Theory of Computation

Tutorial 5 - NFAs

Cesare Spinoso-Di Piano

Plan for today

1. NFAs
2. NFA-to-DFA

NFAs

Introduction to NFAs

Defintion. A **nondeterministic finite automaton (NFA)** M is a 5 element tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is the set of all states

Σ is the alphabet

* δ is the transition function $\delta : Q \times \{\Sigma \cup \{\lambda\}\} \to 2^Q$

q_0 is the (unique) initial state

F is the set of final states

A NFA is a machine that reads an input string and decides whether to accept it.

***Unlike a DFA:** The transition function of an NFA can accept λ and **always** returns a set.

Introduction to NFAs

Consider a NFA M
Given a string w, **M** tries all possible walks. If, at the end of the string, ANY of the walks end in a final state the string is accepted. Otherwise, it is rejected.

Definition. The language **L(M)** includes all strings (over the alphabet Σ) that are accepted by **M**.
$L(M) = \{$strings that drive **M** to a final state$\}$

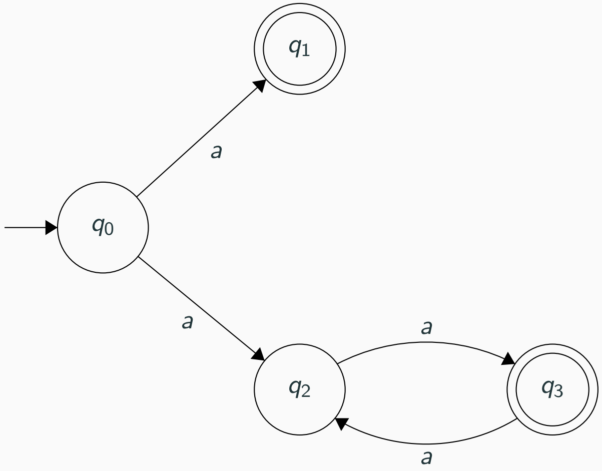
Formally:
$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w)$ contains at least one final state$\}$, where δ^* is the extended transition function $\delta^* : Q \times \Sigma^* \to 2^Q$.

Difference between DFA and NFA

	DFA	NFA
When read a symbol	$\forall a \in \Sigma$, only one path is possible	Set of possible transitions
λ-transition	Not capable	Capable
Structure	Can be best described and understood as one machine	Is more like multiple small machines that are performing computational activities at the same time
Accept a string	Input terminates in a final state	At least one walk terminates in a final state
Reject a string	Input doesn't terminate in a final state	All walks either 1. Leave the input unfinished 2. Do not end in a final state
Space requirement	More space allocation needed	Less space needed
Time Complexity	Less expensive	More expensive

Example

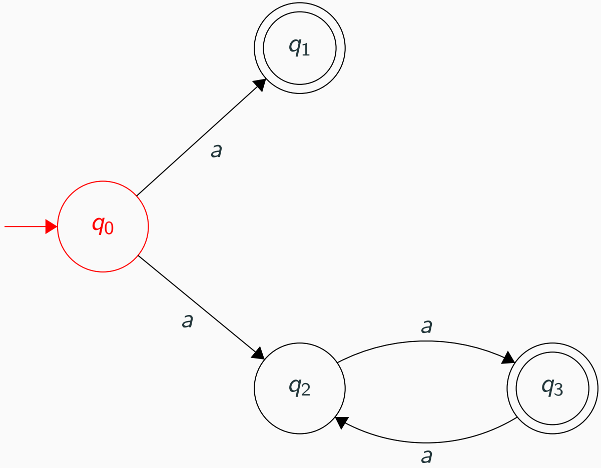
Example. The following is an NFA **M** where
$\mathbf{L(M)} = \{a\} \cup \{a^{2k} : k > 0\}$ $(\Sigma\{a\})$.


$$\begin{tikzcd}
 & & q_1 & \\
\arrow[r] & q_0 \arrow[ur, "a"] \arrow[dr, "a"'] & & \\
 & & q_2 \arrow[r, bend left, "a"] & q_3 \arrow[l, bend left, "a"]
\end{tikzcd}$$

