

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 \in , Oh!

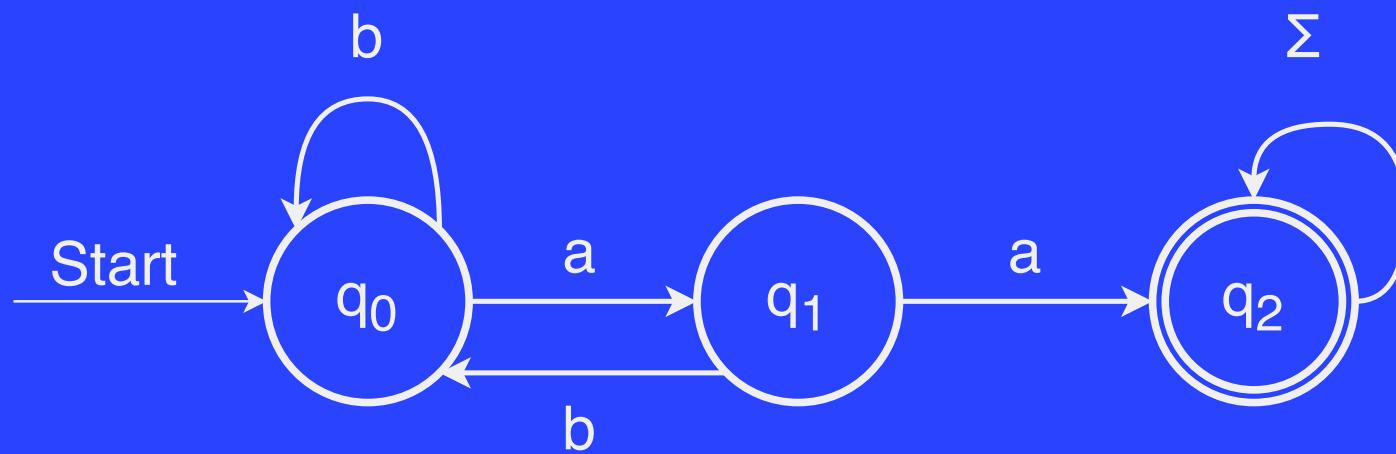
- Here's a quick guide to remembering which is which
 - Typically, we use the symbol Σ (sigma) to refer to an *alphabet*
 - The *empty string* is length 0 and is denoted ϵ (epsilon)
 - In set theory, use \in to say "is an *element of*"
 - In set theory, use \subseteq to say "is a *subset of*"

DFAs

- A **DFA** is a
 - Deterministic
 - Finite
 - Automaton

Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \}$



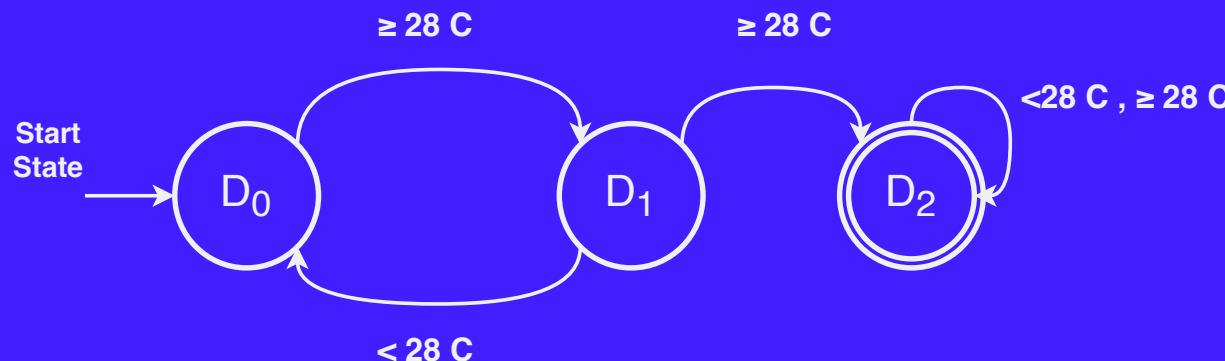
DFAs

- 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)



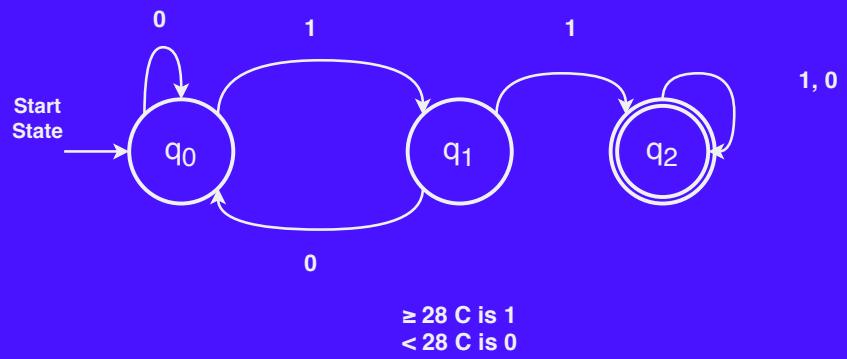
- Input: String of weather data
- 🇬🇧 Heatwave: temperature $\geq 28 \text{ C}$ for 2 consecutive days

