CS340: Theory of Computation Sem I 2021-22

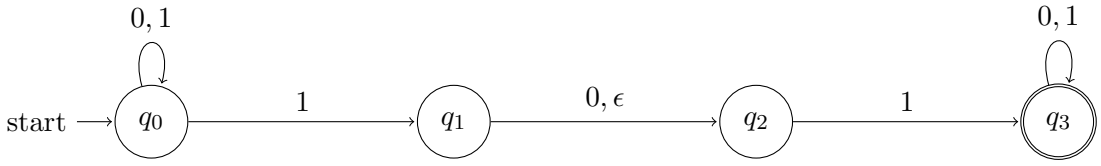
Lecture Notes 3: Nondeterministic Finite Automaton

Raghunath Tewari IIT Kanpur

1 Nondeterminism

- From a state q on an input symbol a there can be 0 or more transitions to other states (outgoing edges).
- The automaton *simultaneously* moves to all the states in *one* move.
- Here transitions can also be labelled by ϵ. On an epsilon transition, the automaton moves to the corresponding state without reading any input bit.
- The automaton accepts its input if at least one of the *computation paths* lead to an accept state.

Let us look at an example before moving further. Here is an example of a nondeterministic finite automaton (NFA in short).



- $0, 1$ or many transitions from a state on seeing a symbol. Such as
  - $(q_0, 1) \longrightarrow \{q_0, q_1\}$
  - $(q_1, 1) \longrightarrow \{\}$
  - $(q_1, 0) \longrightarrow \{q_2\}$
- Labels on transitions marked with symbols from $\Sigma \cup \{\epsilon\}$. If there is a transition from a state q_i to another state q_j labelled with ϵ, then the automaton simultaneously moves to the states q_i (i.e. stays at the same state) and q_j without reading any input symbol. Example
  - $(q_1, \epsilon) \longrightarrow \{q_1, q_2\}$

1.1 Computation of an NFA on a given input

Let us now see how computation happens on an NFA, given an input.
Input: Let the input string be 010110.

Current set of positions of the NFA	Symbol Read
$\{q_0\}$ (Initial position of the NFA)	0
$\{q_0\}$	1
$\{q_0, q_1, q_2\}$	0
$\{q_0, q_2\}$	1
$\{q_0, q_1, q_2, q_3\}$	1
$\{q_0, q_1, q_2, q_3\}$	0
$\{q_0, q_2, q_3\}$	Input is accepted since q_3 is an accept state

You may want to picture a nondeterministic computation as follows. From a state q, on an input symbol a, if there are $k(\geq 0)$ outgoing edges then imagine that the automaton "consumes" the bit a, makes k copies of itself at that instant with each copy going to one of the outgoing states. If $k = 0$, then that particular copy of the automaton "dies" on seeing the symbol a. Similarly, from a state q, if there are $k(\geq 0)$ ϵ edges, then the automaton makes $k + 1$ copies of itself at that instant with each copy going to one of the outgoing states and one copy staying at the current state without consuming any input bit.
Computation proceeds in parallel in all the copies. We usually refer to the different copies of the automaton as different *computation paths*.
The input is accepted if the final set of states of the automaton after it has read the input completely, has an accept state.

Remark. Nondeterminism is not a realistic model in the sense that there are no nondeterministic computers in the real world. However often it makes description of the automaton easier.

1.2 Difference between deterministic and nondeterministic computations

Deterministic Computation	**Nondeterministic Computation**
(state, input symbol) pair $\longrightarrow$ A SINGLE state	(state, input symbol) pair $\longrightarrow$ MULTIPLE state (≥ 0) simultaneously
a single computation path	multiple computation paths
computation is at a unique state in each step	computation is in a set of states in each step
input accepted if computation path leads to an accept state	input accepted if at least one computation path leads to an accept state
input rejected if computation path does not lead to an accept state	input rejected if none of the computation paths lead to an accept state
does not allow ϵ transitions	allows ϵ transitions

2 Nondeterministic Finite Automaton

Let $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$. For a set A, let 2^A denote the power set of A.

