CS 208 : Automata Theory and Logic Feb 13' 2024

Lecture - 15

Topic: Non-Deterministic Finite Automata (NDFA)

Scribed by: Satyankar Chandra (22B0967)
Checked and compiled by:

In the last few lectures, we covered the formalization of deterministic finite automata (DFA) where the transition function outputs a single state for a given input and current state. In this lecture, we will discuss non-deterministic finite automata (NDFA) where the transition function can output multiple states (or a set of states) instead.

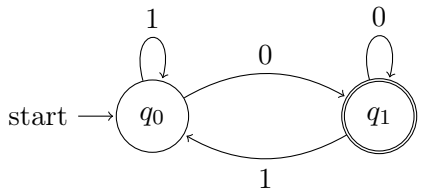
1 Introduction to NDFAs

1.1 Formalization of Deterministic Finite Automata

As discussed earlier, a DFA is a 5-tuple $(Q, \Sigma, q_0, \delta, F)$ where:

- Q is a finite set of all states
- Σ is the alphabet, a finite set of input symbols
- $q_0 \in Q$ is the initial state
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $F \subseteq Q$ is the set of final/accepting states

For example, consider the following automaton:



It can be represented as:

$$\text{DFA}\ (\ \{q_0, q_1\},\ \{0, 1\},\ q_0,\ \delta,\ \{q_1\}\ )$$

where δ is the transition function:

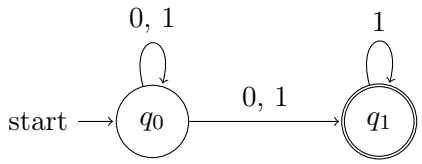
Q	Σ	Q'
q_0	0	q_1
q_0	1	q_0
q_1	0	q_1
q_1	1	q_0

1.2 Formalization of Non-Deterministic Finite Automata

However, in NDFA,

- $q_0 \subseteq Q$ is the set of initial states
- $\delta : Q \times \Sigma \to 2^Q$ is the transition function

Hence, consider the following non-deterministic finite automaton (A):



It can be represented as:

$$\text{NDFA}\left(\ \{q_0, q_1\},\ \{0, 1\},\ \{q_0\},\ \delta',\ \{q_1\}\ \right)$$

where δ' is the transition function:

Q	Σ	2^Q
q_0	0	$\{q_0, q_1\}$
q_0	1	$\{q_0, q_1\}$
q_1	1	$\{q_1\}$

As we can see, δ' is a partial function whose output is a set of states instead of a single state.

2 Acceptance of a String by an NDFA

Due to its transition function, an NDFA gives **choices for path taken** at some states for a given input string. Any string for which there exists a path from the initial state to a final state is considered to be accepted by the NDFA.

Hence, the NDFA shown above has the language $L(A) = \Sigma^* \setminus \{\epsilon\}$, i.e., the set of all strings over the alphabet Σ except the empty string.

For example, the string 011 is accepted by A as it has the following path $q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_1$.

To determine if a string is accepted by an NDFA, we can check if the set of states reachable from the initial state by reading the string contains any final state.

Here we have:

Initial State	Σ^*	Reachable States
q_0	0	$\{q_0, q_1\}$
q_0	01	$\{q_0, q_1\}$
q_0	011	$\{q_0, q_1\}$

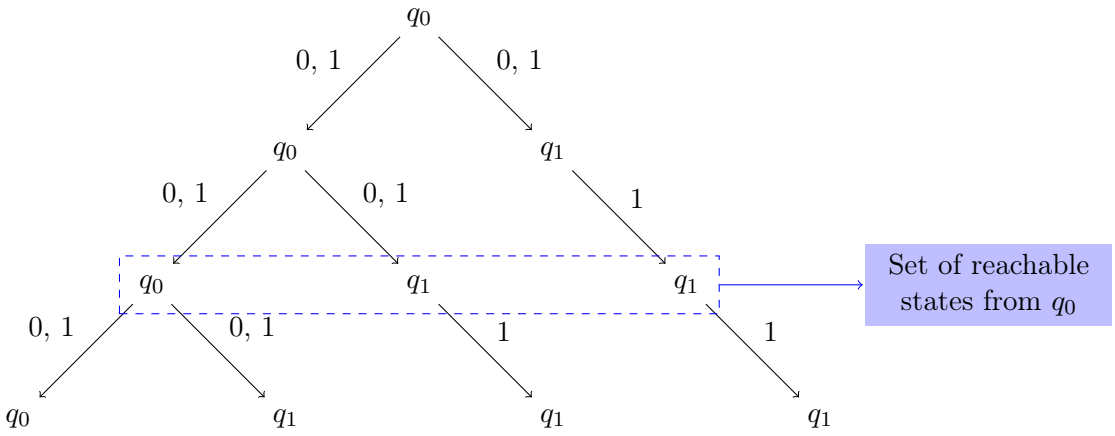
As $\{q_0, q_1\}$ contains $q_1 \in F$, the string 011 is accepted by A. By construction, there will always be a set of choices which reach q_1 from q_0 for the input 011.