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

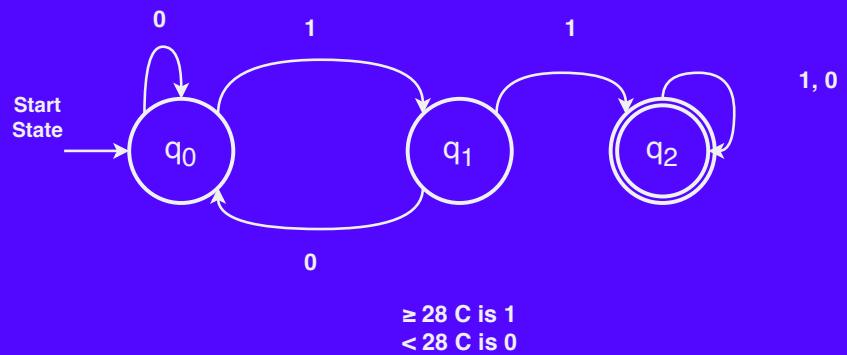


DFA Definition

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

- Q is the set of states [$Q = \{ q_0, q_1, q_2 \}$]
- Σ is the alphabet [$\Sigma = \{1, 0\}$]
- 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_1	q_2
q_2	q_2

Which table best represents the transitions for the following DFA?

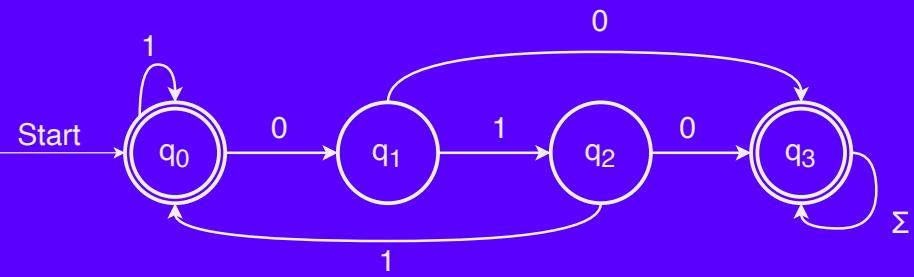


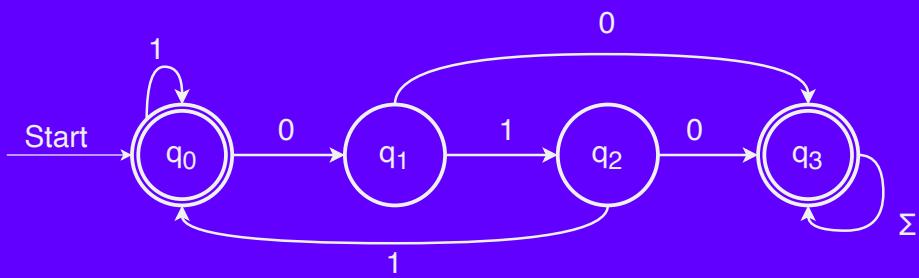
Table A

	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

**When I wrote this code,
only god & I understood what it did.**



Now... only god knows.

Thanks Joris!

