Theory of Computation

BCS1110

Dr. Ashish Sai

🗓️ TOC - Lecture 2

💻 bcs1110.ashish.nl

Plan for Today

- Recap from TOC Lecture 1
- Tabular DFAs
- Regular Languages
- NFAs
- Designing NFAs
- (*if time permits*) Tutorial Questions

Recap From Last Time

Old MacDonald Had a Symbol, 🎶 Σ-eye-ε-ey∈, Oh! 🎶

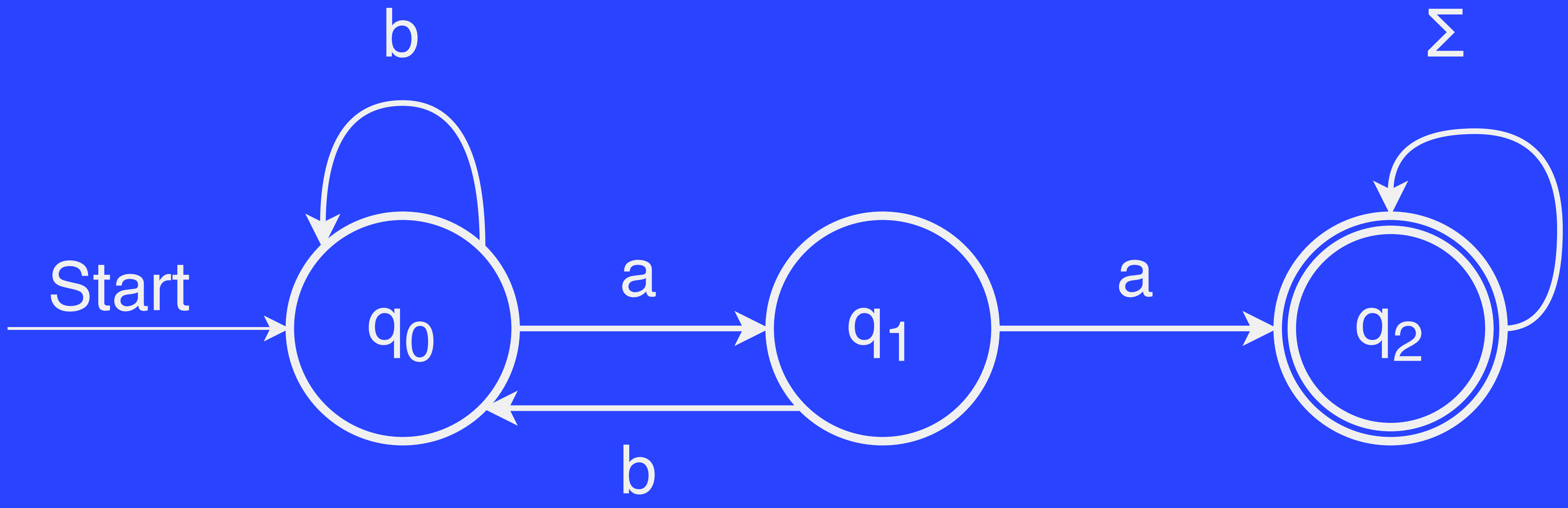
- Here's a quick guide to remembering which is which"
  - Typically, we use the symbol **Σ** (sigma) to refer to an *alphabet*
  - The *empty string* is length 0 and is denoted **ε** (epsilon)
  - In set theory, use **∈** to say "is an *element of*"
  - In set theory, use **⊆** to say "is a *subset of*"

DFAs

- A **DFA** is a
  - **D**eterministic
  - **F**inite
  - **A**utomaton

Recognizing Languages with DFAs

L = { w ∈ {a,b} * | w contains aa as a substring }



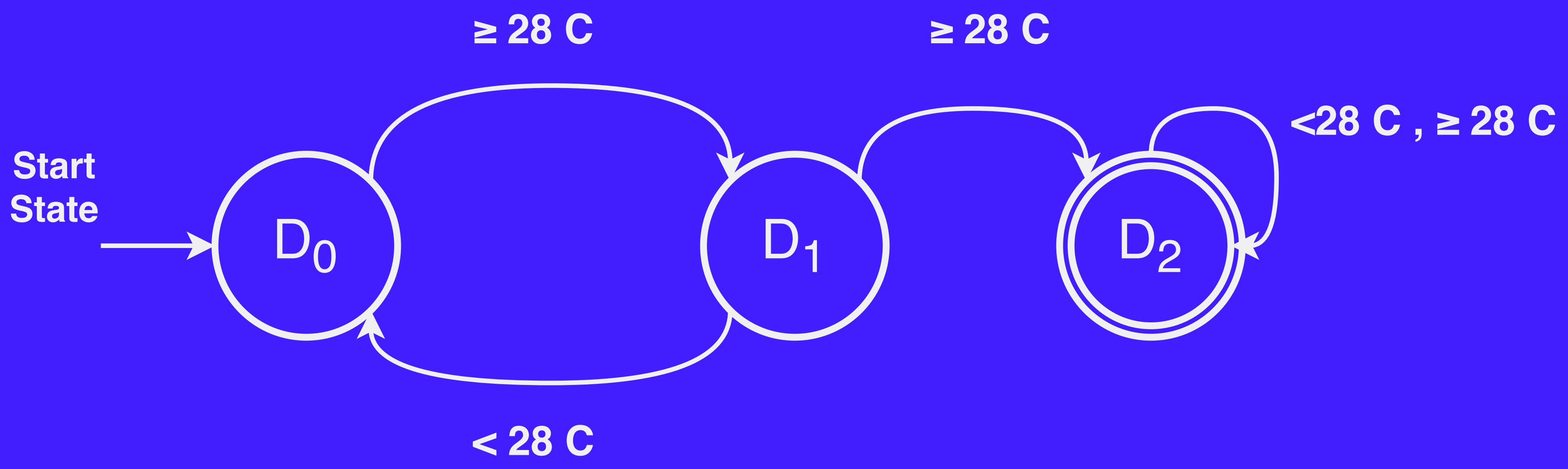
DFAs

- A DFA is defined relative to some alphabet Σ (sigma)
- For each state in the DFA, there must be **exactly one** transition defined for each symbol in Σ
  - This is the "deterministic" part of DFA
- There is a unique start state
- There are zero or more accepting states

Tabular DFAs

Part 1/4

Deterministic Finite Automaton (Formal Definition)



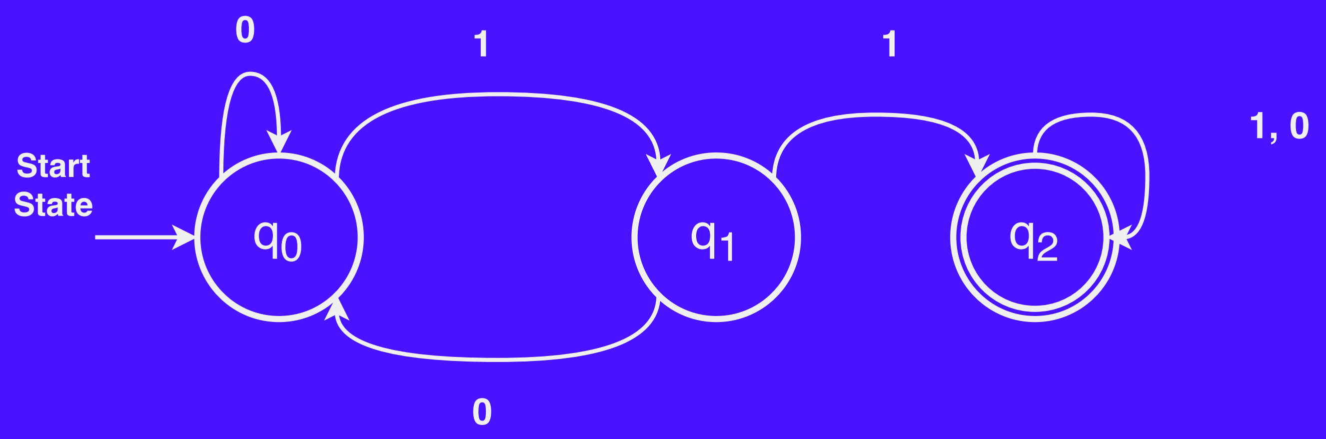
- Input: String of weather data
- 🇬🇧 Heatwave: temperature ≥ 28 C for 2 consecutive days

DFA Definition

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

- Q is the set of states [$Q = \{ q_0, q_1, q_2 \}$]
- Σ is the alphabet [$\Sigma = \{1,0\}$]
- δ is the transition function
- q_0 is the start state
- F is an accepting state [$F = \{ q_3 \}$]

Transition Function



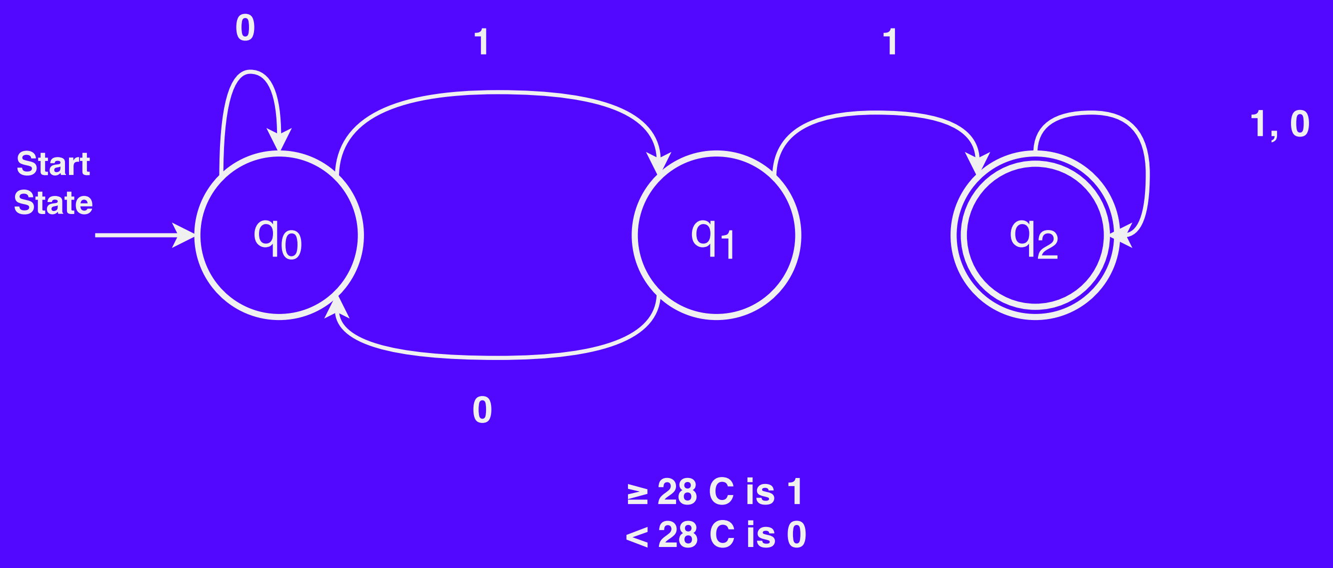
≥ 28 C is 1
< 28 C is 0

DFA Definition

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

- Q is the set of states [Q = { q_0, q_1 , q_2 }]
- Σ is the alphabet [Σ = {1,0}]
- δ is the transition function
- q_0 is the start state
- F is an accepting state [F = { q_3 }]

Transition Function



	1	0
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_2	q_2

Which table best represents the transitions for the following DFA?

