# CSC236 Week 11

Larry Zhang

# Announcements

- Next week’s lecture: Final exam review
- This week’s tutorial: Exercises with DFAs
- PS9 will be out later this week’s.

# Recap

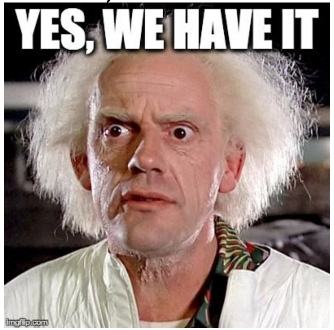
- Last week we learned about **D**eterministic **F**inite **A**utomata.
- It is a model of the computation performed when the computer checks if a string belongs to a regular language.
- Every regular language can be recognized by a DFA with a finite number of states.
- An irregular languages would need an infinite number of states in the DFA, so it cannot be recognized by a computer efficiently.

# Number of DFA States

- Normally, we would expect that the number of states needed by the DFA somehow reflects how complicated it is to describe the regular languages.
- But sometimes, similar languages require quite different numbers of states. For example,
- $(0+1)(0+1)1(0+1)^*$
  - all strings whose third symbol from the **left** is 1.
  - **5 states** needed by the DFA (**Home exercise:** draw the DFA)
- $(0+1)^*1(0+1)(0+1)$
  - all strings whose third symbol from the **right** is 1.
  - **8 states** needed by the DFA (**Home exercise:** draw the DFA)
  - Hint: need to remember the last three symbols read, of which there are 8 possibilities.

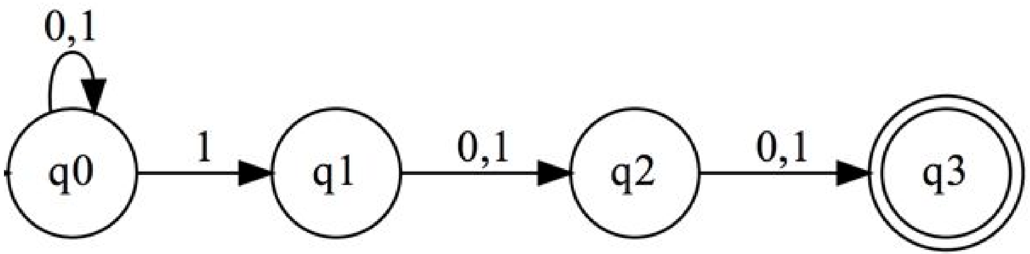
# Why more states?

- (0+1)(0+1)1(0+1)*  vs  (0+1)*1(0+1)(0+1)
- This has to do with how DFA works.
  - Read string **from the left to the right**.
  - It is easier to deal with what's on the left (making decisions with what has happened already)
  - It is harder to deal with what's on the right (making decisions for the **future**!)
  - For the second regex, we don't know when we are at the third-last symbol, so we have to be prepared for everything that could happen in three steps into the future, therefore needing more states.
- It would be nice to have a simpler model for this.



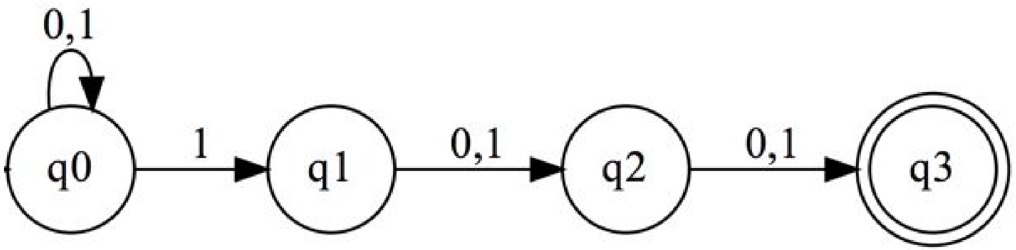
# Nondeterministic Finite Automata
# (**NFA**)

# Here is an NFA. What's different?




- From q0, reading symbol 1 leads to **two possible transitions**!
- q3 has **no outgoing transition** at all!
- Basically, scanning a given string will gives **a set of multiple possible paths** in the diagram, compared to only one path for DFA. (**Non**deterministic)
- For NFAs, we say a string w is accepted if **one of the possible paths** got by scanning the string **reaches the accepting state**.
- The above NFA actually checks if the third-last symbol is 1.

