NFA

Noman Amin
Lecturer, Dept. of CSE, Southeast University

Nondeterministic Finite Automata

A "nondeterministic" finite automaton (*NFA*) has the power to be in several states at once. This ability is often expressed as an ability to "guess" something about its input. For instance, when the automaton is used to search for certain sequences of characters (e.g., keywords) in a long text string, it is helpful to "guess" that we are at the beginning of one of those strings and use a sequence of states to do nothing but check that the string appears, character by character.

Nondeterministic Finite Automata

- In NDFA, for a particular input symbol, the machine can move to any combination of the states in the machine. In other words, the exact state to which the machine moves **cannot be determined.** Hence, it is called Non-deterministic Automaton. As it has finite number of states, the machine is called Non-deterministic Finite Machine or Non-deterministic Finite Automaton.

An Informal View of NFA

Like the DFA, an NFA has a finite set of states, a finite set of input symbols, one start state and a set of accepting states. It also has a transition function, which we shall commonly call δ. The difference between the DFA and the NFA is in the type of δ. For the NFA, δ is a function that takes a state and input symbol as arguments (like the DFA's transition function), but returns a set of zero, one, or more states (rather than returning exactly one state, as the DFA must). We shall start with an example of an NFA, and then make the definitions precise.

Example

Example : Figure 2.9 shows a nondeterministic finite automaton, whose job is to accept all and only the strings of 0's and 1's that end in 01. State q_0 is the start state, and we can think of the automaton as being in state q_0 (perhaps among other states) whenever it has not yet "guessed" that the final 01 has begun. It is always possible that the next symbol does not begin the final 01, even if that symbol is 0. Thus, state q_0 may transition to itself on both 0 and 1.

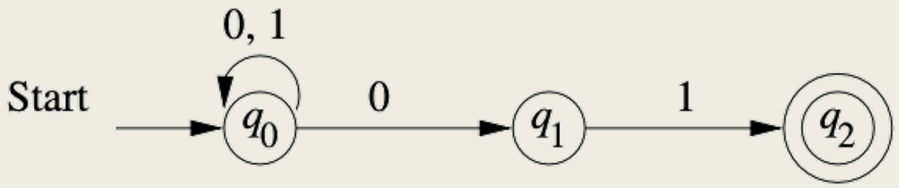


Figure 2.9: An NFA accepting all strings that end in 01

Example

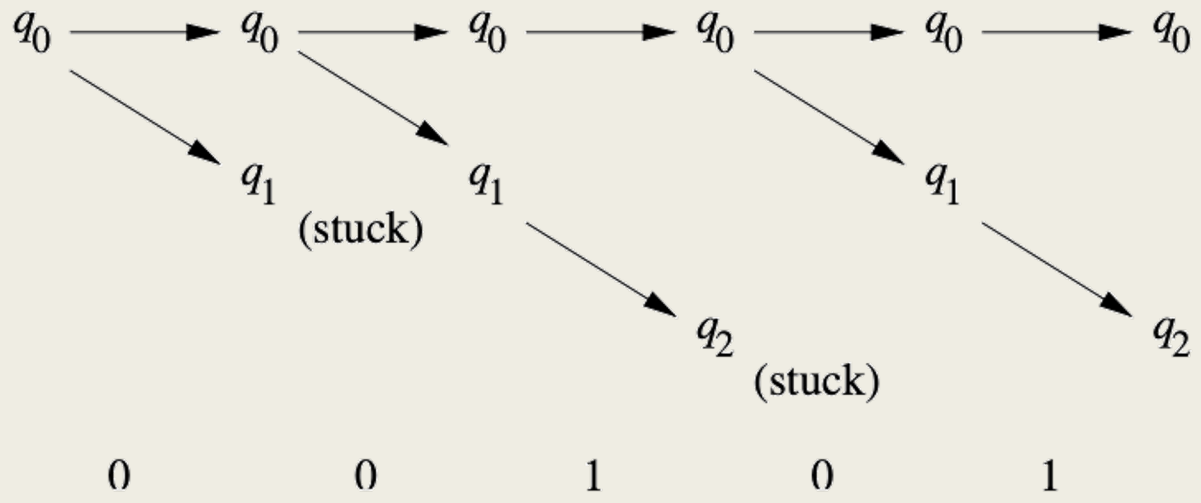


Figure 2.10: The states an NFA is in during the processing of input sequence 00101

