CSE-233 : Section A

Summer 2020

NFA

Reference:

Book2 Chapter 1.2

Md. Saidul Hoque Anik
anik@cse.uiu.ac.bd

Practice

Draw the state diagram of the NFA that can recognize the following languages:

- L(M) = { w | w begins with 101 } over Alphabet Σ = {0, 1}
- L(M) = { w | w begins with abb } over Alphabet Σ = {a, b}

- L(M) = { w | w ends with 101 } over Alphabet Σ = {0, 1}
- L(M) = { w | w ends with aa } over Alphabet Σ = {a, b}

- L(M) = { w | w contains with 110 } over Alphabet Σ = {0, 1}
- L(M) = { w | w contains with abb } over Alphabet Σ = {a, b}

- L(M) = { w | w is exactly 101 } over Alphabet Σ = {0, 1}

String Examples

Construct NFAs for the following languages over the alphabet {a, b, …, z}:

- All strings that contain **eat** or **sea** or **easy**
- All strings that contain both **sea** and **tea**
- All strings that do not contain **food**

String Examples

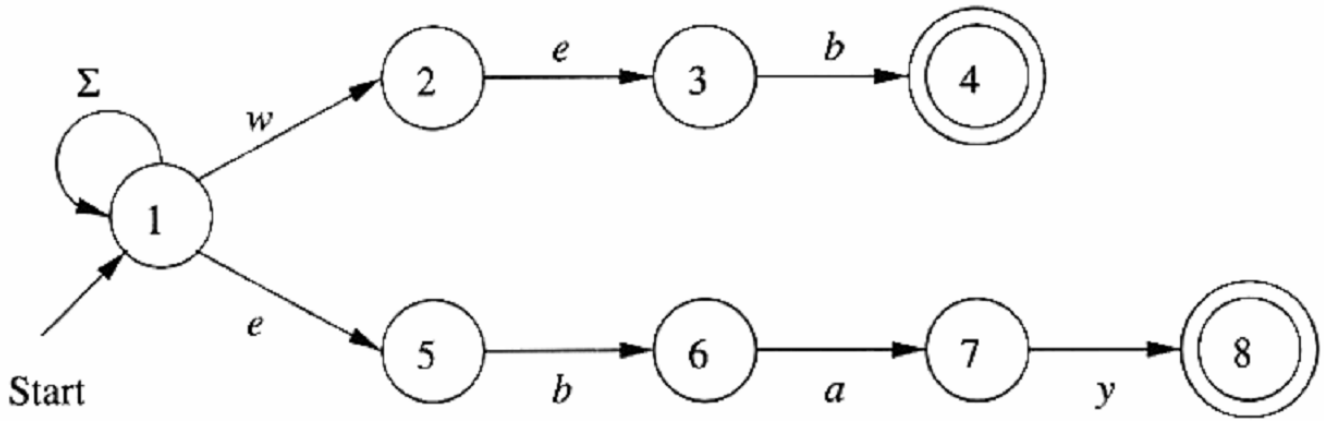


Figure 2.16: An NFA that searches for the words web and ebay

Equivalent DFA

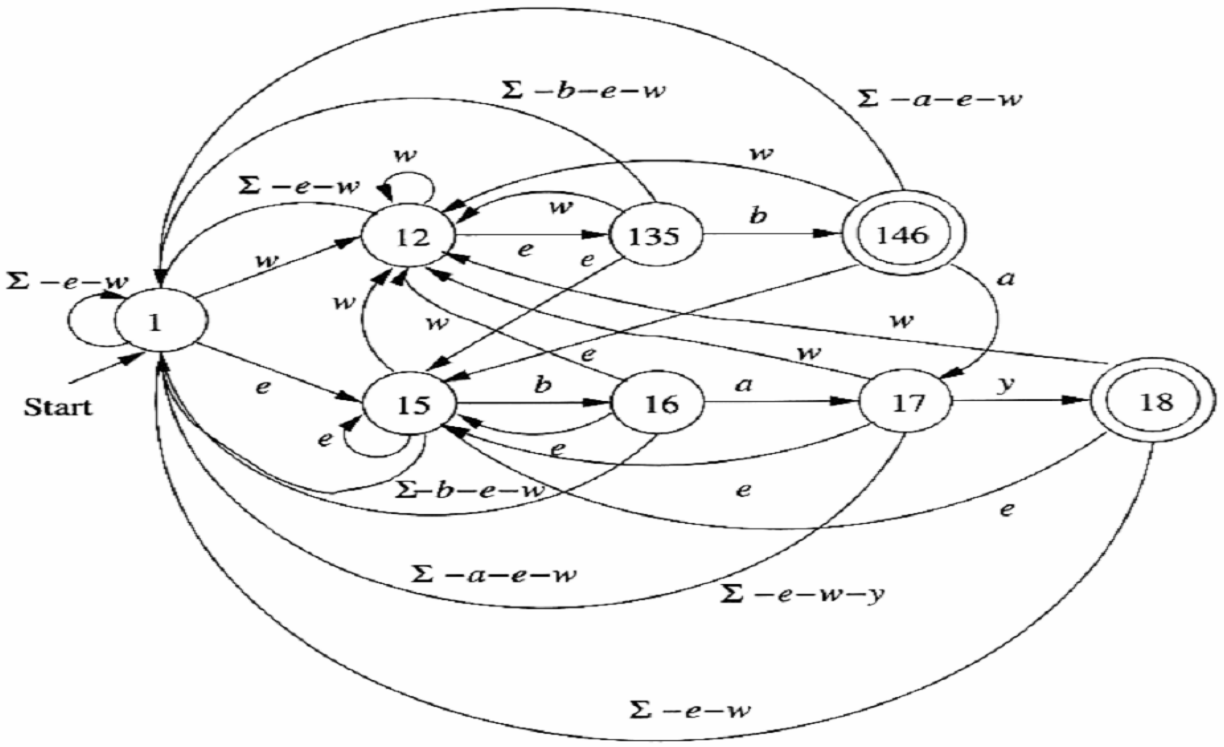
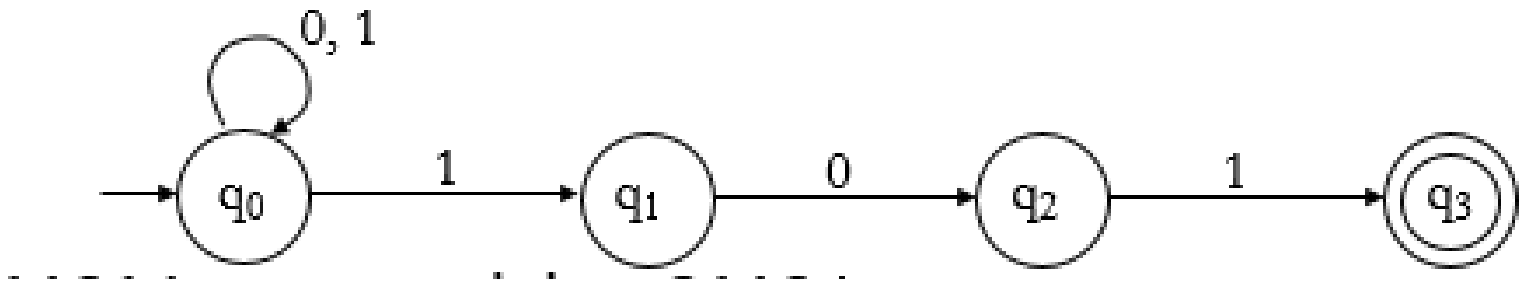