# Scan string 010110




- Start from q0
- Read **0**, reach q0
- Read 0**1**, reach q0, or q1
- Read 01**0**, reach q0, or q2
- Read 010**1**, reach q0, or q1, or **q3** (accepting)
- Read 0101**1**, reach q0, or q1, or q2, or **nowhere (from q3)**
- Read 01011**0**, reach q0, or q2, or **q3** (accepting)

# Formal Definition of NFA

# NFA: Formal Definition

A Nondeterministic Finite Automaton (NFA) $\mathcal{N}$ is similar to a DFA.

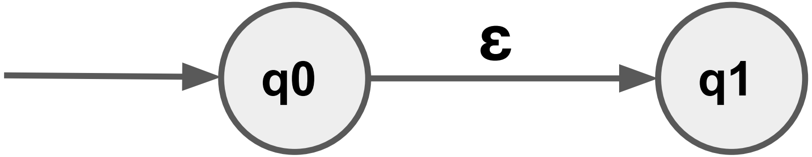
- $Q, \Sigma, s, F$: same as before
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the *transition function* representing the state transitions
  - Given a state and a symbol, it returns the **set of states** to which that symbol transitions

- Recall that a DFA accepts string $w$ iff $\delta(s, w) \in F$
- An NFA accepts string $w$ iff $\delta(s, w) \cap F \neq \emptyset$
  - i.e. $w$ is accepted iff at least one of its paths terminates in a final state

# one more thing ...

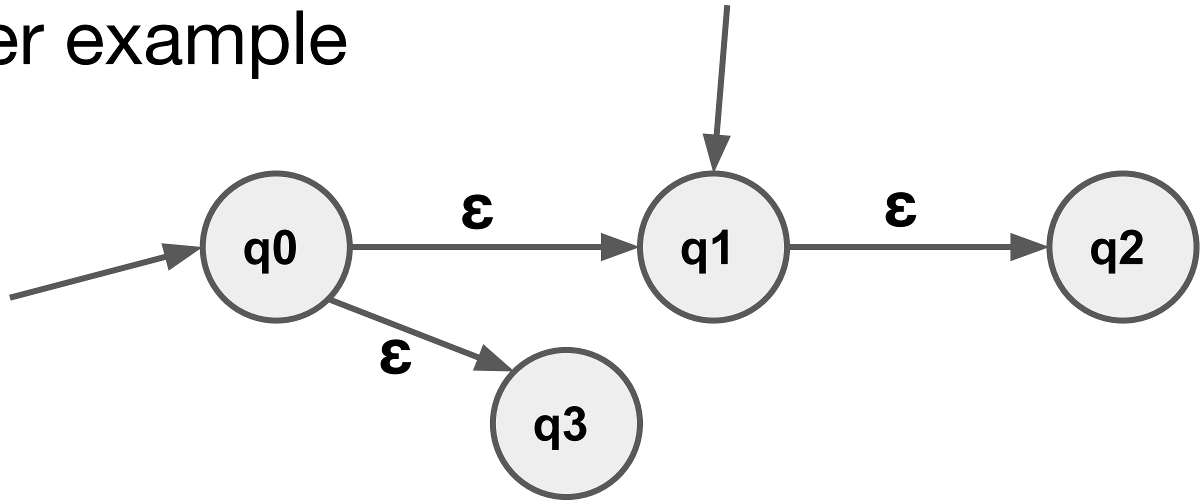

## ε-transition

Transition from one state to another without reading a symbol.