Definition of NFA

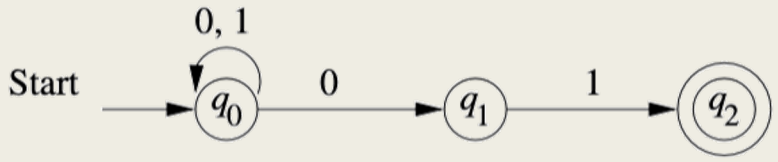
Now, let us introduce the formal notions associated with nondeterministic finite automata. The differences between DFA's and NFA's will be pointed out as we do. An NFA is represented essentially like a DFA:

$$A = (Q, \Sigma, \delta, q_0, F)$$

where:

1. Q is a finite set of *states*.
2. Σ is a finite set of *input symbols*.
3. q_0, a member of Q, is the *start state*.
4. F, a subset of Q, is the set of *final* (or *accepting*) states.
5. δ, the *transition function* is a function that takes a state in Q and an input symbol in Σ as arguments and returns a subset of Q. Notice that the only difference between an NFA and a DFA is in the type of value that δ returns: a set of states in the case of an NFA and a single state in the case of a DFA.

Example



Example : The NFA of Fig. 2.9 can be specified formally as

$$(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$$

where the transition function δ is given by the transition table of Fig. 2.11. □

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

Figure 2.11: Transition table for an NFA that accepts all strings ending in 01

Extended Transition Function

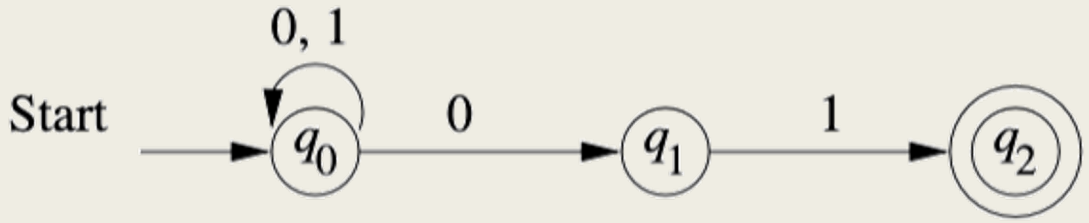
Example : Let us use $\hat{\delta}$ to describe the processing of input 00101 by the NFA of Fig. 2.9. A summary of the steps is:

1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$.
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.
5. $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
6. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.

NFA to DFA Conversion

- **Step 1 – NFA Transition Table**
  - *List all NFA states, input symbols, and transitions in table form.*
- **Step 2 – DFA Start State**
  - *Take ε-closure of NFA start state ⟶ this becomes DFA start state.*
- **Step 3 – Construct DFA States & Transitions**
  - *Each DFA state = a set of NFA states.*
  - *For each DFA state and input symbol, compute move + ε-closure and add as new DFA state if needed.*
- **Step 4 – Identify DFA Final States**
  - *Any DFA state that contains at least one NFA final state is a final state.*
- **Step 5 – Simplify / Minimize DFA**
  - *Remove unreachable states.*
  - *Remove dead (non-accepting, non-progressing) states.*
  - *Merge equivalent states (state minimization).*
- **Step 6 – Result**
  - *Obtain the minimized DFA equivalent to the given NFA.*