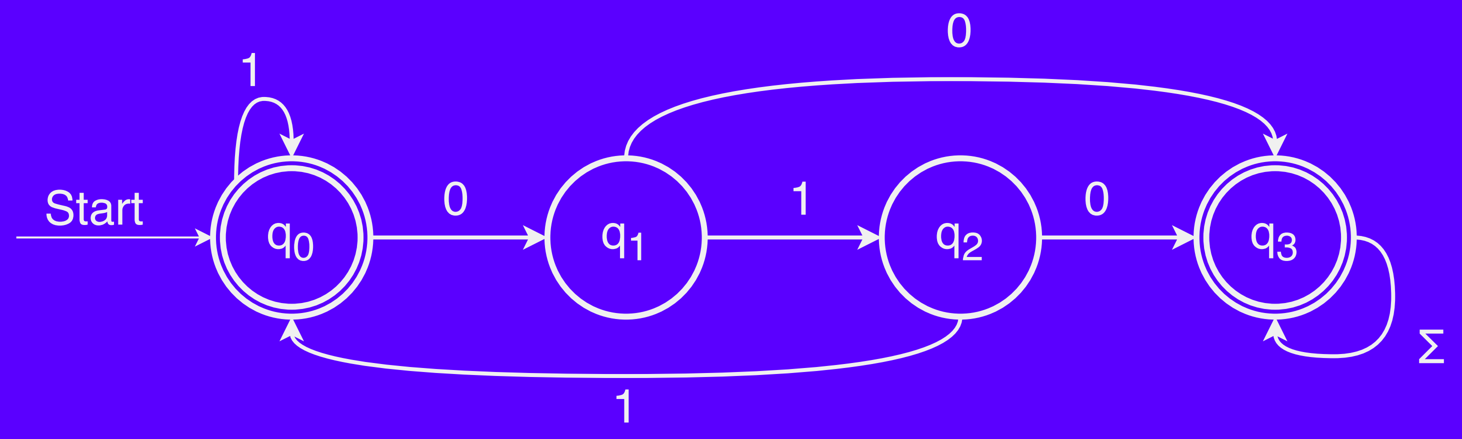


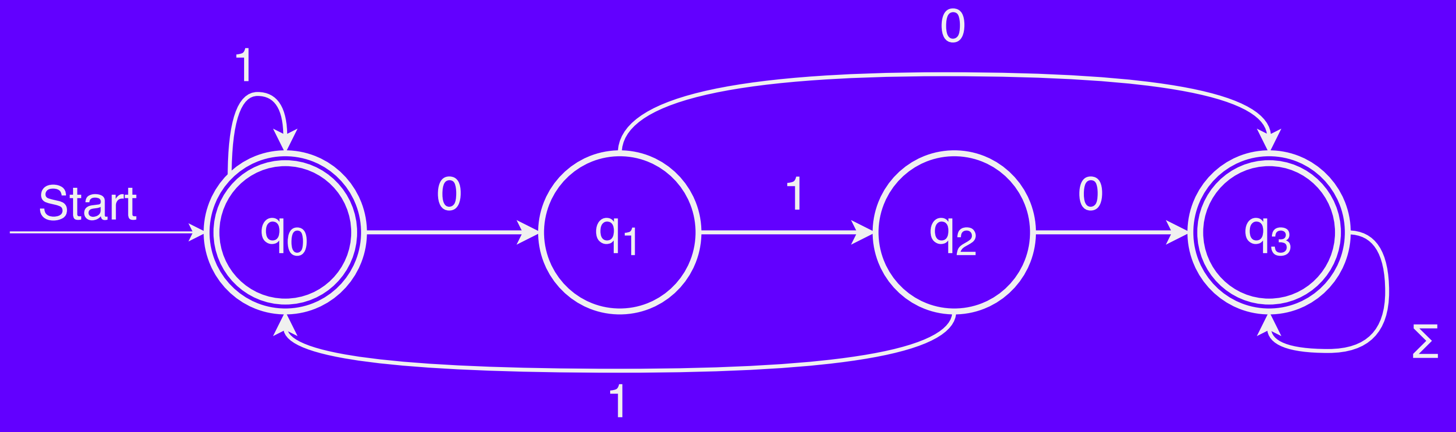
Table A

	0	1
q_0	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
q_3	q_3	q_3

Table B

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_3
q_3	q_3	q_3

Tabular DFAs

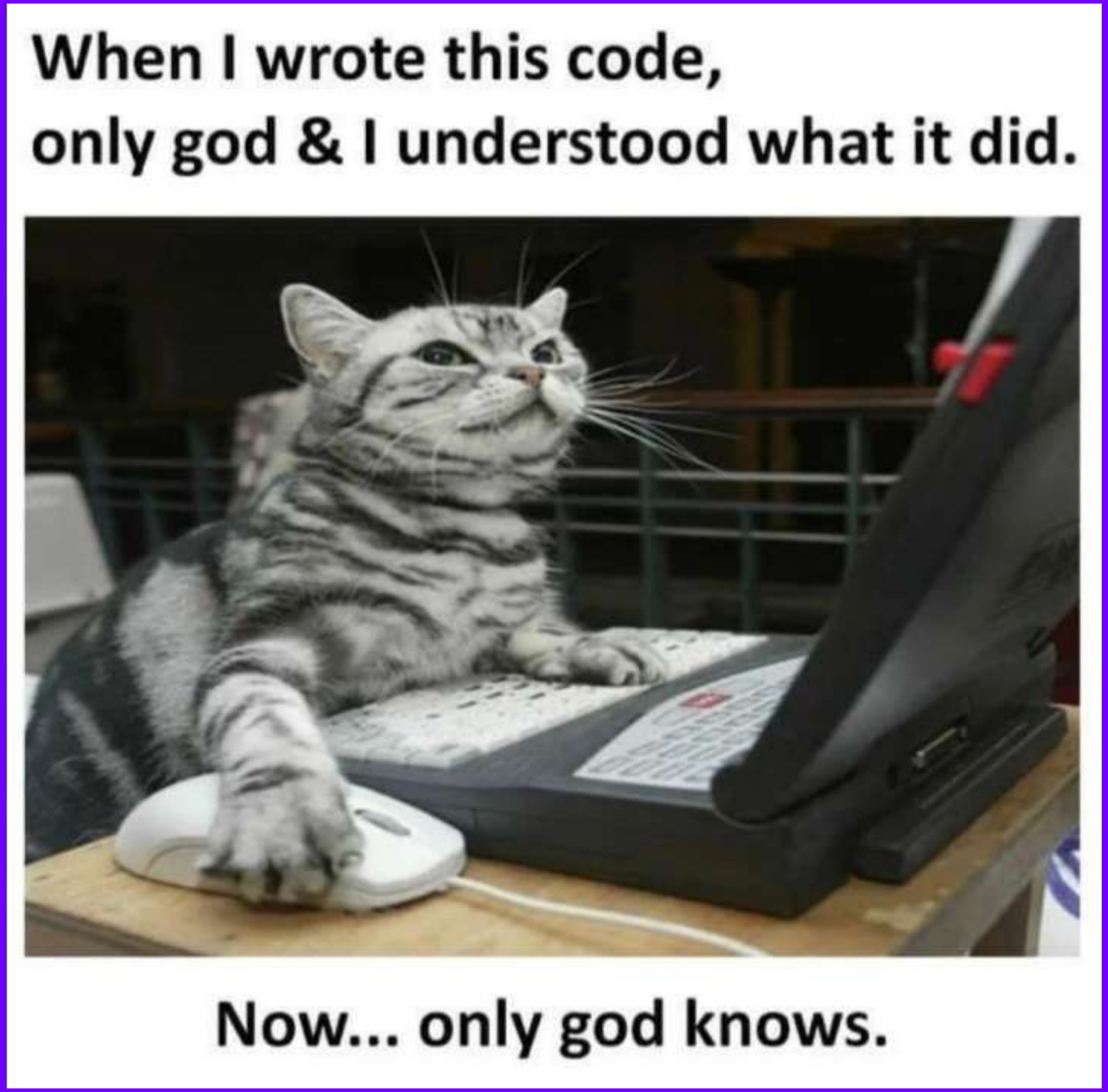


	0	1
$^*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$^*q_3$	q_3	q_3

— These starts indicate accepting states

— First row is the start state

Code Demo



Thanks Joris!

```java
public class DFASimulator {

    private static final int kNumStates = 4; // 4 states based on the table
    private static final int kNumSymbols = 2; // 2 symbols (0 and 1) based on the table

    private static final int[][] kTransitionTable = {
        {1, 0},
        {3, 2},
        {3, 0},
        {3, 3}
    };

    private static final boolean[] kAcceptTable = {
        true,
        false,
        false,
        true
    };

    public static boolean simulateDFA(String input) {
        int state = 0;
        char[] inputArray = input.toCharArray();
        for (int i = 0; i < inputArray.length; i++) {
            char ch = inputArray[i];
            if (ch != '0' && ch != '1') {
                throw new IllegalArgumentException("Invalid input symbol: " + ch);
            }
            state = kTransitionTable[state][ch - '0'];
        }
        return kAcceptTable[state];
    }

    public static void main(String[] args) {
        String testInput = "1011"; // Example input
        boolean isAccepted = simulateDFA(testInput);
        System.out.println("The input " + testInput + " is " + (isAccepted ? "accepted" : "rejected") + " by the DFA.");
    }
}
```