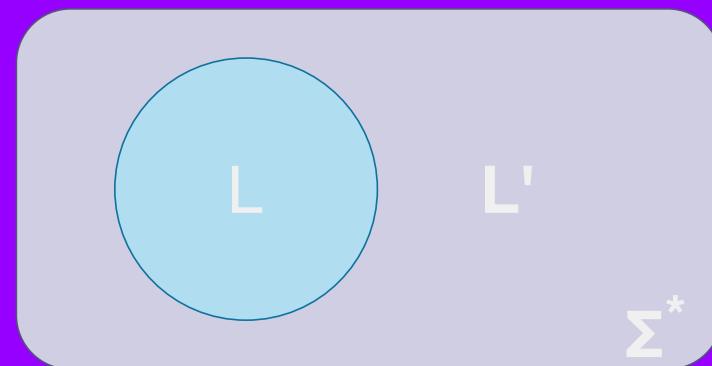
The Regular Languages

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

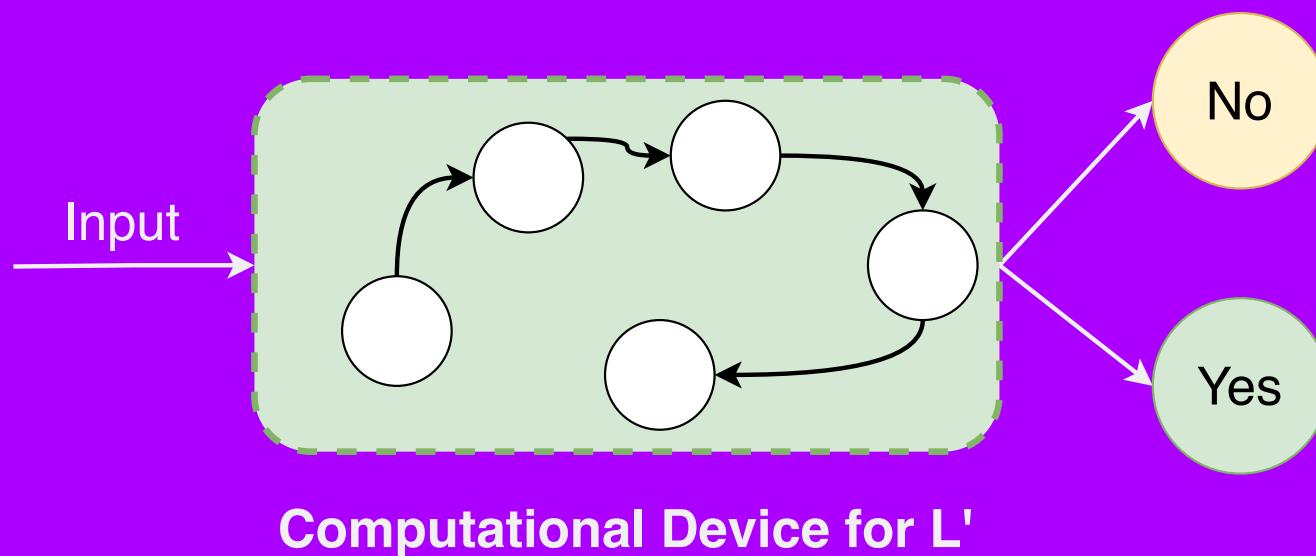
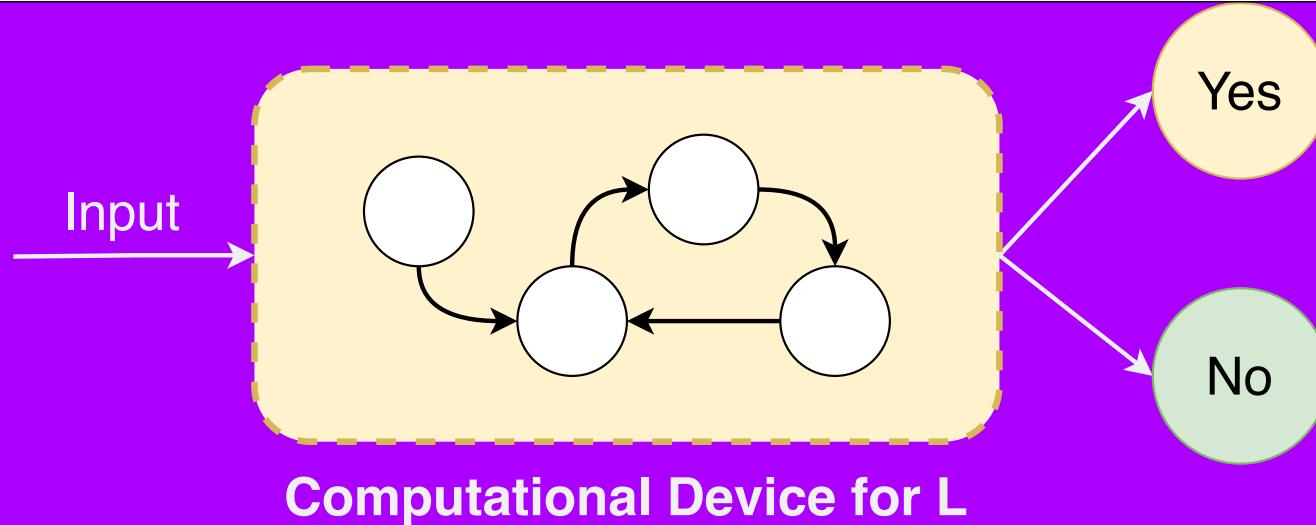
The Complement of a Language