Example - Tracing Input

$\mathbf{L(M)} = \{a\} \cup \{a^{2k} : k > 0\}$

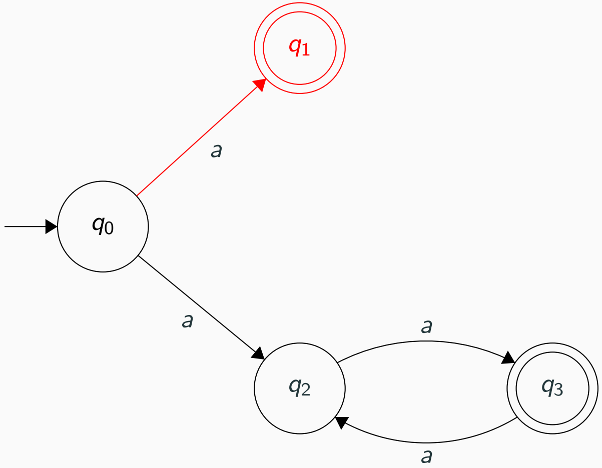
Input String: ^aa



Example - Tracing Input

L(M) $= \{a\} \cup \{a^{2k} : k > 0\}$

Input String: aa

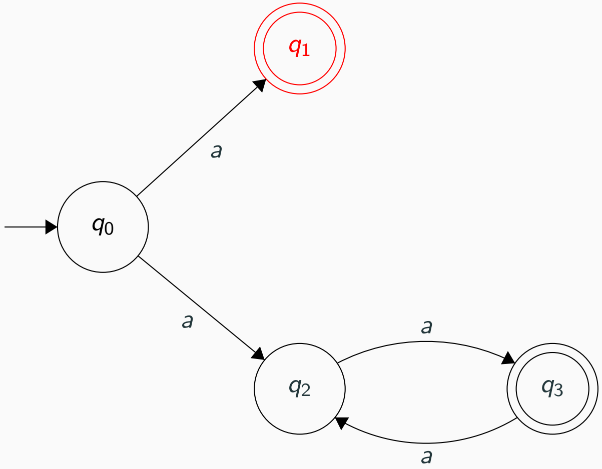


Example - Tracing Input

$\mathbf{L(M)} = \{a\} \cup \{a^{2k} : k > 0\}$

Input String: aa

$\delta(q_1, a) = \emptyset$, no where to go. Are we done?

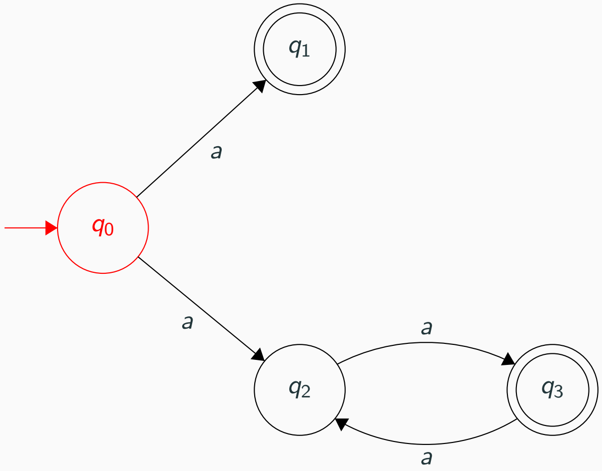


Example - Tracing Input

L(M) $= \{a\} \cup \{a^{2k} : k > 0\}$

Input String: ˆaa

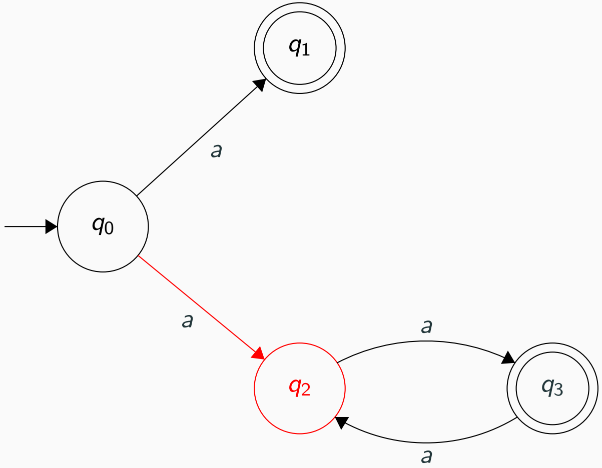
Are we done? No, **M** tries another walk.



Example - Tracing Input

$\mathbf{L(M)} = \{a\} \cup \{a^{2k} : k > 0\}$

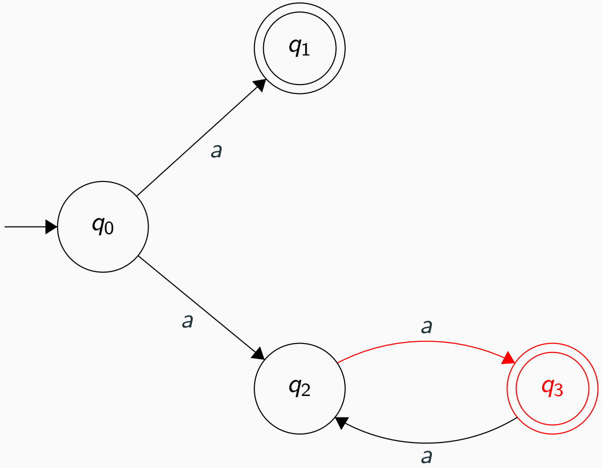
Input String: aa



Example - Tracing Input

$\mathbf{L(M)} = \{a\} \cup \{a^{2k} : k > 0\}$

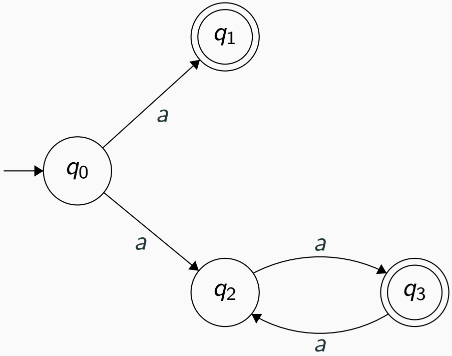
Input String: aa



One of the possible walks ends in a final state, **M** accepts this string.