Figure 2.17: Conversion of the NFA from Fig. 2.16 to a DFA

Language of NFA

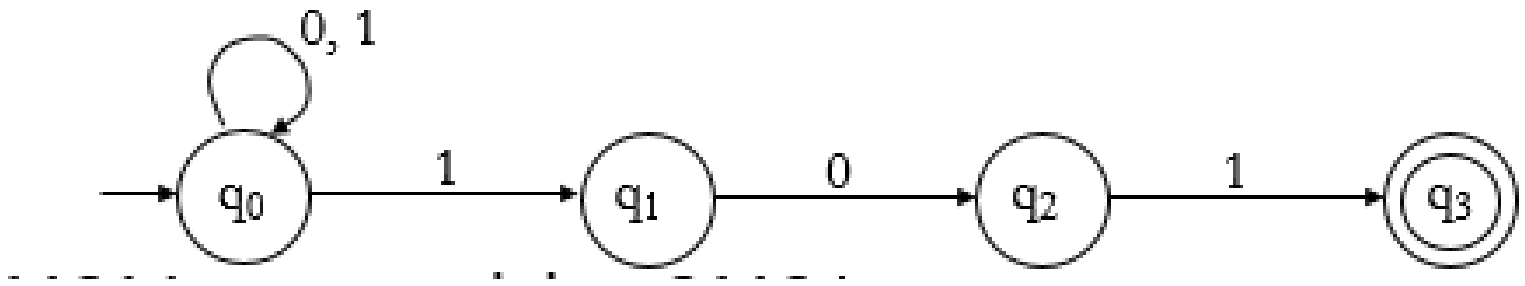


The language of an NFA is the set of all strings for which there is some path that, starting from the initial state, leads to an accepting state as the string is read left to right.

Does this NFA accept-

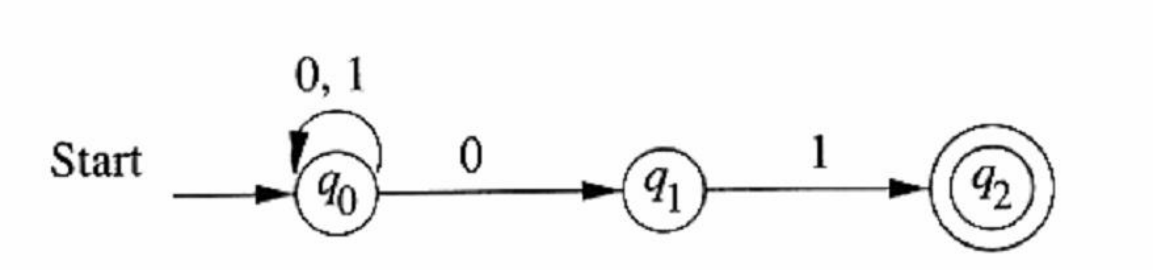
- 1101?
- 0110?

Language of NFA

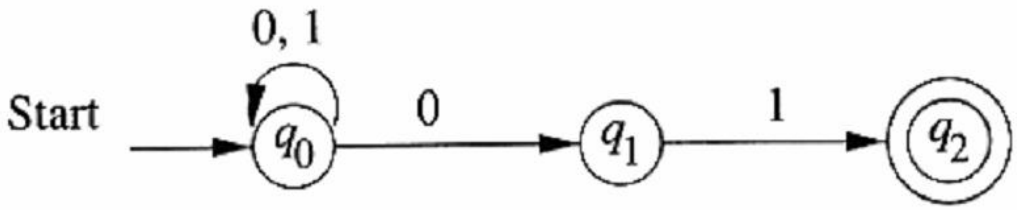


- A string w is accepted by an NFA if $\delta(q0, w)$ contains at least one final state.
- The language of the NFA is the set of strings it accepts.

Extended Transition Function

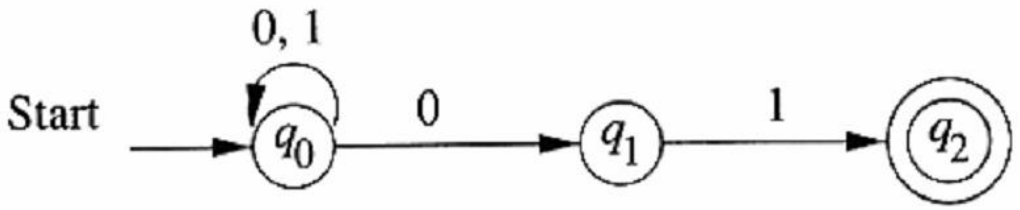


Extended Transition Function



1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.