Definition 2.1. A *nondeterministic finite automaton* (in short NFA) is a 5 tuple $N = (Q, \Sigma, \delta, q_0, F)$, where,

- Q is a finite set called the set of *states*,
- Σ is a finite set called the *alphabet*,
- $\delta : Q \times \Sigma_\epsilon \to 2^Q$ is the *transition function*,
- $q_0 \in Q$ is the *initial state*, and
- $F \subseteq Q$ is the set of *final states*.

We say that N *accepts* $w = a_1 a_2 \dots a_n$ if we can write w as $w = b_1 b_2 \dots b_m$, where each $b_i \in \Sigma_\epsilon$ and there exists a sequence of states $r_0, r_1, \dots, r_m \in Q$ (not necessarily distinct), such that

1. $r_0 = q_0$,
2. $r_i \in \delta(r_{i-1}, b_i)$ for $i = 1, 2, \dots m$
3. $r_m \in F$.

Then,

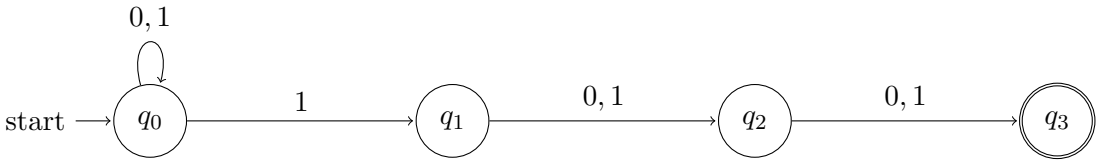
$$L(N) = \{w \in \Sigma^* \mid N \text{ accepts } w\}.$$

2.1 Examples of NFA

1. Consider the following language

   $$L = \{w \in \{0,1\}^* \mid \text{the 3rd last symbol of } w \text{ is } 1\}$$

   Below is an NFA for the above language.



   Exercise 1. Now construct a DFA for the above language. What can you say about the size (i.e. no. of states) of the DFA compared to the NFA? Consider the language

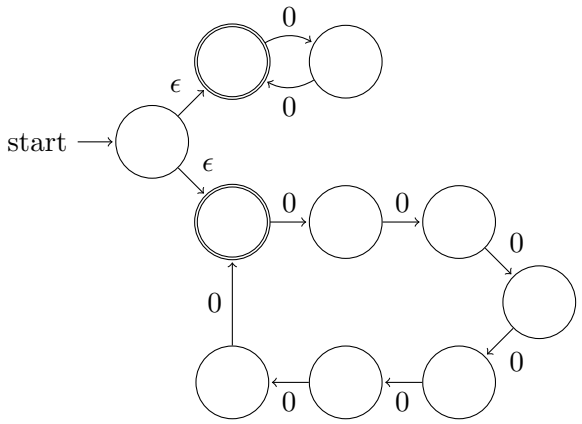
   $$L_k = \{w \in \{0,1\}^* \mid \text{the } k\text{-th last symbol of } w \text{ is } 1\}$$

   What is the smallest sized NFA that can accept L_k (as a function of k)? What about the smallest sized DFA?

2. Recall the following language discussed earlier

$$L = \{w \in \{0\}^* |\ |w| \text{ is divisible by 2 or 7}\}$$

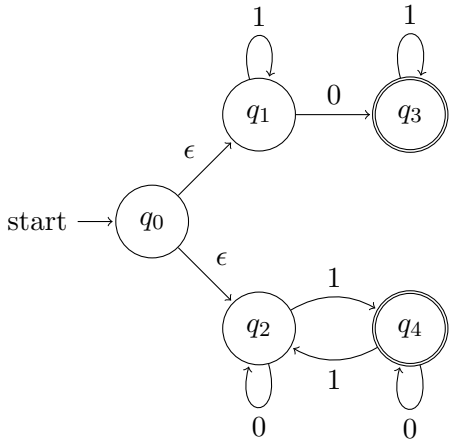
Below is an NFA that accepts L.