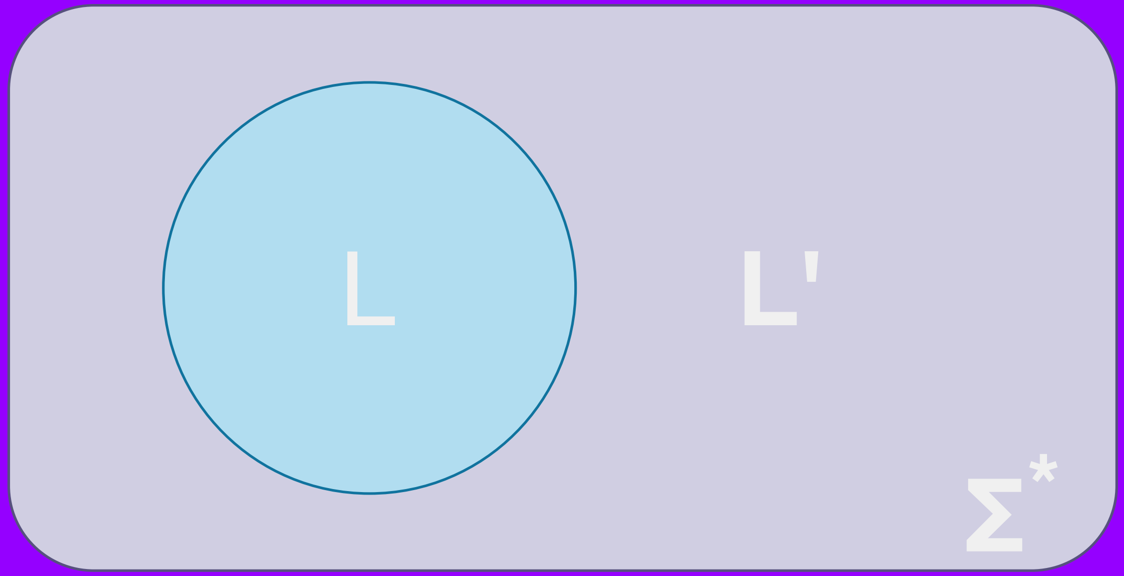
The Regular Languages

Part 2/4

- A language L is called a **regular language** if there exists a *DFA D* such that L(D)=L
- If L is a language and $L(D)=L$, we say that D **recognises** the language L

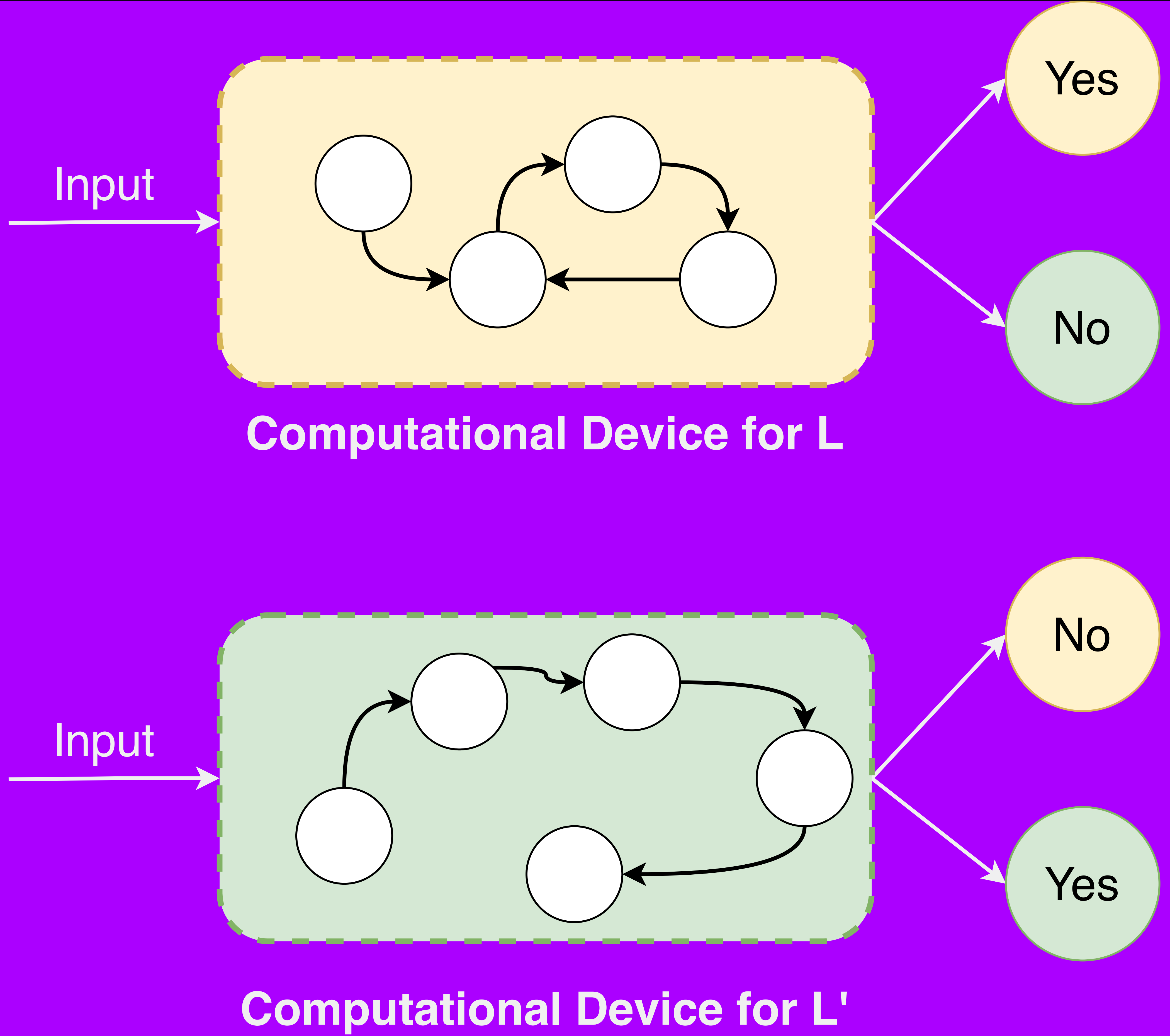
The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the **complement** of that language (denoted **L′**) is the language of all strings in Σ^* that aren't in L
- Formally: $L' = \Sigma^* - L$



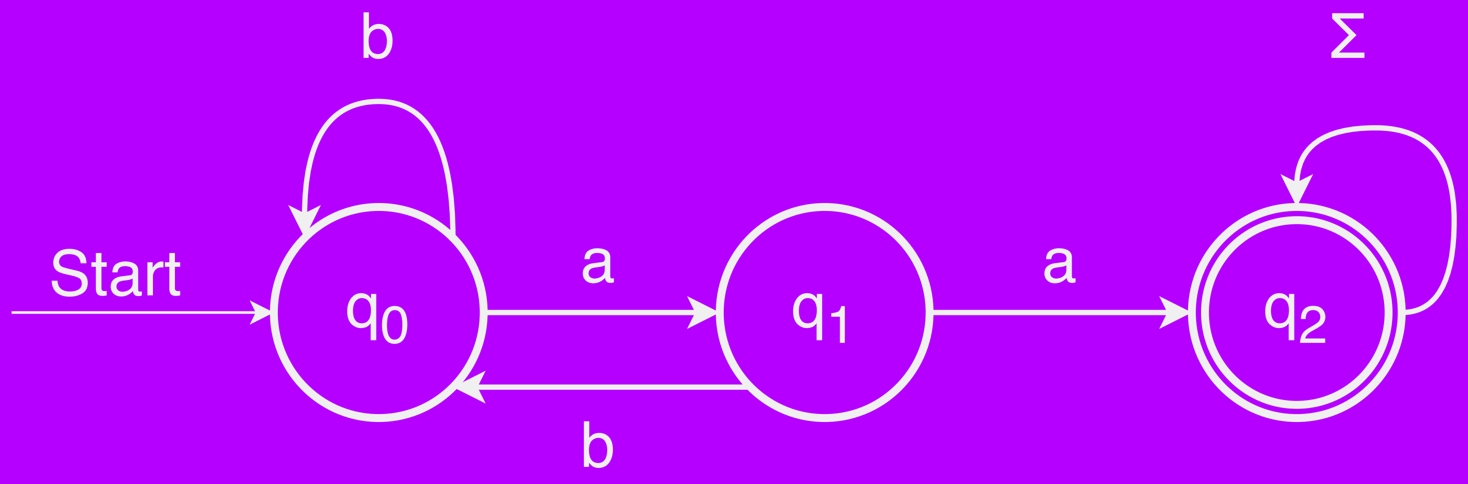
Complements of Regular Languages

- As we saw a few minutes ago, a **regular language** is a language accepted by some DFA
- **Question:** If L is a regular language, is *L′* necessarily a regular language?
- If the answer is "yes," then if there is a way to construct a DFA for L, there must be some way to construct a DFA for *L′*
- If the answer is "no," then some language L can be accepted by some DFA, but *L′* cannot be accepted by any DFA