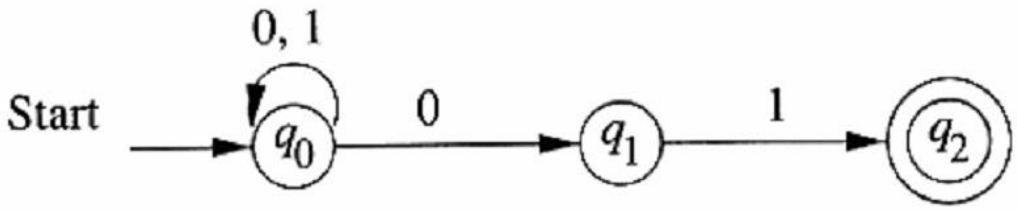
Extended Transition Function



1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.

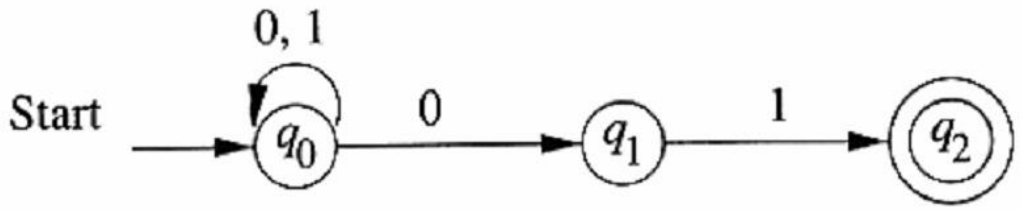
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$.

Extended Transition Function



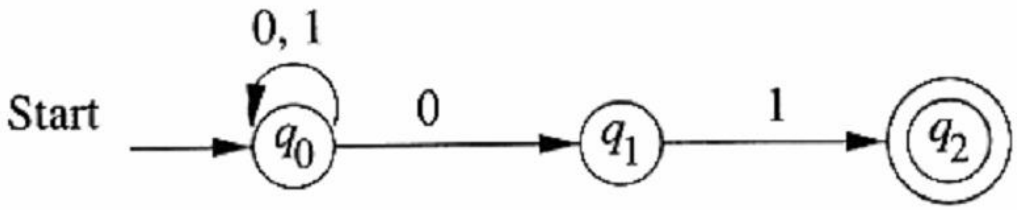
1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$.
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.

Extended Transition Function



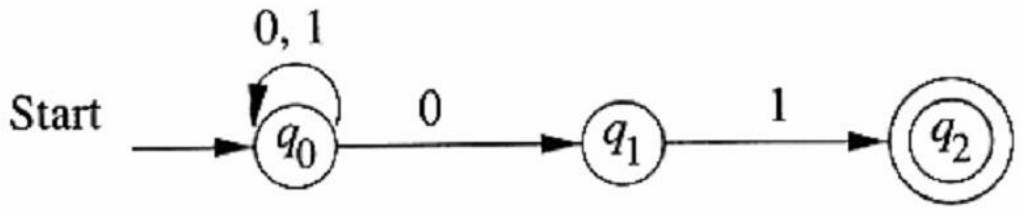
1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$.
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} =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

Extended Transition Function



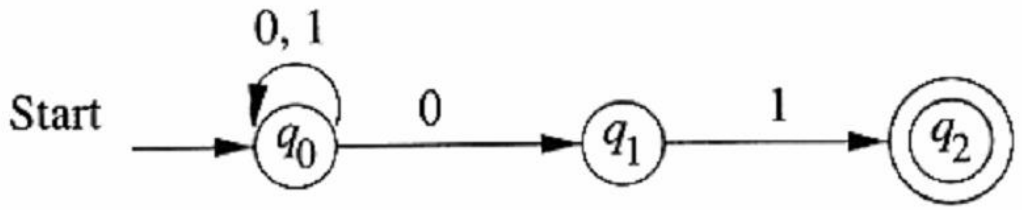
1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$.
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} =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

Extended Transition Function



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. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.

Extended Transition Function



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. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.