- If you are reaching **q0**, then you are reaching **q1** for free (without needing to read a symbol)!
- In other words, reaching state **q0** is equivalent to reaching the **set of states {q0, q1}**

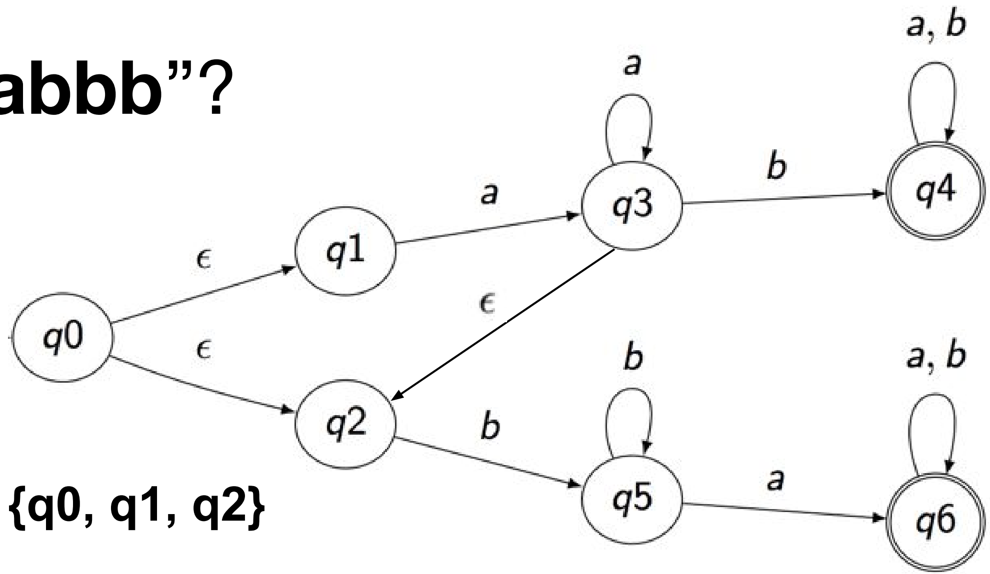
# another example




- Reaching **q0** is equivalent to reaching the set **{q0, q1, q2, q3}**.
- Reaching **q1** is equivalent to reaching the set **{q1, q2}**.
    - Note: no free ε-transition from q1 back to q0

# Exercise

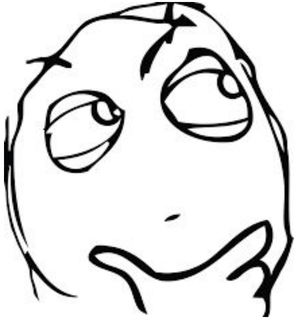
# Does this NFA accept “**abbb**”?




- Initial state **s = q0**
- Actual initial set of states **δ(s, ε) = {q0, q1, q2}**
- **δ(s, a) = {q3, q2}**   # ε-transition q3 to q2
- **δ(s, ab) = {q4, q5}**
- **δ(s, abb) = {q4, q5}**
- **δ(s, abbb) = {q4, q5}**
  - accepting because q4 is an accepting state

So, Regex, DFA, NFA, they all recognize regular languages.

## How are they related to each other?



# Equivalence of Representations

Let ***L*** be a languages over an alphabet ∑. Then the following are **equivalent.**

- There is a regular expression that matches ***L*.**
- There is a DFA that accepts ***L***
- There is an NFA with ε-transitions that accepts ***L***
- There is an NFA without ε-transitions that accepts ***L***

# In other words

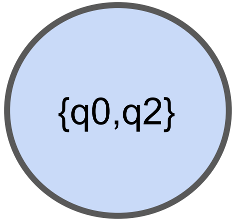
- Given a regex, we can construct a DFA that accepts the same language
- Given a regex, we can construct an NFA that accepts the same language
- Given a DFA, we can construct a regex that accepts the same language
- Given a DFA, we can construct an NFA that accepts the same language
- Given an NFA, we can construct a regex that accepts the same language
- Given an NFA we can construct a DFA that accepts the same language