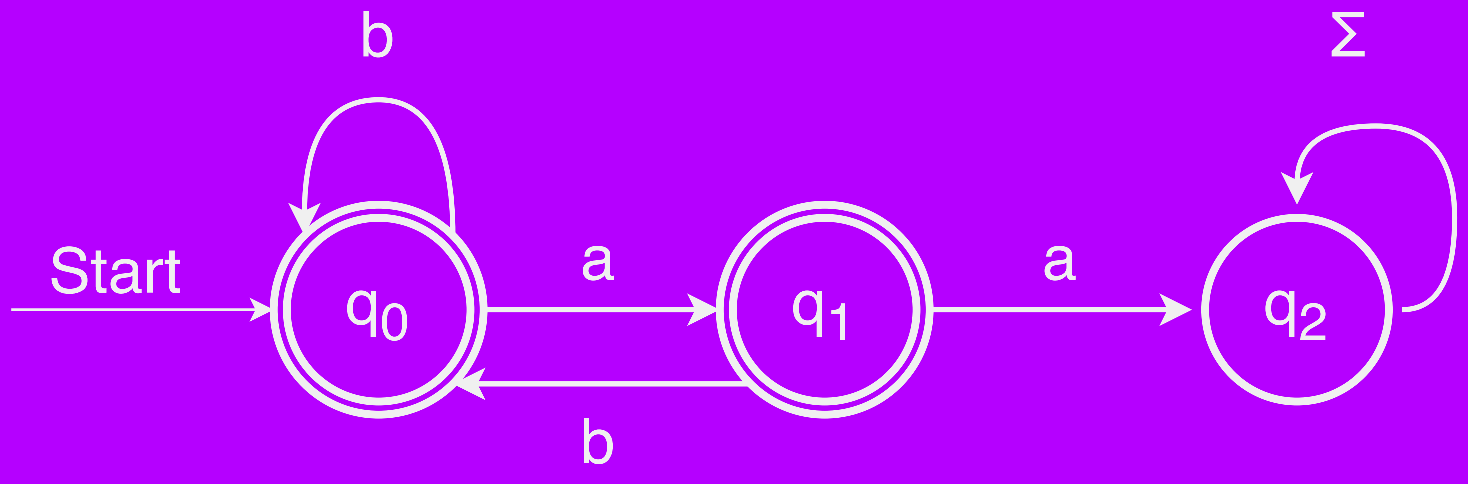

Input
Yes
No
Computational Device for L
Input
No
Yes
Computational Device for L'

Complementing Regular Languages

L = { w ∈ {a,b}* | w contains aa as a substring }

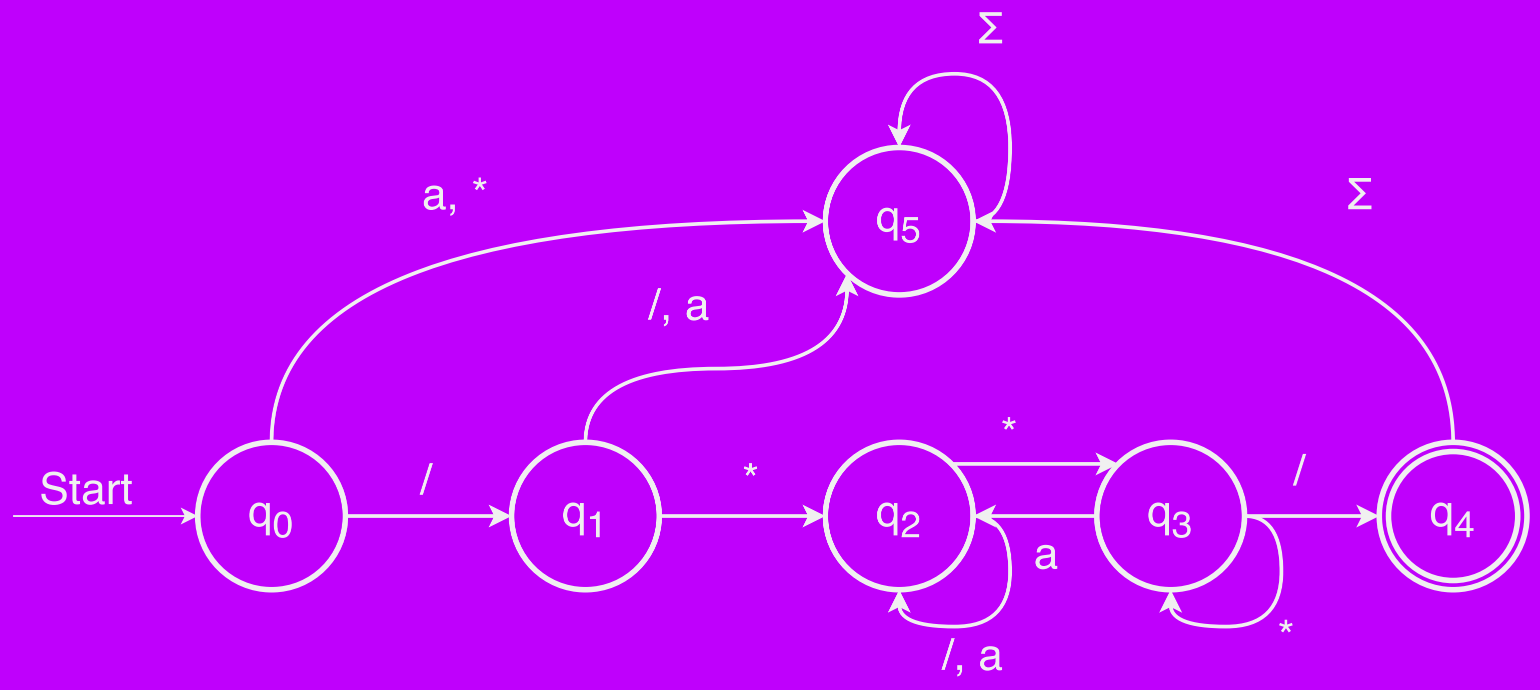


L′ = { w ∈ {a,b}* | w **does not** contain aa as a substring}



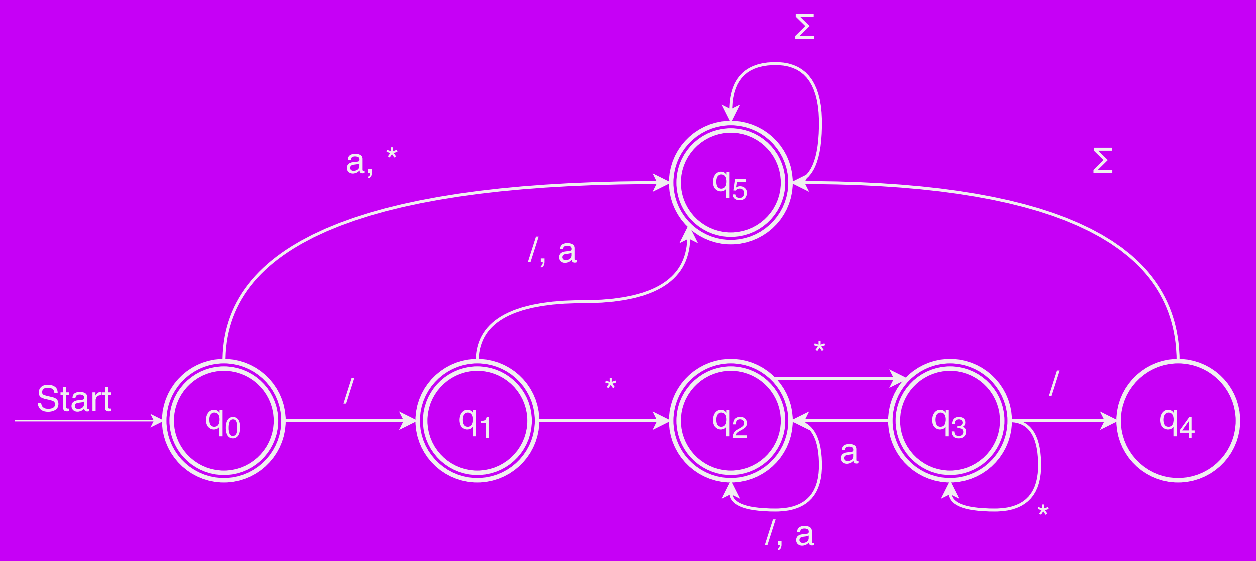
More Elaborate DFAs

L = { w ∈ {a,*,/} | w represents a **(multi-line)** Java-style comment }



More Elaborate DFAs

L′ = { w ∈ {a,*,/} | w doesn't represents a (**multi-line**) Java-style comment }



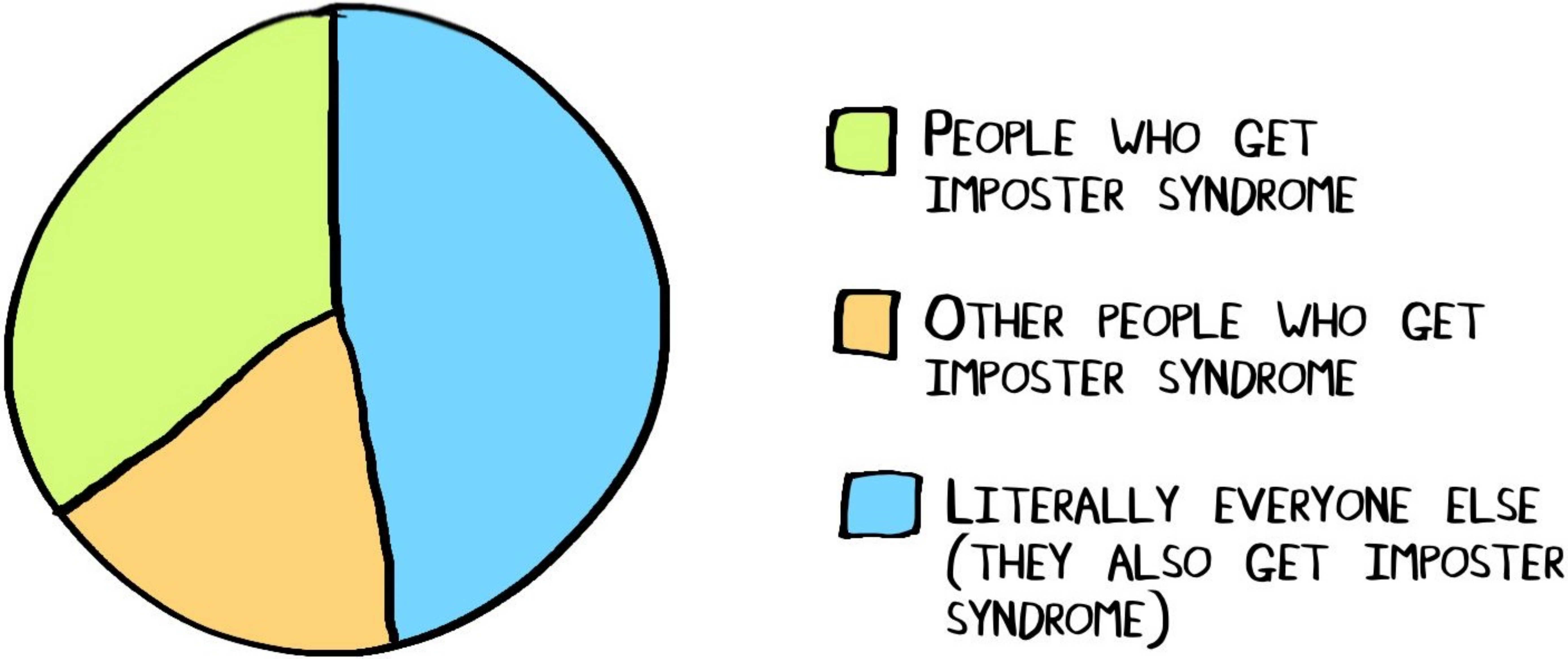
Closure Properties

- **Theorem:** If L is a regular language, then L' is also a regular language
- As a result, we say that the regular languages are **closed under complementation**

Time Out

(Not A Break)

Ever felt you weren't good enough to be in STEM? Afraid of being "found out" because you don't think you belong?



