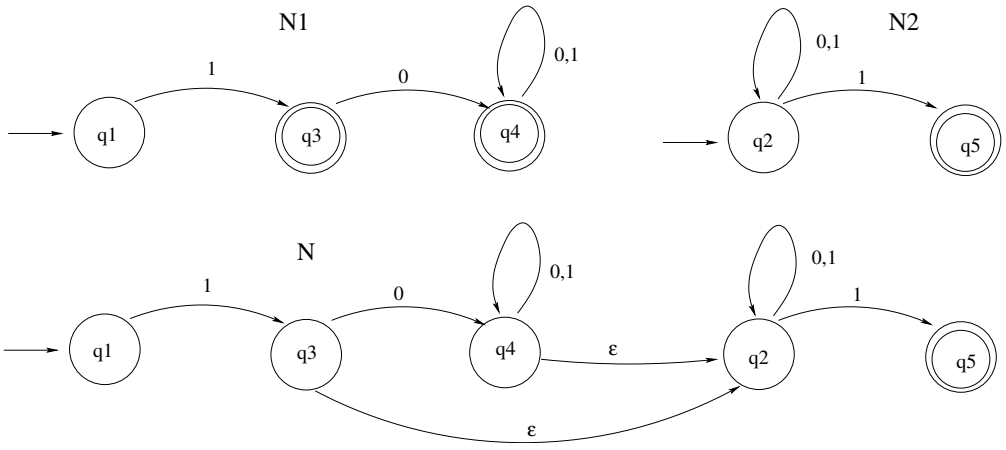
January 18, 2011

1 Closure under concatenation and star operations

Last time, we showed that the class of regular languages is closed under the Union operation (lemma 2.) Now we will finish the proof of Theorem 1 by stating how to show the closure of the class of regular languages under concatenation and star operations.

Continuation of the proof of Theorem 1. Recall that a language A is a concatenation of languages A_1 and A_2 if every string in A is of the form $w = \{xy | x \in A_1 \wedge y \in A_2\}$. Let $N_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma_2, \delta_2, q_2.F_2)$ be NFAs accepting A_1 and A_2, respectively, Then define an NFA N accepting A as follows: $Q = Q_1 \cup Q_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $q_0 = q_1$, $F = F_2$ and δ consists of δ_1, δ_2 and ϵ-transitions between states from F_1 and q_2. So N' non-deterministically decides that a string from A_1 has ended and a string from A_2 started.

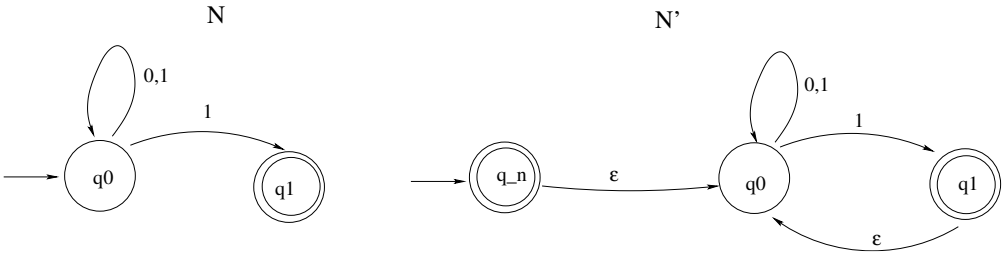
Example 1. Here are two NFAs N_1 and N_2 and their concatenation:



*The material in this set of notes came from many sources, in particular "Introduction to Theory of Computation" by Sipser and course notes of U. of Toronto CS 364.

Now, let's consider the case of the Star operation. Let N be an NFA, $\mathcal{L}(N) = A$. A natural approach to build N' with $\mathcal{L}(N') = A^*$ could be to just add an ϵ-arrow from every final state N to its start state. This almost works, except for the case of the empty string as an input, which is in A^* by definition. To avoid this problem, after adding ϵ-arrows from the final states of N to its start state, add one more accepting state and make it the start state with ϵ-arrow from it to the original start. That is, $Q' = Q \cup \{q_{newstart}\}$, $F' = F \cup \{q_{newstart}\}$, $q'_0 = q_{newstart}$ and δ contains extra transitions from $q_{newstart}$ to q_0 of N and from each state in F to q_0.

Example 2. Here is an NFA for A_2 from the previous example and an NFA for A_2^*



□

2 Equivalence of NFAs and DFAs

When we defined regular languages, we said that a language is regular if it is accepted by some finite automaton, without really specifying whether it is a DFA or an NFA. It is clear that if a language can be recognized by some DFA can also be recognized an NFA, at least because DFAs are a special case of NFAs. But what about the other direction, are there languages that can be done by NFAs but not DFAs? In general, does non-determinism make the model of computation more powerful?