- 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: $L' = \Sigma^* - L$



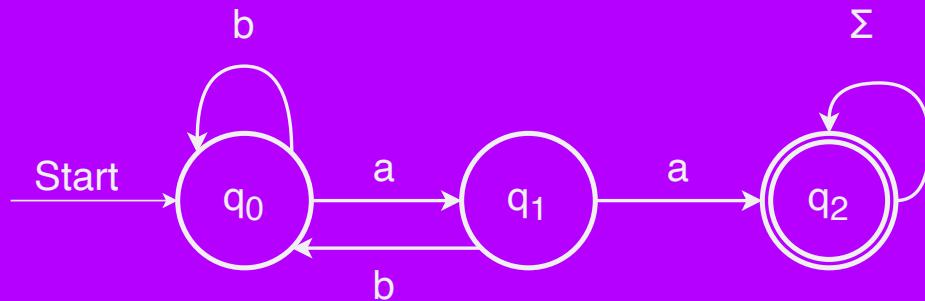
Complements of Regular Languages

- 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

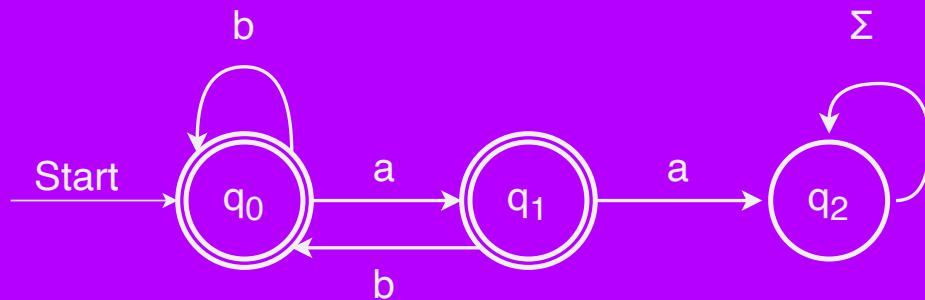


Complementing Regular Languages

$L = \{ w \in \{a,b\}^* \mid w \text{ contains aa as a substring} \}$

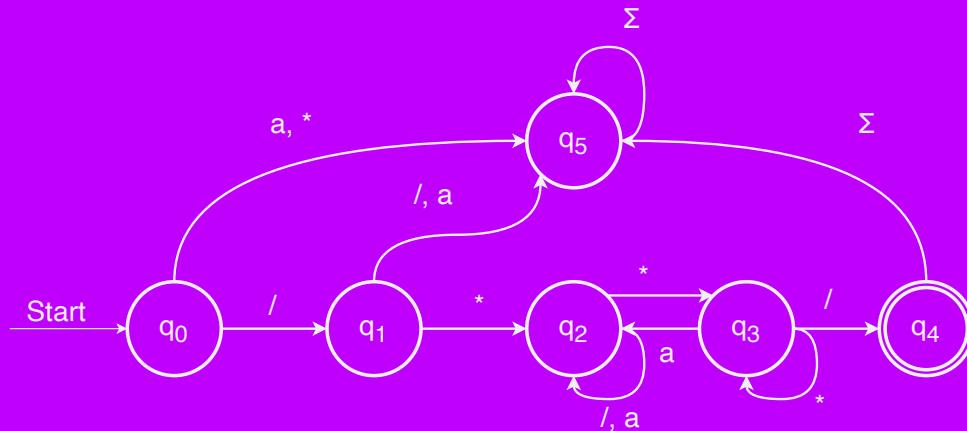


$L' = \{ w \in \{a,b\}^* \mid w \text{ does not contain aa as a substring} \}$



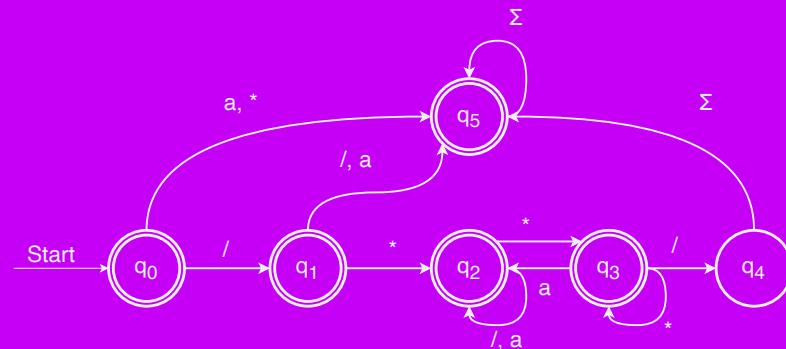
More Elaborate DFAs

$L = \{ w \in \{a, , /\}^* \mid w \text{ represents a } (\text{multi-line}) \text{ Java-style comment} \}$



More Elaborate DFAs

$L' = \{ w \in \{a, , /\}^* \mid w \text{ doesn't represent a } (\mathbf{multi-line}) \text{ Java-style comment} \}$



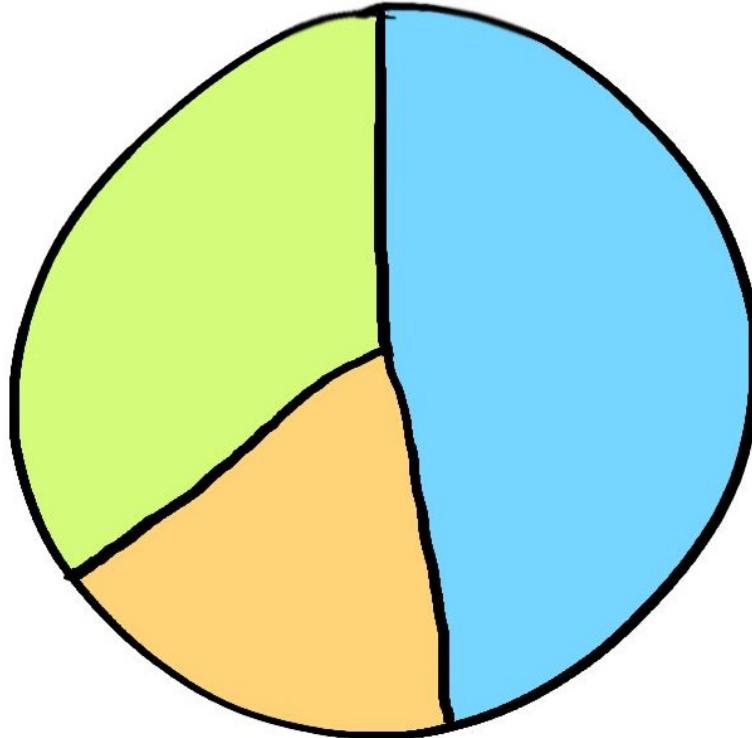
Closure Properties

- **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?**



- PEOPLE WHO GET IMPOSTER SYNDROME
- OTHER PEOPLE WHO GET IMPOSTER SYNDROME
- LITERALLY EVERYONE ELSE (THEY ALSO GET IMPOSTER SYNDROME)