Example - Tracing Input

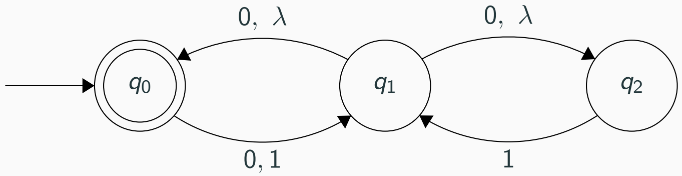
L(M) $= \{a\} \cup \{a^{2k} : k > 0\}$
Input String: aaa



In both traces (top and bottom), do not end up in a final state. **M** rejects this string.

Exercise

Exercise. Given the following NFA **M**,



What is

1. $\delta^*(q_0, 01) =$? Is 01 accepted by this NFA?

NFA-to-DFA

NFA-to-DFA

Given an NFA $N = (Q, \Sigma, \delta, q_0, F)$, how can we convert it to a DFA $M = (Q', \Sigma, \delta', q_0', F')$?

Converting a NFA to DFA is also called **Subset Construction**.

1) Step 1: Start from the initial state S ($S = \{q_0\}$).

2) Step 2: For each symbol $a \in \Sigma$, find all the states that can be reached from S: $\delta'(S, a) = \bigcup_{p \in S} \delta(p, a)$.

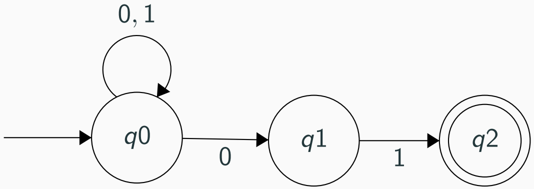
3) Step 3: Repeat Step 2 on every new state that is generated. Repeat until no new states are produced.

4) Step 4: Draw the DFA with states and edges from Step 3.

The initial state for the DFA will be $\{q_0\}$. The final states of the DFA will be all those states S that contain a final state from F. If the original NFA N accepts λ, make $\{q_0\}$ a final state.

Example

Example. Convert the following NFA *M* to a DFA. *M* is the NFA accepting all strings (over $\Sigma = \{0, 1\}$) that end in 01.

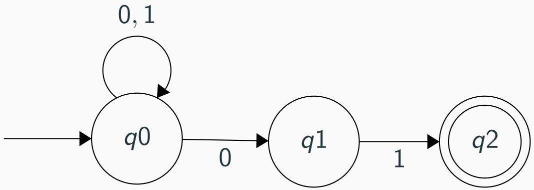


Example

Example. Converting the NFA M to a DFA:

1) Step 1: Start from the start state S. $S = \{q0\}$

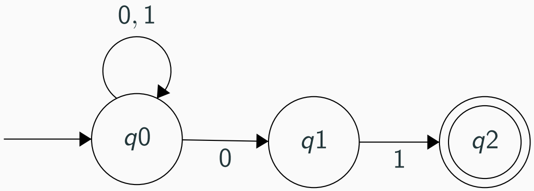
2) Step 2: Find all the states that can be reached from S: $\forall a \in \Sigma$, $\delta'(S, a) = \bigcup_{p \in S} \delta_N(p, a)$. $\delta'(S, 0) = \{q0, q1\}$, $\delta'(S, 1) = \{q0\}$



Example

Example.

3) Step 3: Repeat Step 2 on every new state that is generated. Repeat until no new states are produced.



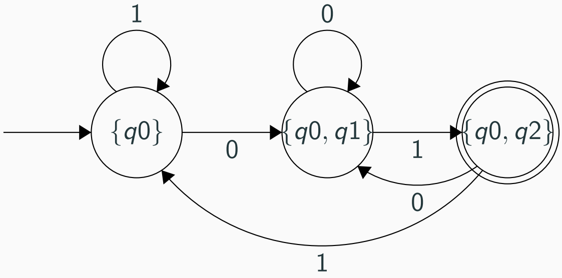
	0	1
→{q0}	{q0, q1}	{q0}
{q0, q1}	{q0, q1}	{q0, q2}
*{q0, q2}	{q0, q1}	{q0}

Example

Example.

4) Step 4: Draw the DFA

	0	1
→{q0}	{q0, q1}	{q0}
{q0, q1}	{q0, q1}	{q0, q2}
*{q0, q2}	{q0, q1}	{q0}



Theorem

Theorem. The set of all languages accepted by NFAs is the same as the set of all languages accepted by DFAs.
Why?

1. Every DFA is an NFA.
2. Every NFA can be converted into a DFA.

Corollary. A language is regular if there is an FA (either DFA or NFA) that accepts it.