EVERYONE FEELS LIKE AN IMPOSTER SOMETIMES, AND THAT'S OKAY

NFAs

Part 3/4

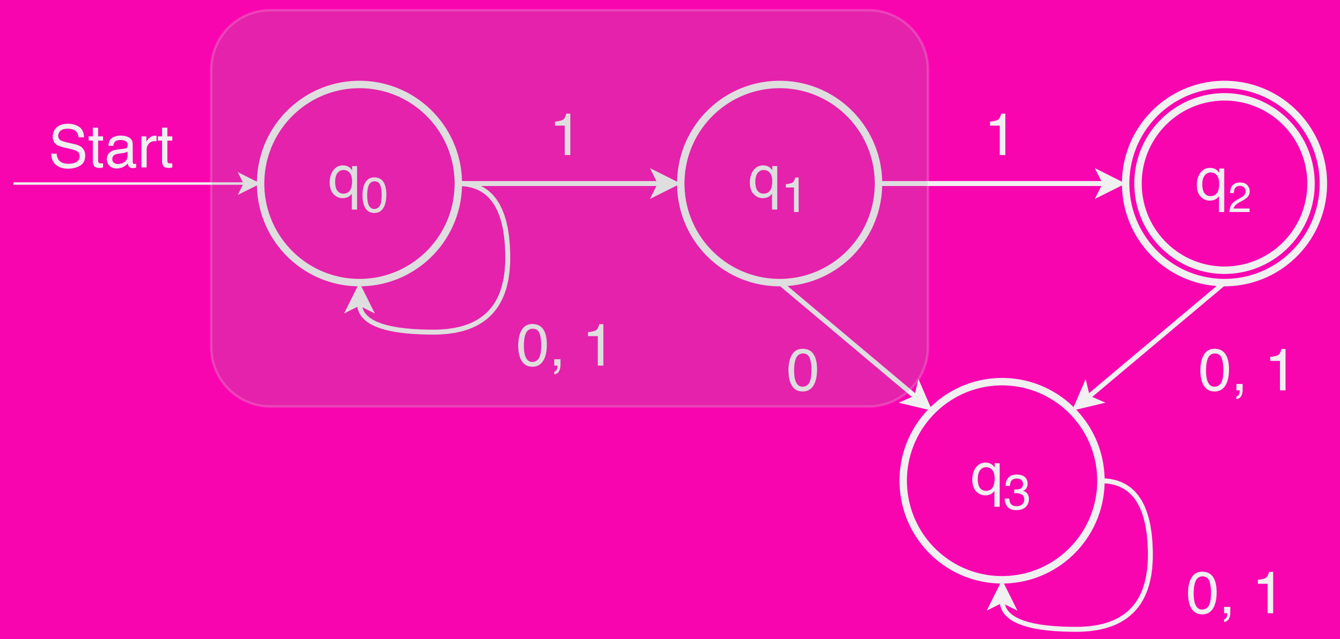
NFAs

- An **NFA** is a
  - **N**ondeterministic
  - **F**inite
  - **A**utomaton
- Structurally similar to a DFA, but represents a fundamental shift in how we'll think about computation

(Non)determinism

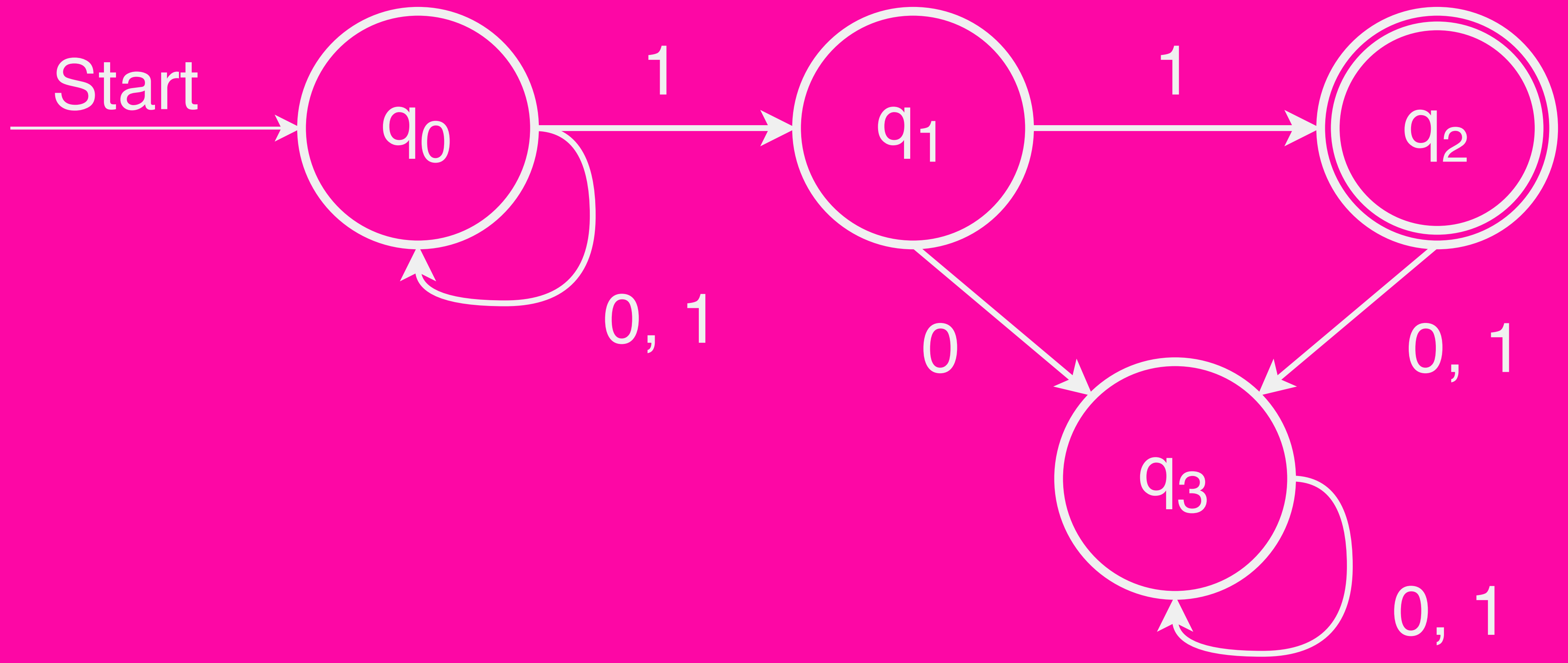
- A model of computation is **deterministic** if at every point in the computation, there is exactly one choice that can make
- The machine accepts if that series of choices leads to an accepting state
- A model of computation is **nondeterministic** if the computing machine may have multiple decisions that it can make at one point
- The machine accepts if any series of choices leads to an accepting state
  - (This sort of nondeterminism is technically called existential nondeterminism, the most philosophical-sounding term we'll introduce)

A Simple NFA



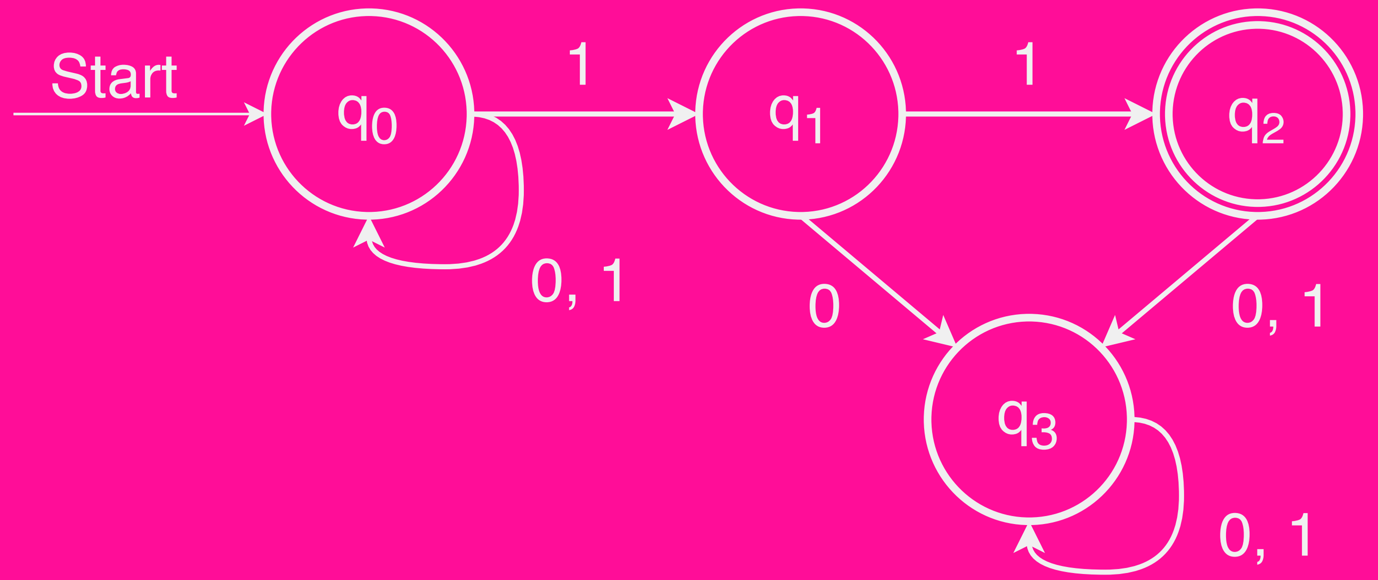
> q_0 has two transitions defined on 1!