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

NFAs

Part 3/4

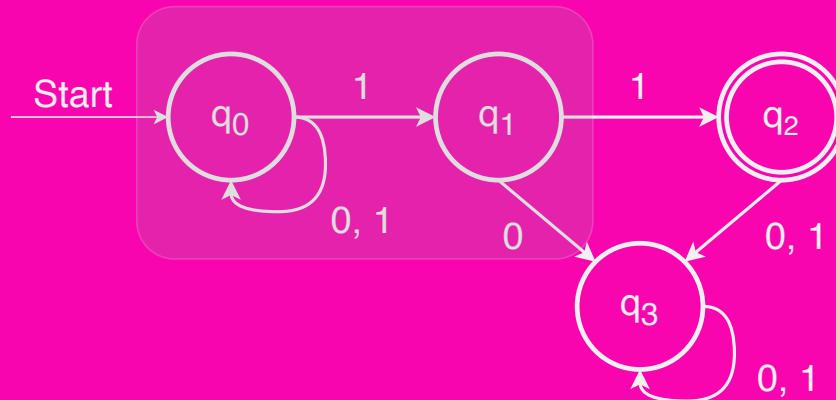
NFAs

- An **NFA** is a
 - Nondeterministic
 - Finite
 - Automaton
- Structurally similar to a DFA, but represents a fundamental shift in how we'll think about computation

(Non)determinism

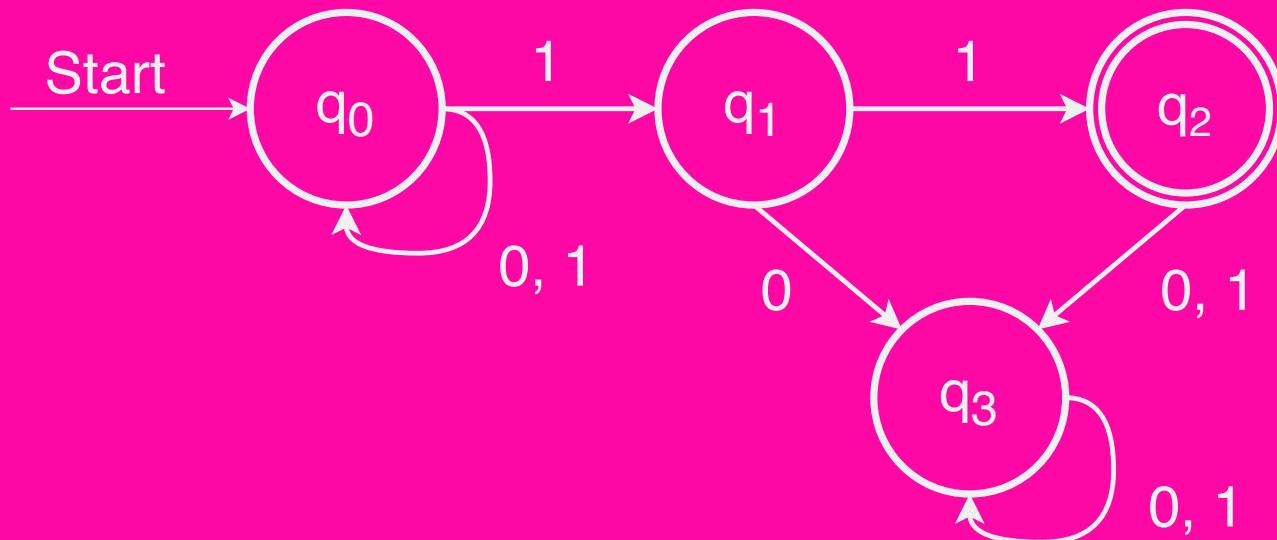
- A model of computation is **deterministic** if at every point in the computation, there is exactly one choice that can make
- The machine accepts if that series of choices leads to an accepting state
- A model of computation is **nondeterministic** if the computing machine may have multiple decisions 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



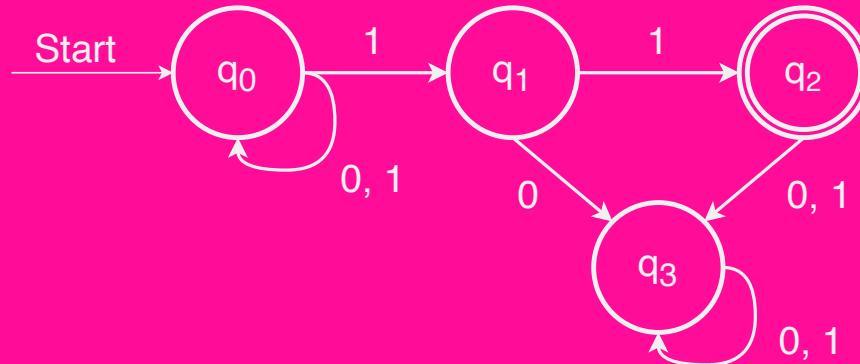
| q_0 has two transitions defined
| on 1!