How does the size of the DFA and NFA compare with each other for L?

3. Another example.

$$L = \{w \in \{0\}^* |\ \#_0(w) = 1 \text{ or } \#_1(w) \text{ is odd}\}$$



Exercise 2. Problem 1.5 from chapter 1 in the textbook.

Note 1. Observe that a DFA is also an NFA. It just so happens that a DFA does not use its nondeterminism. Therefore if $\mathcal{N}$ is the class of all languages accepted by some NFA then regular languages are a subset of $\mathcal{N}$. But what about the converse?

Theorem 1. *Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Then there exists a DFA $M = (Q', \Sigma, \delta', q_0', F')$ such that $L(N) = L(M)$.*

We will give the construction of the DFA M. The idea is to consider all subset of states of N.

Construction of M

1. $Q' = 2^Q$.
2. Let R be a state in Q' and $a \in \Sigma$. Then by definition of Q', $R \subseteq Q$. We define δ' as follows:
$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$
3. $q'_o = \{q_0\}$.
4. $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$.

What about ϵ-transitions?
Let $R \subseteq Q$. Define the ϵ-closure of R as,

$$E(R) = \bigcup_{r \in R} \{q \mid q \text{ can be reached from } r \text{ by using 0 or more } \epsilon\text{-transitions}\}.$$

The transition function δ' can be modified as

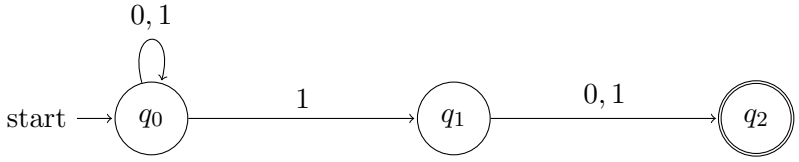
$$\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a)),$$

and $q'_o = E(\{q_0\})$.

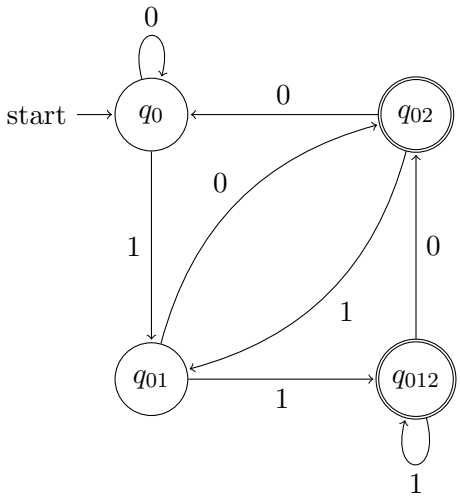
Corollary 2. *A language is regular if and only if it is accepted by some NFA.*

An Example

Consider the language $L = \{w \in \{0,1\}^* \mid \text{the 2nd last symbol of } w \text{ is } 1\}$. Here is an NFA for L.



Based on the construction given earlier, the corresponding DFA for L will be as follows.



Although there are $2^3 = 8$ subsets of the set $\{q_0, q_1, q_2\}$, but only 4 of them are *reachable* from the start state q_0. Hence it is sufficient to consider these 4 states.

Remark. States of a DFA/NFA that are not reachable from the start state via any sequence of symbols are redundant and therefore, need not be considered.