A Simple NFA



Input: 01011

Non-Deterministic Finite Automaton (Formal Definition)



D = (Q, Σ, δ, q_0, F)

– Q is the set of states [Q = { q_0, q_1 , q_2, q_3 }]

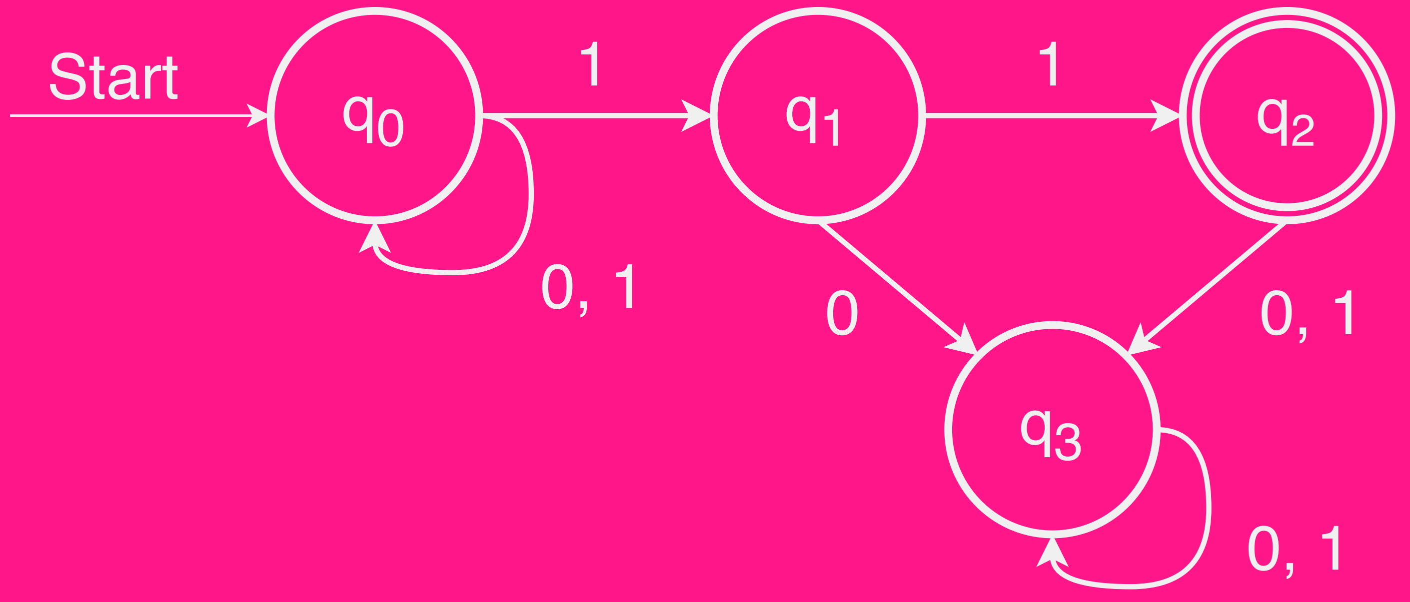
– Σ is the alphabet [Σ = {1,0}]

– δ is the transition function [Same table as DFA, see the next slide]

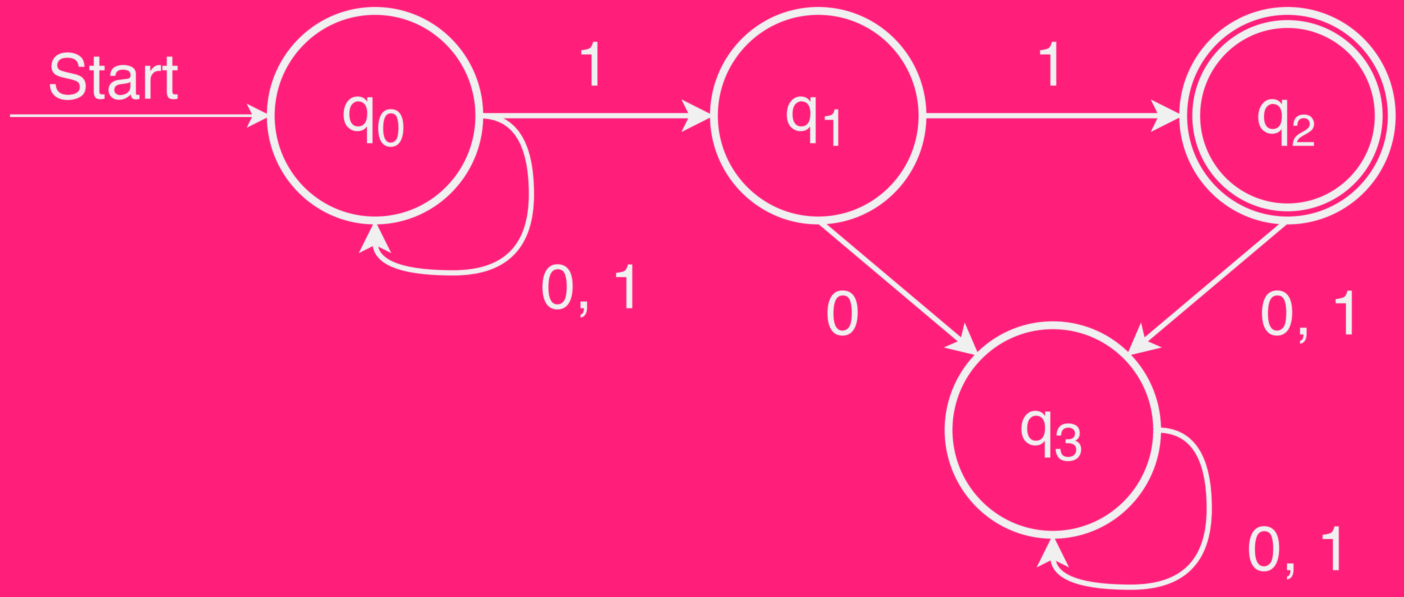
– q_0 is the start state

– F is an accepting state [F = { q_2 }]

A Simple NFA: Transaction Function

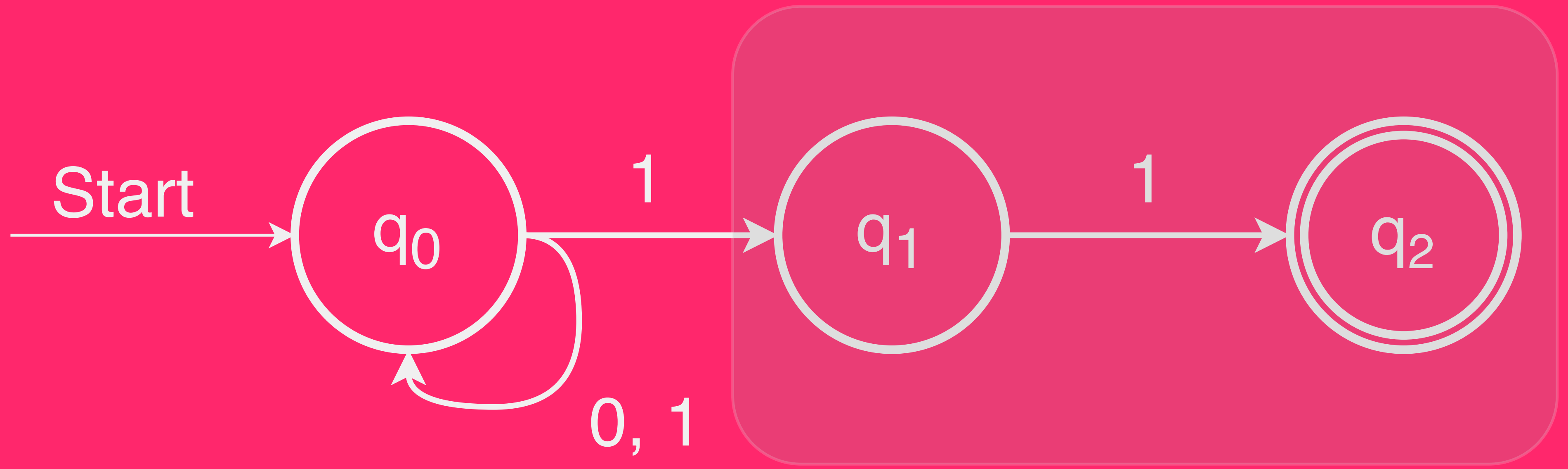


A Simple NFA: Transaction Function

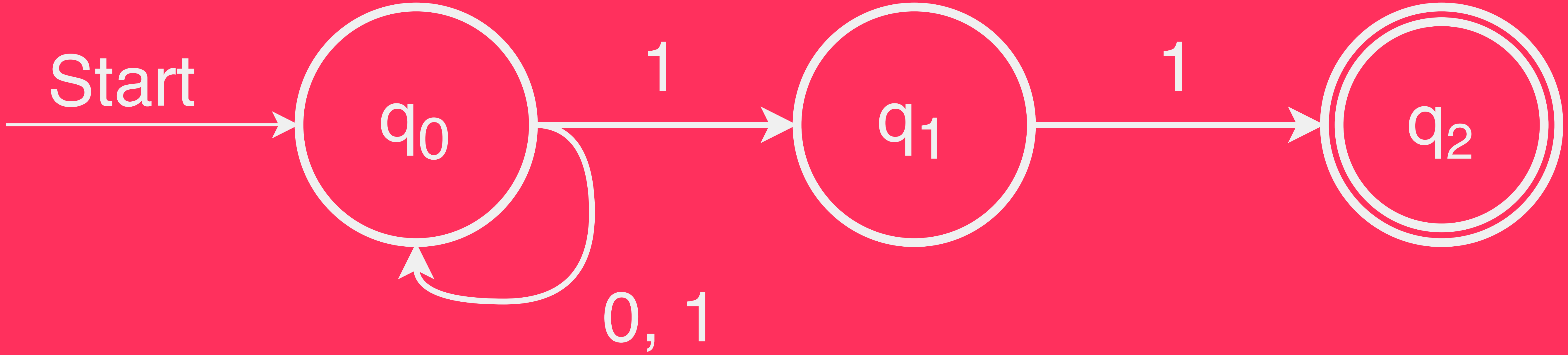


State	0	1
q_0	$\{\ q_0\ \}$	$\{\ q_0,\ q_1\ \}$
q_1	$\{\ q_3\ \}$	$\{\ q_2\ \}$
q_2	$\{\ q_3\ \}$	$\{\ q_3\ \}$
q_3	$\{\ q_3\ \}$	$\{\ q_3\ \}$

A More Complex NFA



If a NFA needs to make a transition when no transition exists, the automaton dies and that particular path does not accept



As with DFAs, the language of an NFA N is the set of strings N accepts:

$L(N) = \{ w \in \Sigma^* \mid N \text{ accepts } w \}$

What is the language of the NFA shown above?