A Simple NFA



Input: 01011

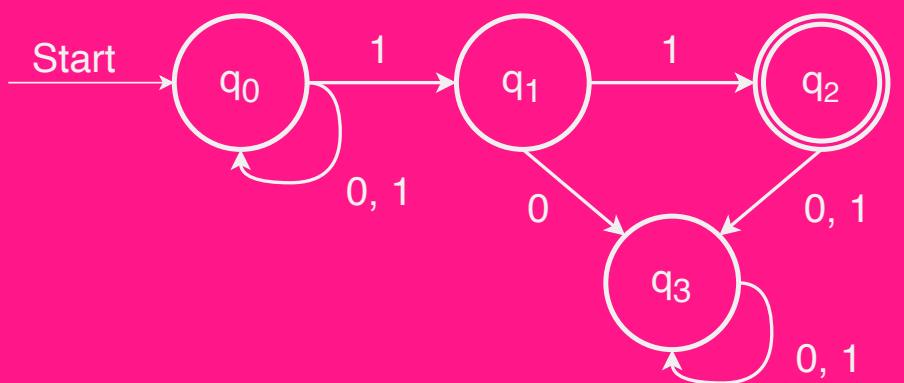
Non-Deterministic Finite Automaton (Formal Definition)



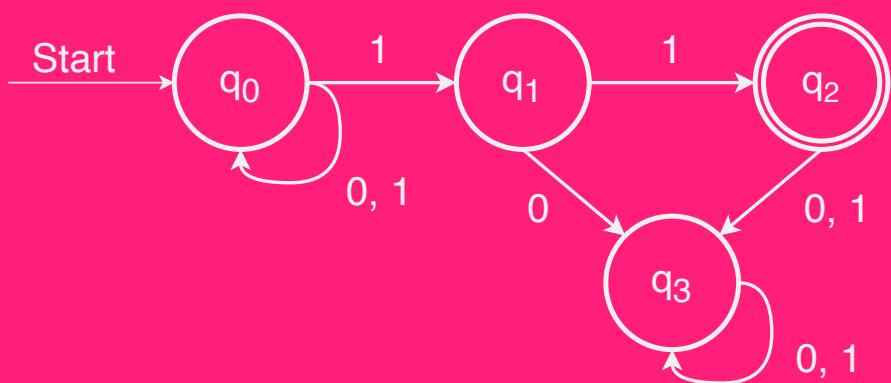
$$D = (Q, \Sigma, \delta, q_0, F)$$

- Q is the set of states [$Q = \{ q_0, q_1, q_2, q_3 \}$]
- Σ is the alphabet [$\Sigma = \{1,0\}$]
- 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