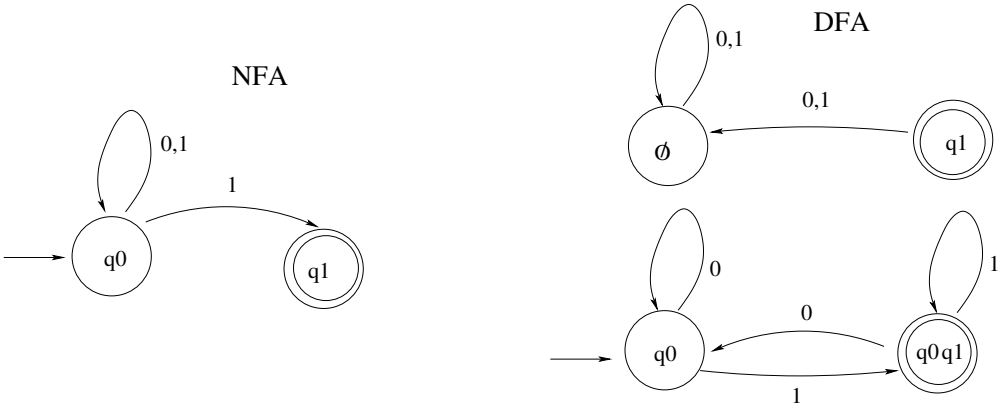
In this section we will show how to simulate NFAs by DFAs, althought the simulation is not very efficient. To the most general question, about the power of non-determinism, there is no single answer: sometimes non-determinism does not help much (as for the finite automata or Turing machines), sometimes it allows the model to recognize languages it would not be able to recognize without (that will be the case for pushdown automata, which we will talk about next week) and sometimes the answer is not known: a major, literally million-dollar, problem P vs. NP asks whether non-deterministic Turing machines can be simulated efficiently by deterministic ones. We will talk about this problem at length at the last part of the course.

Theorem 3. *For every NFA N there is a DFA D such that $\mathcal{L}(N) = \mathcal{L}(D)$.*

Corollary 4. *The class of languages accepted by NFAs is the same as the class of languages accepted by NFAs (regular languages.)*

Proof. The idea of the construction is as follows. Think of tracking an execution of N on a string. On every step of the NFA, maintain a set of states in which N could be at the moment. Then, on a next symbol, see where it is possible to get from any of the current states on that symbol. If there are ϵ-transitions, then every time after computing the states reachable from the current also add to the set any states reachable from the computed ones on ϵ-arrows.

Example 3. Here is an example of an NFA and a corresponding DFA. Consider an execution of this NFA on a string 1101. The first state is q_0, after that, on seeing a 1, it can go to either q_0 or q_1; record it as being in a state $\{q_0, q_1\}$. After seeing another 1, it can again go to either q_0 or q_1 already from q_0, so both states are possible. From there it will go back to q_0 on a 0 since there are no transitions from q_1 on 0, and from q_0 on 0 N stays in q_0. And, finally, it will go again to $\{q_0, q_1\}$ on 1. Since this means it is possible to end in an accepting state q_1, we treat $\{q_0, q_1\}$ as an accepting state.

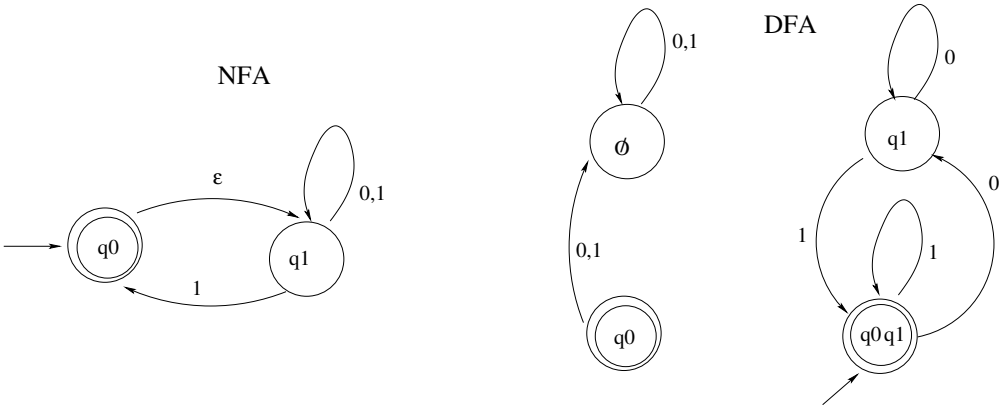


To formalize this intuition, let's describe $D(Q', \Sigma, \delta', q_0, F\}$. We will have to talk about sets of states reachable from a given set by following ϵ-arrows: let's call, for a set of states Q_i, a $E(Q_i) = \{q \in Q | \exists q' \in Q$ such that q is reachable from q' by following ϵ-arrows$\}$.

- Set $Q' = \mathcal{P}(Q)$, that is, a set of all subsets of Q. Thus, this conversion is not very efficient: the new DFA is exponentially larger than the original NFA.
- The new start state will be all states accessible from q_0 by following ϵ-arrows. That is, $q'_0 = E(\{q_0\})$.
- The final states of D are all sets that contain at least one final state of N. That is, $F' = \{Q_i | \exists q \in F, q \in Q_i\}$.
- Finally, we need to define the transition function of D, $\delta'(Q_i, a)$. It should be a set of states reachable from states in Q_i by doing δ-transitions on a, together with everything else that can be reached from them by ϵ-arrows. More formally, $\delta'(Q_i, a) = \{q | q \in E(\delta(q', a))$ for some $q' \in Q_i\}$.

□

Example 4. Here is an example of an NFA with ϵ-transitions and a corresponding DFA. Note that the start state is now $\{q_0, q_1\}$, since q_1 is reachable from q_0 by an ϵ-transition. Also, q_1 is reachable from q_1 two different ways: by the self-loop, and by going to q_0 and back on ϵ-arrow.



Note that in these two examples there are groups of states that are not reachable from the start. If this is the case, then we often just draw the part of the automaton which is reachable, ignoring the states we never get to.