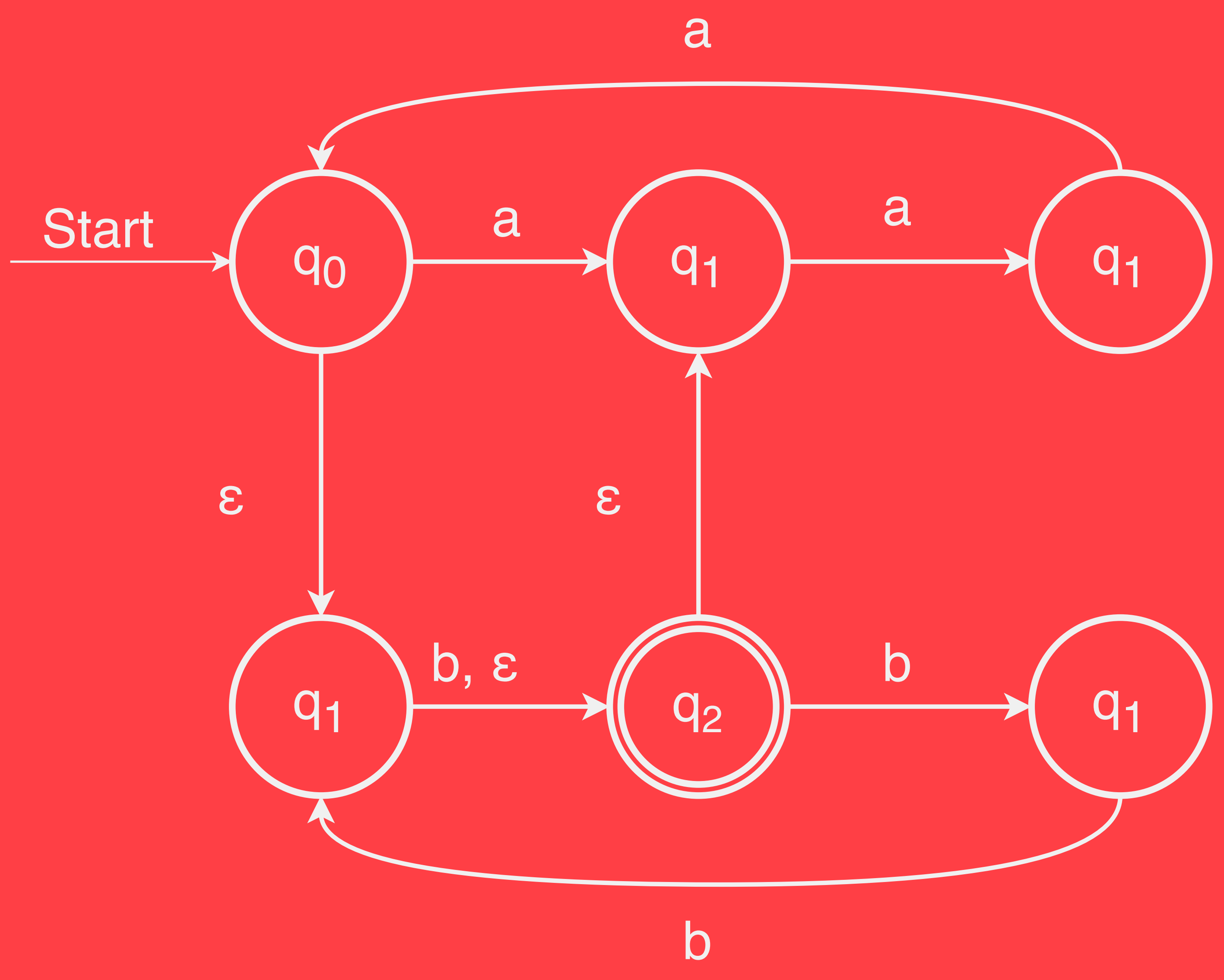
- A) {01011}
- B) $\{ w \in \{0,1\}^* \mid w \text{ contains at least two 1s} \}$
- C) $\{ w \in \{0, 1\}^* \mid w \text{ ends with 11} \}$
- D) $\{ w \in \{0, 1\}^* \mid w \text{ ends with 1} \}$
- E) None of these, or two or more of these

NFA Acceptance

- An NFA *N* accepts a string *w* if there is some series of choices that lead to an accepting state
- Consequently, an NFA *N* rejects a string w if no possible series of choices lead it into an accepting state
- It's easier to show that an NFA does accept something than to show that it doesn't

ε-Transitions

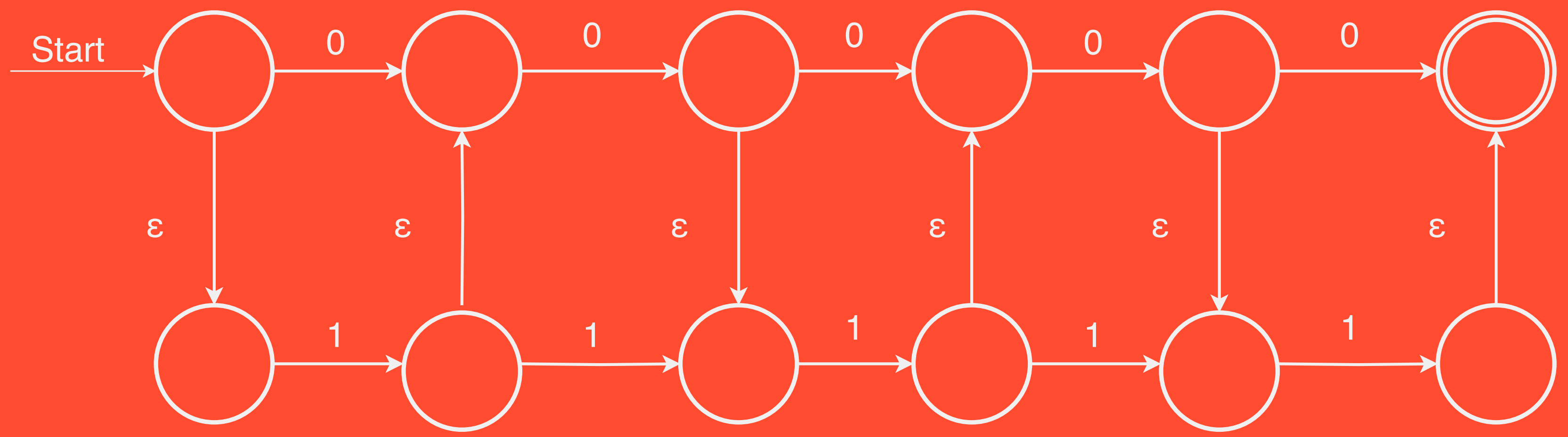
- NFAs have a special type of transition called the **ε-transition**
- An NFA may follow any number of ε-transitions at any time without consuming any input



Input: b a a b b

ε-Transitions

NFAs are **not required** to follow ε–transitions. It's simply another option at the machine's disposal



Suppose we run the above NFA on the string 10110. How many of the following statements are true?

- There is at least one computation that finishes in an accepting state
- There is at least one computation that finishes in a rejecting state
- There is at least one computation that dies
- This NFA accepts 10110
- This NFA rejects 10110