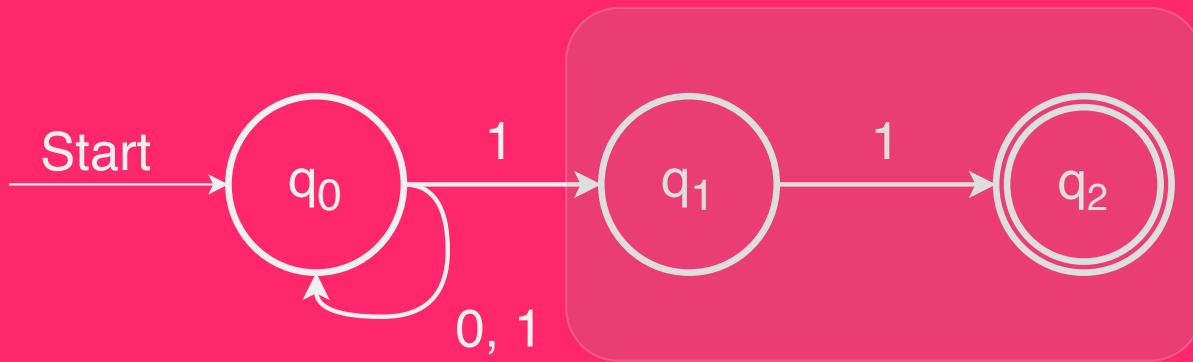


A Simple NFA: Transaction Function

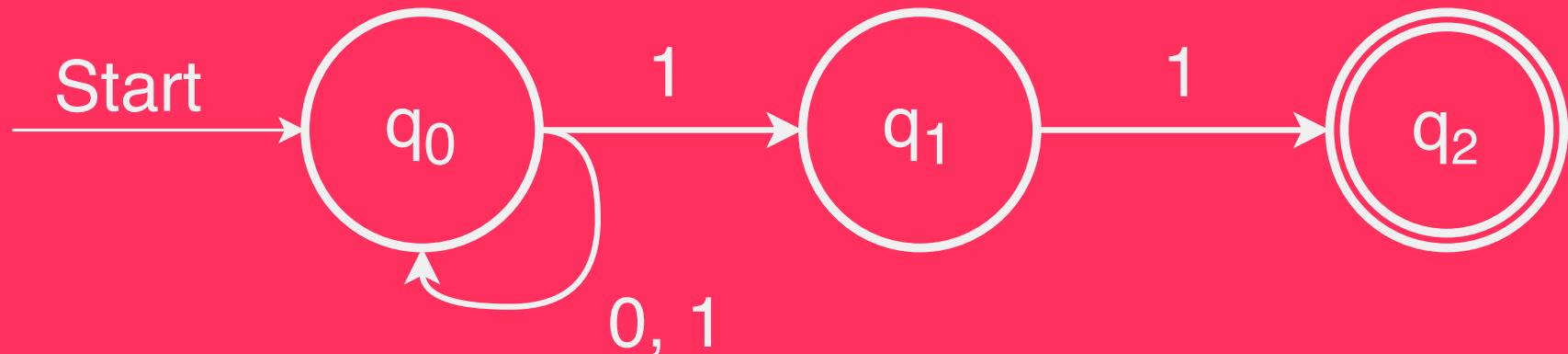


State	0	1
q_0	$\{ q_0 \}$	$\{ q_0, q_1 \}$
q_1	$\{ q_3 \}$	$\{ q_2 \}$
q_2	$\{ q_3 \}$	$\{ q_3 \}$
q_3	$\{ q_3 \}$	$\{ q_3 \}$

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?

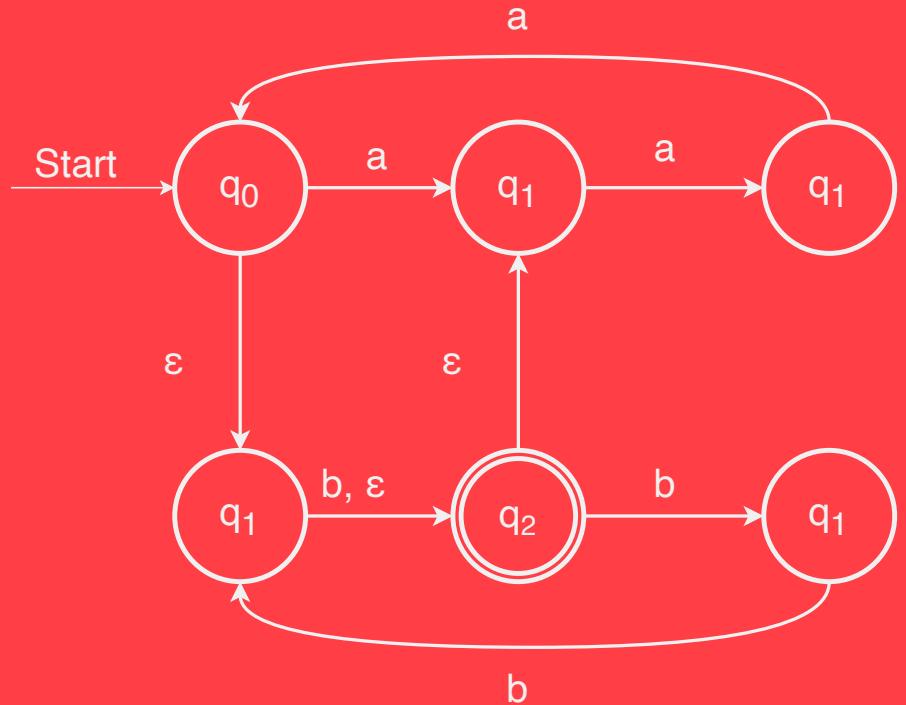
- A) $\{01011\}$
- B) $\{ w \in \{0,1\}^* \mid w \text{ contains at least two 1s } \}$
- C) $\{ w \in \{0, 1\}^* \mid w \text{ ends with } 11 \}$
- D) $\{ w \in \{0, 1\}^* \mid w \text{ ends with } 1 \}$
- E) None of these, or two or more of these