DFA vs. NFA

- NFA can do everything a DFA can do
- Can it do more than what a DFA can do?

Any NFA can be converted into equivalent DFA

- NFA can do everything a DFA can do
- Can it do more than what a DFA can do? NO!

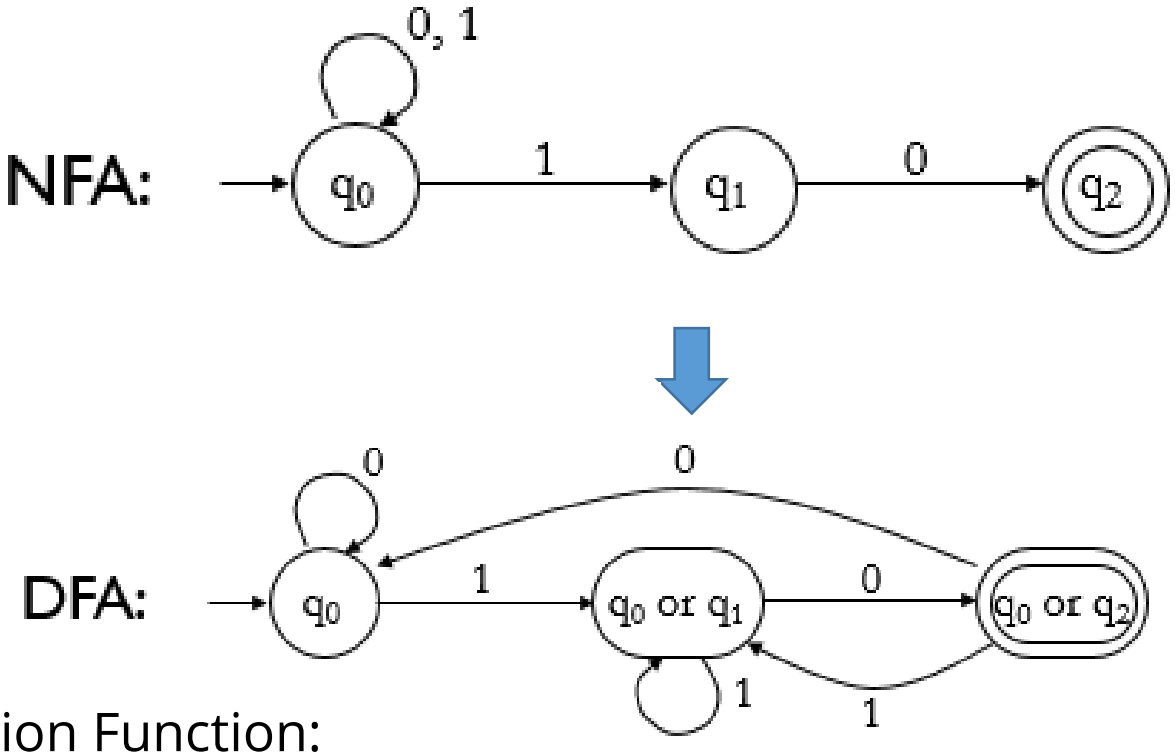
Theorem: A language L is accepted by some DFA if and only if it is accepted by some NFA.

Informal Proof

To prove the theorem, we have to show that for every NFA there is a DFA that accepts the same language

- We will give a general method for simulating any NFA by a DFA

Example



NFA Transition Function:

states	inputs 0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\varnothing$
q_2	$\varnothing$	$\varnothing$

We'll need to draw the equivalent function for DFA

Formal Description

	NFA	DFA
states	$q_0, q_1, \ldots, q_n$	$\{\ \}, \{q_0\}, \{q_1\}, \{q_0,q_1\}, \ldots, \{q_0,\ldots,q_n\}$ one for each subset of states in the NFA
initial state	q_0	$\{q_0\}$
transitions	d	$d'(\{q_{i1},\ldots,q_{ik}\}, a) =$ $d(q_{i1}, a) \cup \ldots \cup d(q_{ik}, a)$
accepting states	$F \subseteq Q$	$F' = \{S: S$ contains some state in $F\}$