Designing NFAs

Part 4/4

Designing NFAs

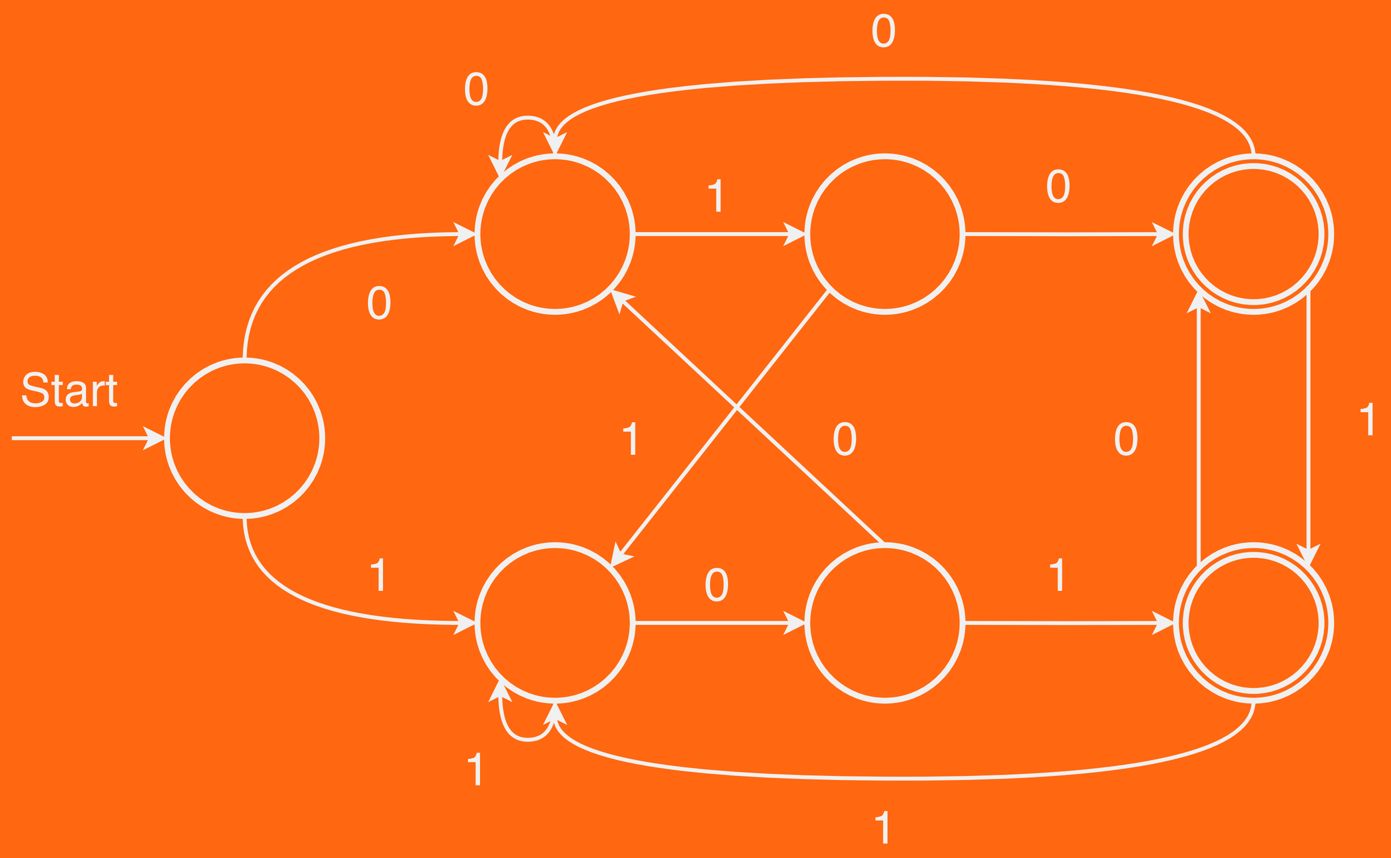
- When designing NFAs, embrace the nondeterminism!
- Good model: **Guess-and-check:**
  - Is there some information that you'd really like to have? Have the machine nondeterministically guess that information
  - Then, have the machine deterministically check that the choice was correct
- The *guess* phase corresponds to trying lots of different options
- The *check* phase corresponds to fltering out bad guesses or wrong options

Guess-and-Check

$L = \{ w \in \{0,1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$

Guess-and-Check

$L = \{ w \in \{0,1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



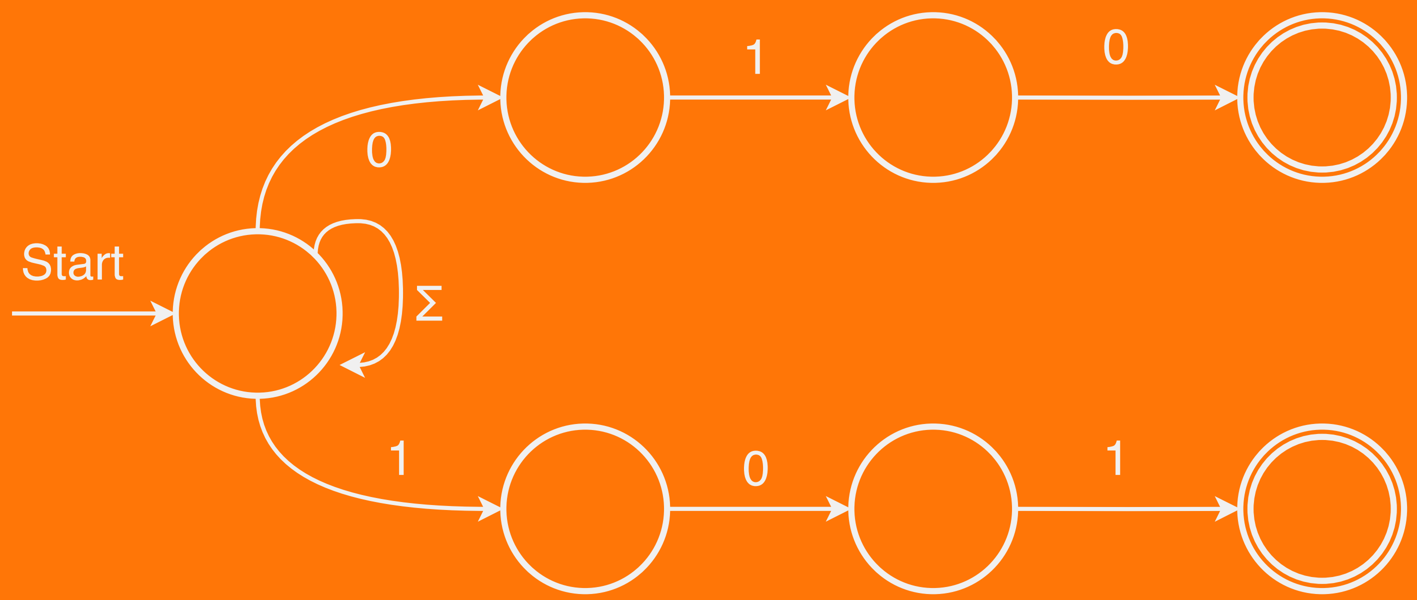
Guess-and-Check

$L = \{ w \in \{0,1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$

- Nondeterministically guess when to leave the start state
- Deterministically check whether that was the right time to do so

Guess-and-Check

$L = \{ w \in \{0,1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



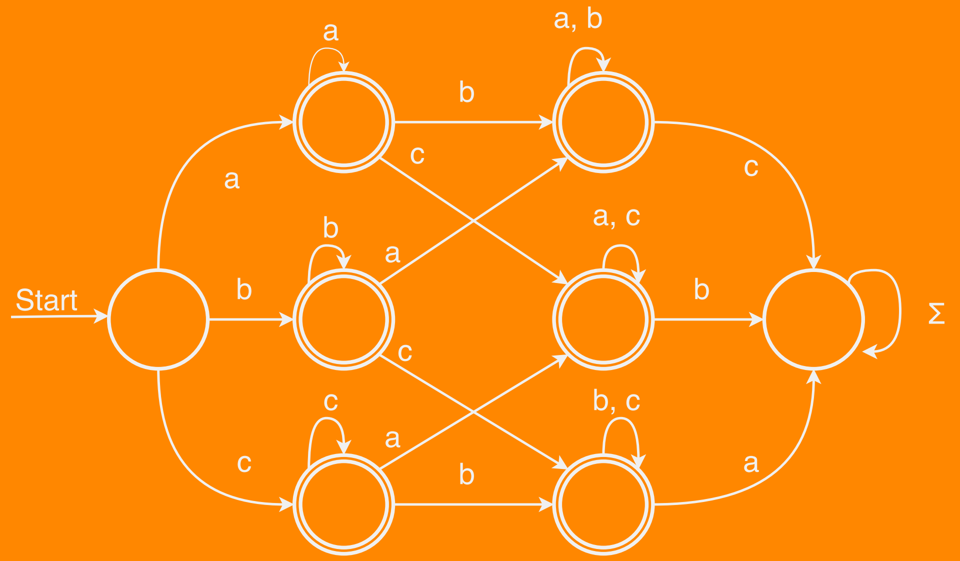
- Nondeterministically guess when to leave the start state
- Deterministically check whether that was the right time to do so

Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid$ at least one of a, b, or c is not in w $\}$

Guess-and-Check

L = { w ∈ {a, b, c}* | at least one of a, b, or c is not in w }



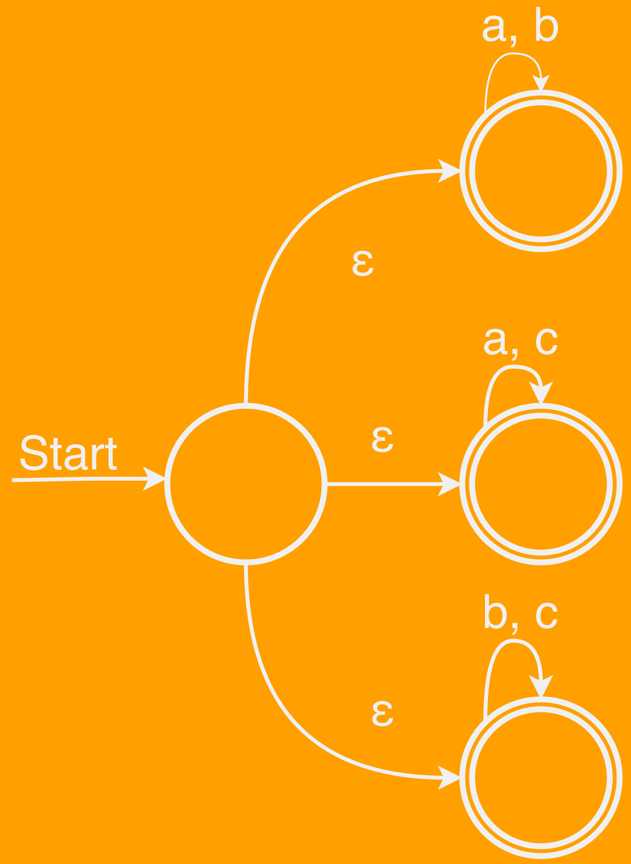
Guess-and-Check

L={w∈{a, b, c} * | at least one of a, b, or c is not in w}

- Nondeterministically guess which character is missing
- Deterministically check whether that character is indeed missing

Guess-and-Check

L={w∈{a, b, c} * | at least one of a, b, or c is not in w}



- Nondeterministically guess which character is missing
- Deterministically check whether that character is indeed missing

NFAs and DFAs

- Any language that can be accepted by a DFA can be accepted by an NFA.
- Why?
  - Every DFA essentially already is an NFA!
- Question: Can any language accepted by an NFA also be accepted by a DFA?
  - Surprisingly, the answer is **yes**!

See you in the lab! 👋