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

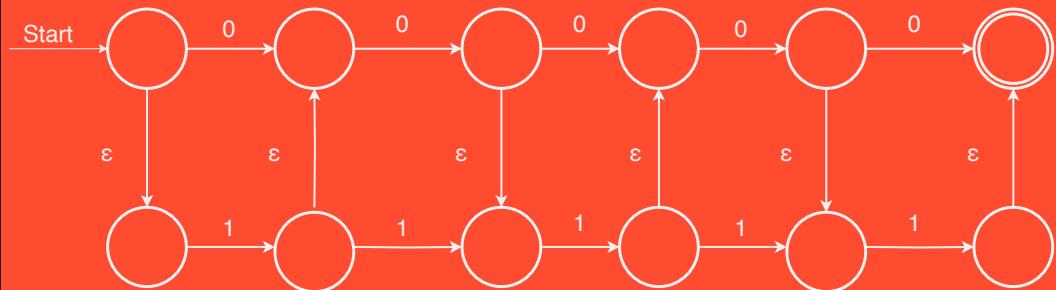


Input: b a a b b

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

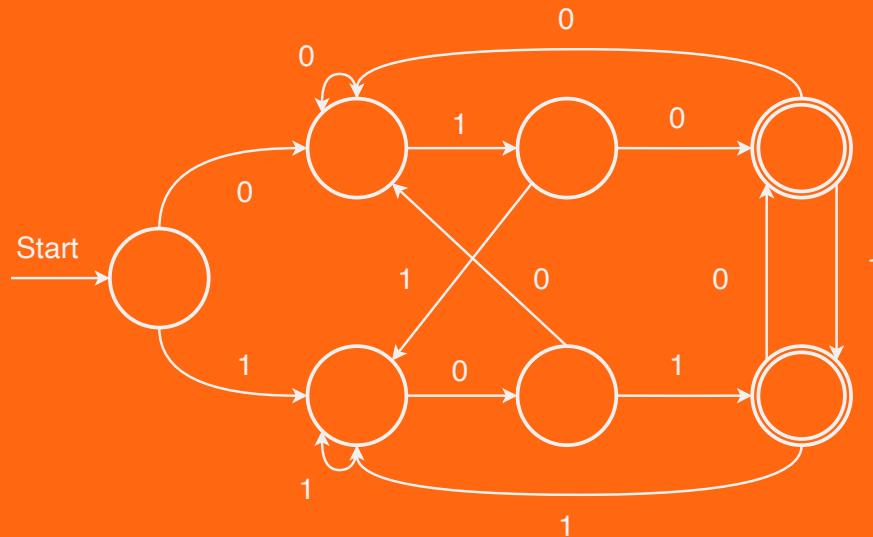
- When designing NFAs, embrace the nondeterminism!
- Good model: **Guess-and-check**:
 - Is there some information that you'd really like to have? Have the machine nondeterministically guess that information
 - Then, have the machine deterministically check that the choice was correct
 - The *guess*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

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



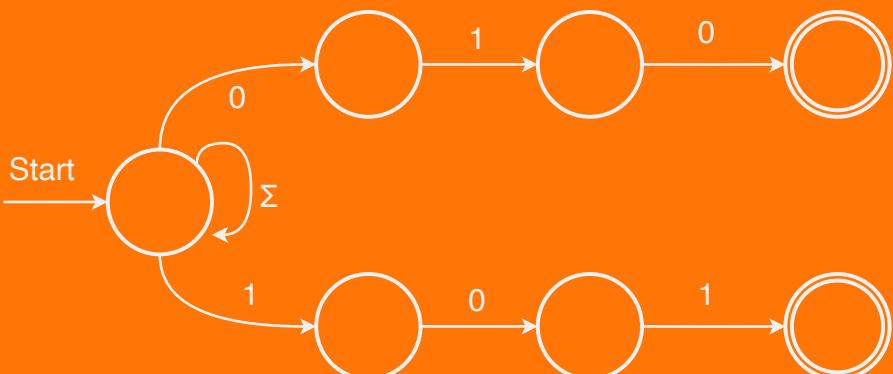
Guess-and-Check

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

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

Guess-and-Check

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



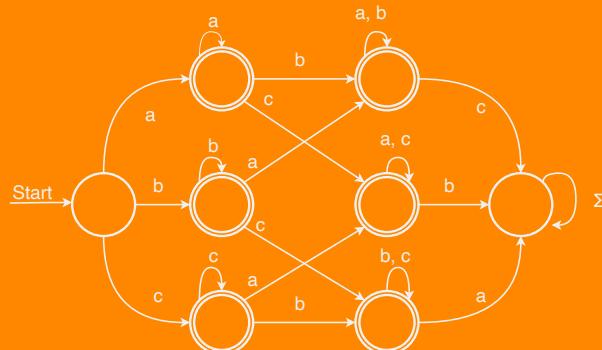
- Nondeterministically guess when to leave the start state
- Deterministically check whether that was the right time to do so

Guess-and-Check

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

Guess-and-Check

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



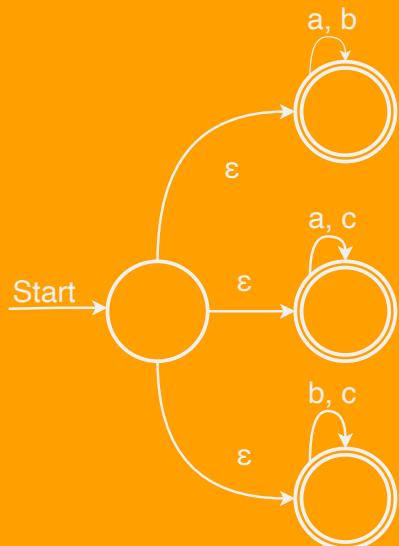
Guess-and-Check

$L = \{w \in \{a, b, c\}^* \mid$
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

$L = \{w \in \{a, b, c\}^* \mid$
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! 