**We now show how this is done.**

# Subset Construction

an algorithm that converts NFA to DFA

# Subset Construction




- Each state **Q** in the DFA represents **a set of states** in the NFA

The algorithm:

- Let Q0 = δ(s, ε), i.e., the set of states that reachable from initial NFA state **s** via zero or more ε-transitions
- Repeat until no more new DFA states are added
    - For each state Q in the DFA, for each symbol x, determine the set R of NFAs that are reached from any NFA state in Q via symbol x.
    - For each R from the last step, **update R = δ(R, ε)**, which is the set of all NFA states reachable from some state in R via zero or more ε-transitions.

# The initial state and accepting states of the DFA

Initial state of the DFA:

- $Q0 = \delta(s, \varepsilon)$

Accepting states of the DFA:

- Any states that contains an accepting state of the NFA.

# Example

# Find the equivalent DFA for the following 4-state NFA

| old state | symbol | new state |
|---|---|---|
| q0 | 0 | q1 |
| q0 | 1 | q0 |
| q0 | ε | q3 |
| q1 | 0 | q2 |
| q2 | ε | q1 |
| q2 | 1 | q3 |

**The initial state is q0; the accepting state is q3.**

| old state | symbol | new state |
| --- | --- | --- |
| q0 | 0 | q1 |
| q0 | 1 | q0 |
| q0 | ε | q3 |
| q1 | 0 | q2 |
| q2 | ε | q1 |
| q2 | 1 | q3 |

$Q_0 : \{q_0\} \xrightarrow{\epsilon} \{q_0, q_3\}$ (initial state of the DFA)

$\{q_0, q_3\} \xrightarrow{0} \{q_1\}$ (new state!)

$\{q_0, q_3\} \xrightarrow{1} \{q_0\} \xrightarrow{\epsilon} \{q_0, q_3\}$

$\{q_1\} \xrightarrow{0} \{q_2\} \xrightarrow{\epsilon} \{q_1, q_2\}$ (new state!)

$\{q_1\} \xrightarrow{1} \emptyset$ (new state!)

$\{q_1, q_2\} \xrightarrow{0} \{q_2\} \xrightarrow{\epsilon} \{q_1, q_2\}$

$\{q_1, q_2\} \xrightarrow{1} \{q_3\}$ (new state!)

$\{q_3\} \xrightarrow{0} \emptyset$

$\{q_3\} \xrightarrow{1} \emptyset$

No more new states. Done.

How many states in the DFA?

- 5
- {q0, q3}, {q1}, {q1, q2}, {q3}, ∅

Which state is the initial state?

- Q0: {q0, q3}

Which states are accepting states?

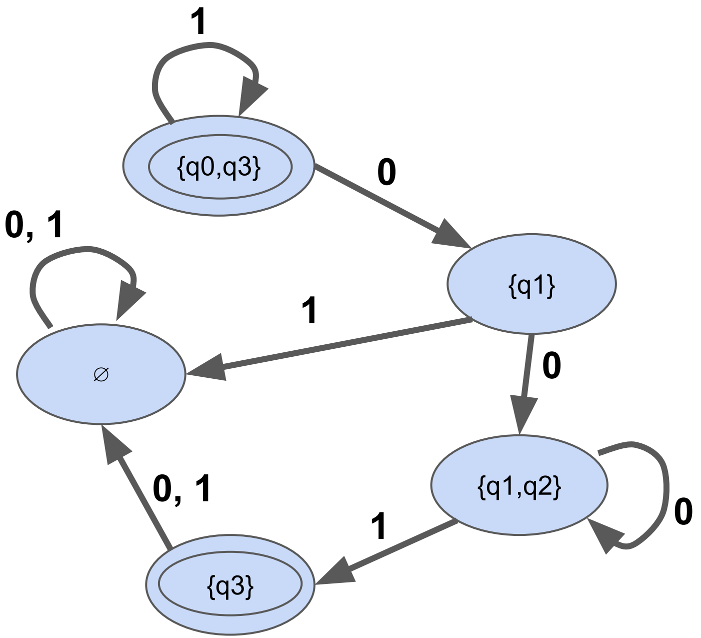
- any state that contains q3
- {q0, q3} and {q3}.