3 Representing NDFAs as DFAs

Even though NDFA gives choices at some states, it can still be represented as some equivalent DFA. This implies that **NDFAs have no more expressive power than DFAs** in terms of string acceptance. However, representation in form of NDFA is much more succinct than DFA.

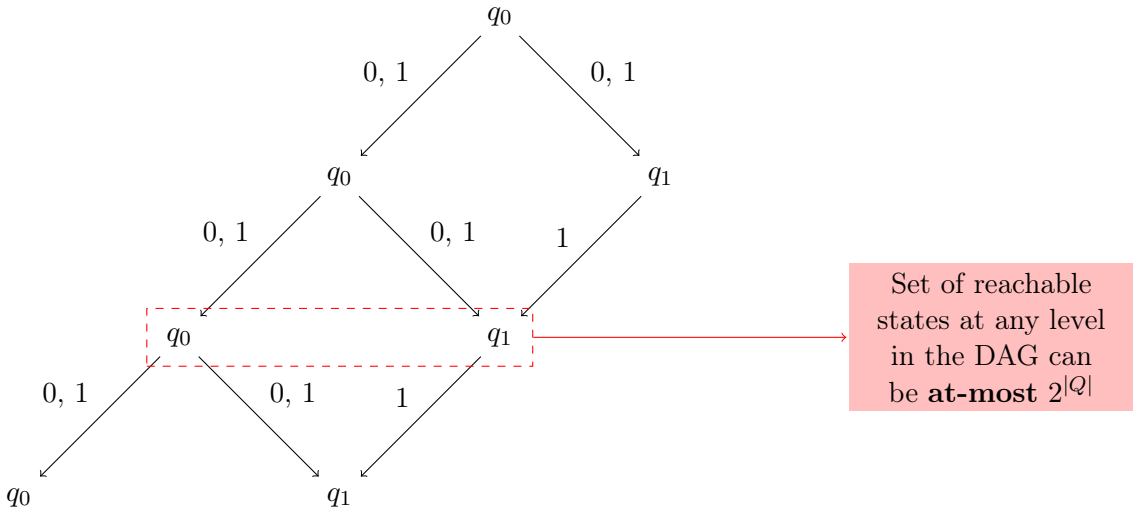
We can convert a NDFA to a DFA by considering the set of reachable states as states of the equivalent DFA.

Let us again consider the previous NDFA A. Since there is only one initial state q_0, we can model the set of reachable states from q_0 for strings in Σ^* as a tree. Branches in the tree correspond to choices available in the NDFA for that state.

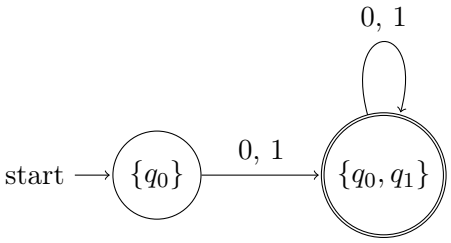


As we can see, any level of the tree contains the set of reachable states from the initial state for strings of length equal to the level. Note that it is redundant to display the same state (e.g., q_1) multiple times in the same level. Hence, we can compress this tree to a Directed Acyclic Graph (DAG) by merging the same states at the same level.

Since the number of subsets of set Q is $2^{|Q|}$, the DAG can have at-most $2^{|Q|}$ states at any level.



From the above DAG, we can try forming an equivalent DFA keeping track of the set of states reachable by exercising some set of choices. Note that we don't care about the choices themselves, only for the fact that reachable set of states should contain some accepting state.



Equivalent DFA for the NDFA A

Hence we can see how to convert a NDFA with $|Q|$ states to an equivalent DFA with at-most $2^{|Q|}$ states. In this DFA, the automata on set of reachable state is completely deterministic.

3.1 Parameters of the Equivalent DFA

Let's now write the DFA parameters from parameters of the original NDFA. Suppose the NDFA is $N = (Q, \Sigma, q_0, \delta, F)$ and the equivalent DFA is $D = (Q', \Sigma', q_0', \delta', F')$. Then:

- $Q' = 2^Q$ (set of all subsets of Q)
- $\Sigma' = \Sigma$ remains the same
- δ' can be computed from δ as follows:
  Suppose $S \in Q'$ is the current state. Then, for all $a \in \Sigma$,

  $$\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)$$

  Note that $\delta : Q \times \Sigma \to 2^Q$ and hence δ' is well-defined and returns a set of states.
- $q_0' = q_0$ (set of initial states)
- $F' = \{S \in Q' \mid S \cap F \neq \emptyset\}$ (set of states containing at least one final state)

3.2 Proof of equivalence

We can prove that the equivalent DFA D accepts exactly the same language as the original NDFA N, i.e, $L(D) = L(N)$.

For this we can show 2 things:

- $L(D) \subseteq L(N)$
- $L(N) \subseteq L(D)$

We can prove both of these things by induction. This will be covered in detail in the next lecture.