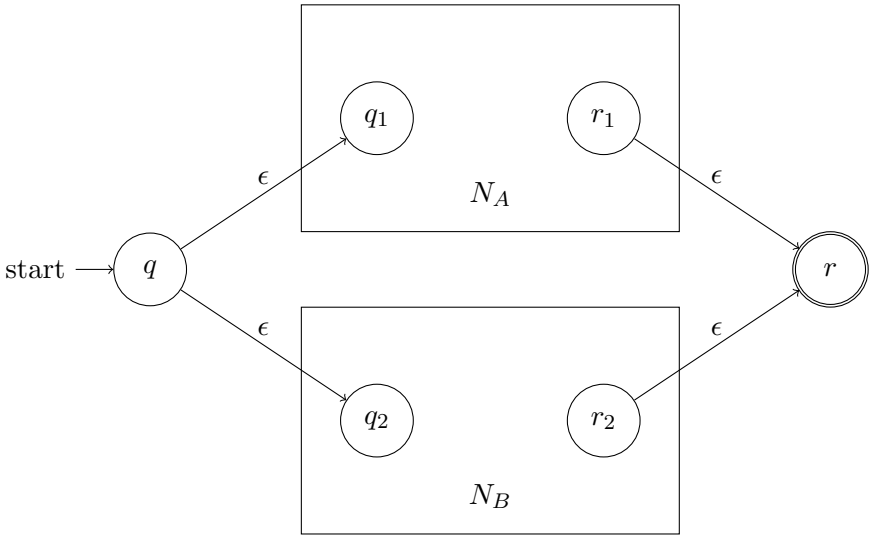
3 Closure properties of Regular Languages

Recall the following theorem discussed in the previous Lecture Notes.

Theorem 3. *Let A, B be two regular languages. Then $A \cup B$, AB and A^* are also regular. In other words, regular languages are closed under the union, concatenation and star operations.*

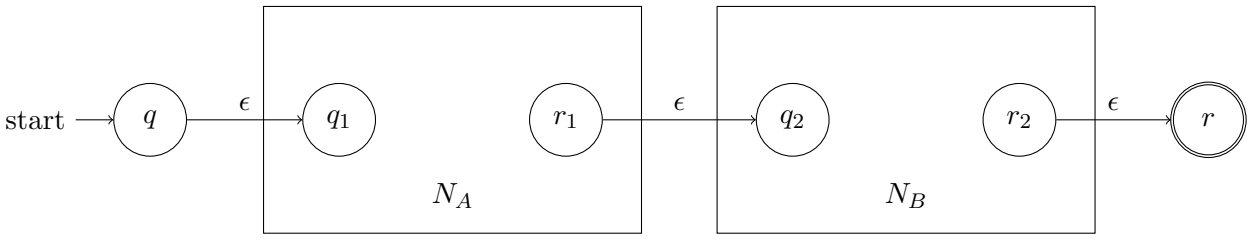
We will give a constructive proof of the theorem. Let N_A and N_B be two NFA's accepting A and B respectively. We assume without loss of generality that N_A and N_B have a unique start state and a unique accept state, with no incoming transitions to the start state and no outgoing edges from the accept state. This is because if the NFA has more than one accept state, then we add a new accept state to the NFA and add ϵ-transitions from the old accept states to the new accept state.

1. **Union operation**



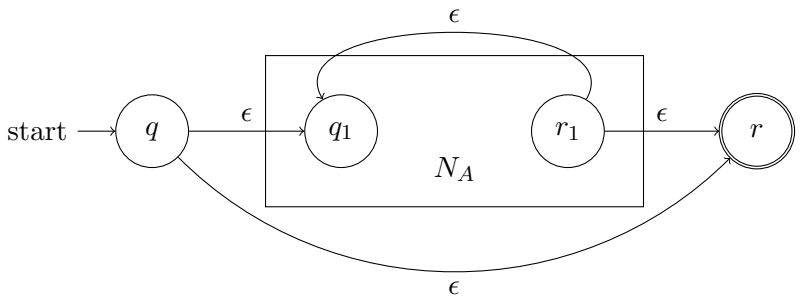
Construction of an NFA that accepts the language $A \cup B$.

2. **Concatenation operation**



Construction of an NFA that accepts the language AB.

3. **Star operation**



Construction of an NFA that accepts the language A^*.

Exercise 3. Read Section 1.2 from textbook.