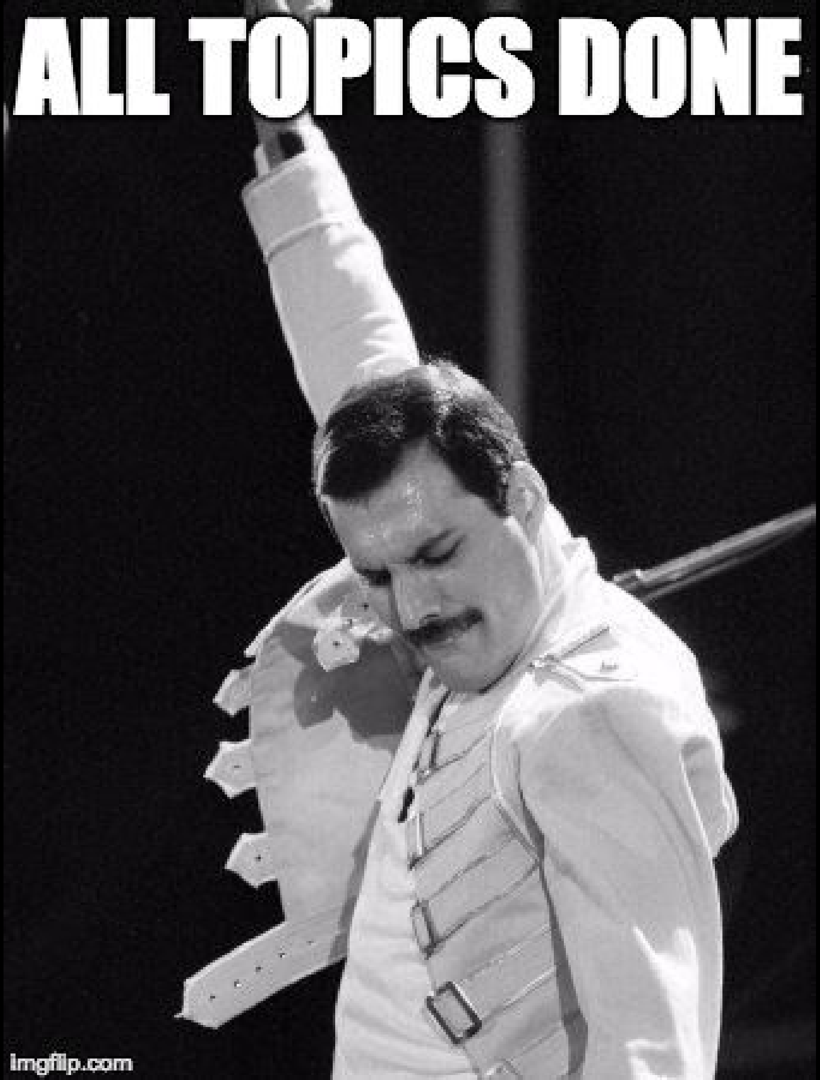
# The resulting DFA from subset construction

| old state | symbol | new state |
|---|---|---|
| {q0, q3} | 0 | {q1} |
| {q0, q3} | 1 | {q0, q3} |
| {q1} | 0 | {q1, q2} |
| {q1} | 1 | ∅ |
| {q1, q2} | 0 | {q1, q2} |
| {q1, q2} | 1 | {q3} |
| {q3} | 0 | ∅ |
| {q3} | 1 | ∅ |
| ∅ | 0 | ∅ |
| ∅ | 1 | ∅ |

Initial state: {q0, q3}

Accepting states: {q0, q3}, {q3}




ALL TOPICS DONE
imgflip.com