Example



	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

Example

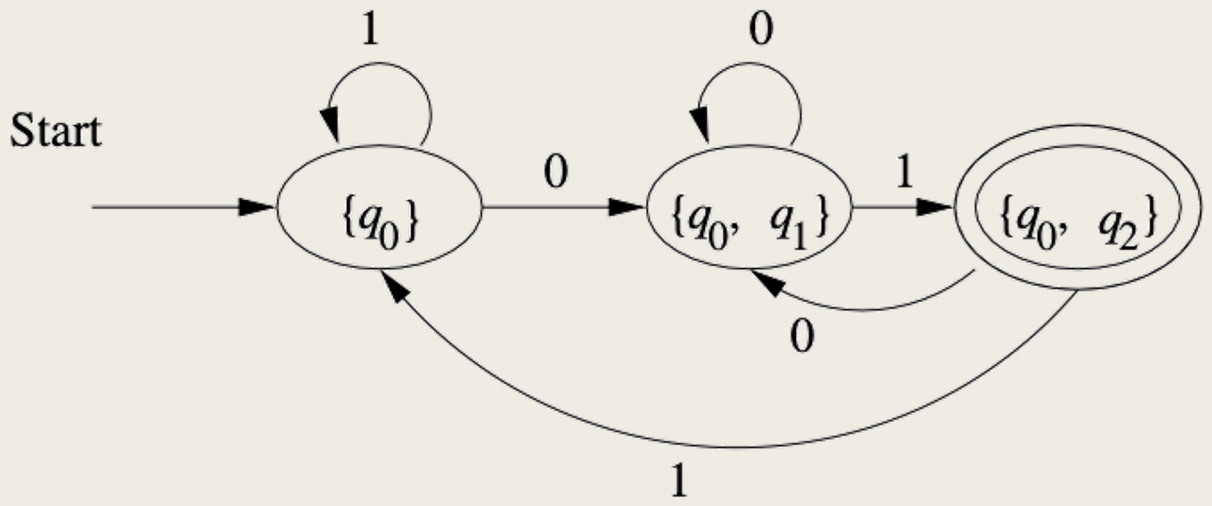


Figure 2.14: The DFA constructed from the NFA of Fig 2.9

Exercise

*** Exercise** : Convert to a DFA the following NFA:

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	$\emptyset$
$*s$	$\{s\}$	$\{s\}$

Exercise

Exercise : Convert to a DFA the following NFA:

	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
$*q$	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
$*s$	$\emptyset$	$\{p\}$

Exercise

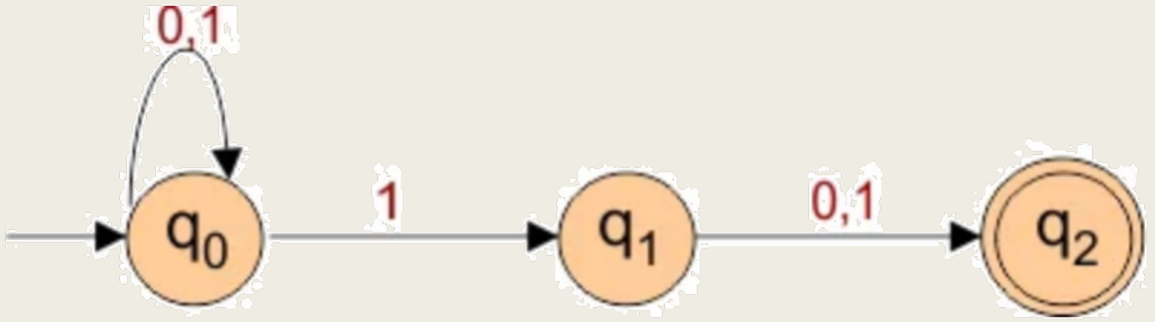
- Design an NFA over the alphabet Σ = {a, b} that accepts only the strings that start with “b”
- Design an NFA over the alphabet Σ = {a, b} that accepts only the strings that start with “bab”
- Design an NFA over the alphabet Σ = {a, b} that accepts only the strings that start with “aa” or “bb”
- Design an NFA with input alphabet $\sum$ = {a, b} that accepts all the strings that end with “bab”.
- Design an NFA with $\sum$ = {0, 1} for all binary strings where the second last bit is “1”.
- Draw an NFA with $\sum$ = {0, 1} such that the third symbol from the right is “1”.

Exercise

- Design an NFA with input alphabet $\sum$ = {0, 1} that accepts all the strings that end with “01”.
- design an NFA with input alphabet $\sum$ = {a, b} that accepts all the strings that end with “bbb” or “bab”.
- Construct an NFA with $\sum$ = {0, 1} that accepts all strings that begin with 1 and end with 0.
- Design an NFA with $\sum$ = {0, 1} such that every string must contain the substring “1101”.

NFA to DFA

- Draw an NFA that accepts all the strings ending with “1” over Σ {0,1} and convert this NFA to its corresponding DFA.
- The following is the NFA graph that has to be converted into a DFA.



What is a Dead State?

- A dead state is a state in a DFA from which no sequence of inputs can lead to an accepting (final) state. Once the automaton enters a dead state, it cannot recover or proceed to an accepting state regardless of the inputs that follow.