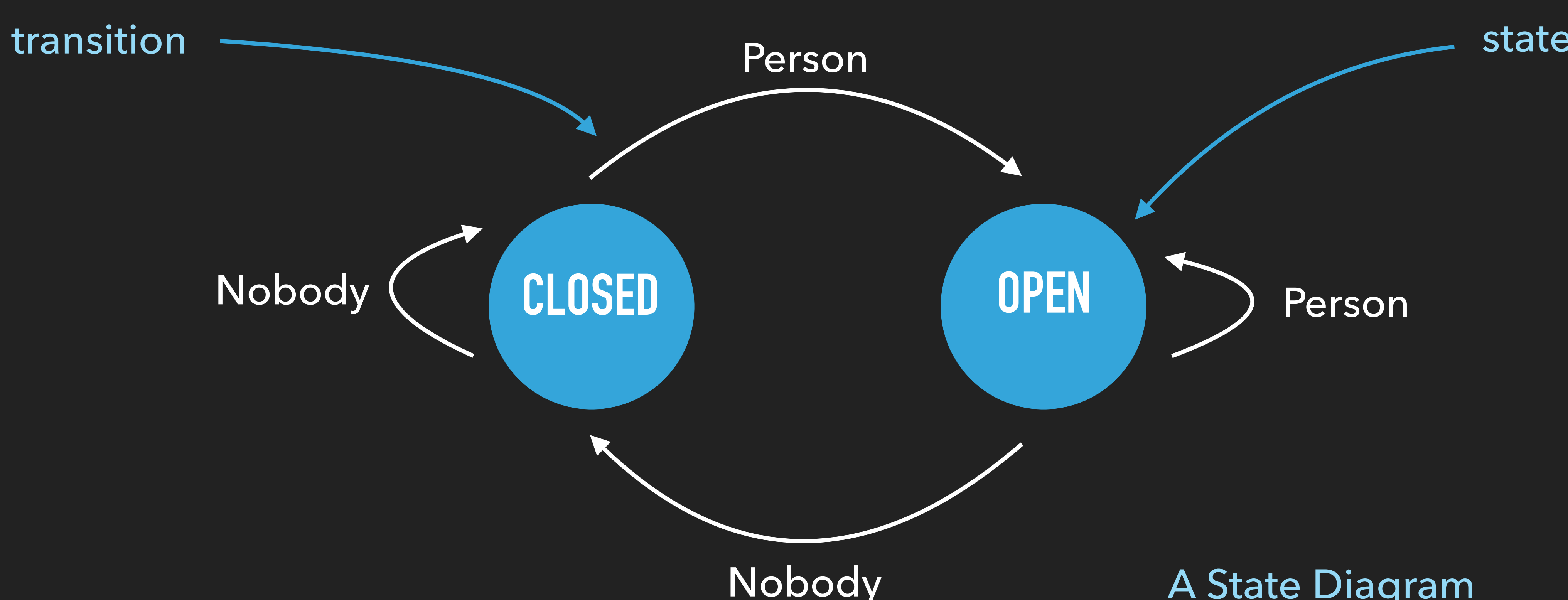
CSC 240

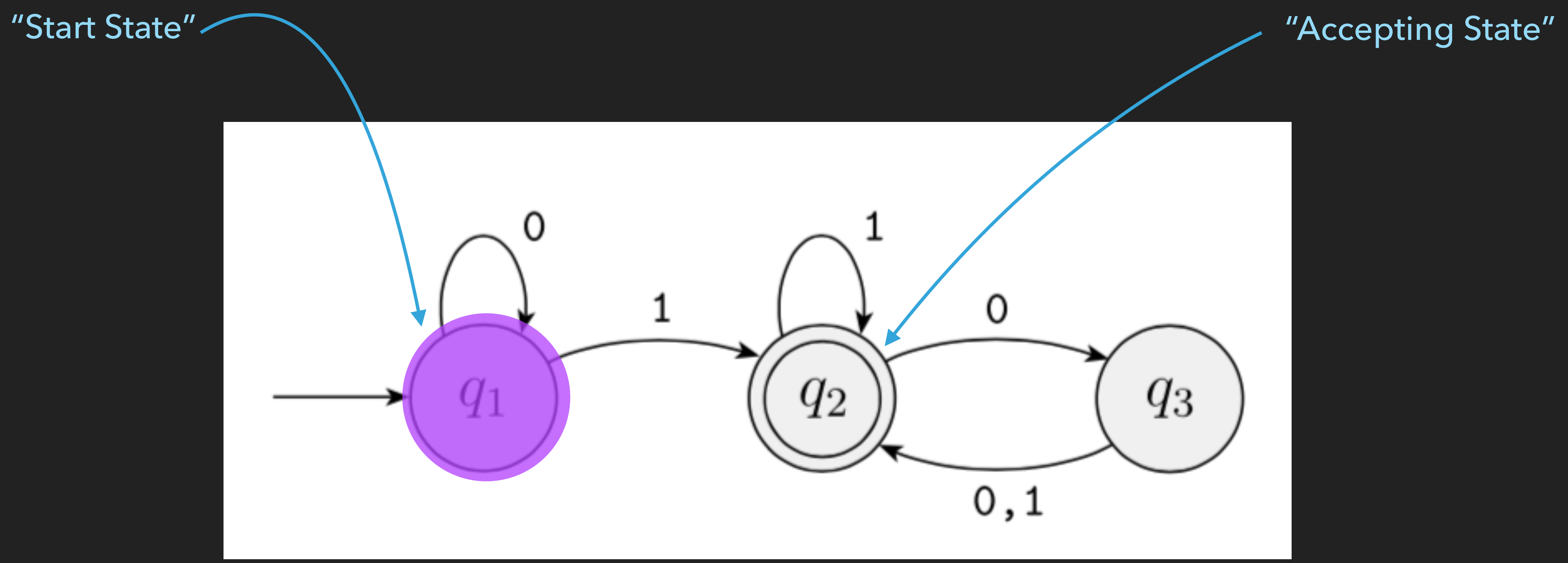
NON-DETERMINISTIC FINITE AUTOMATA (NFA)

AUTOMATON



A State Diagram

FINITE AUTOMATA



Input: 1101 Accepted

Input: 0010 Not Accepted

This finite automata accepts any string that ends in 1 and any string that ends with an even number of 0's following a 1.

DETERMINISTIC FINITE AUTOMATA (DFA)

- Defined relative to an alphabet.
- Each state has exactly one transition for each symbol in the alphabet.
- Has a unique Start State.
- Has zero or more more accepting states.

DFA: Defined by a 5-tuple: $(Q, \Sigma, \delta, q_0, F)$

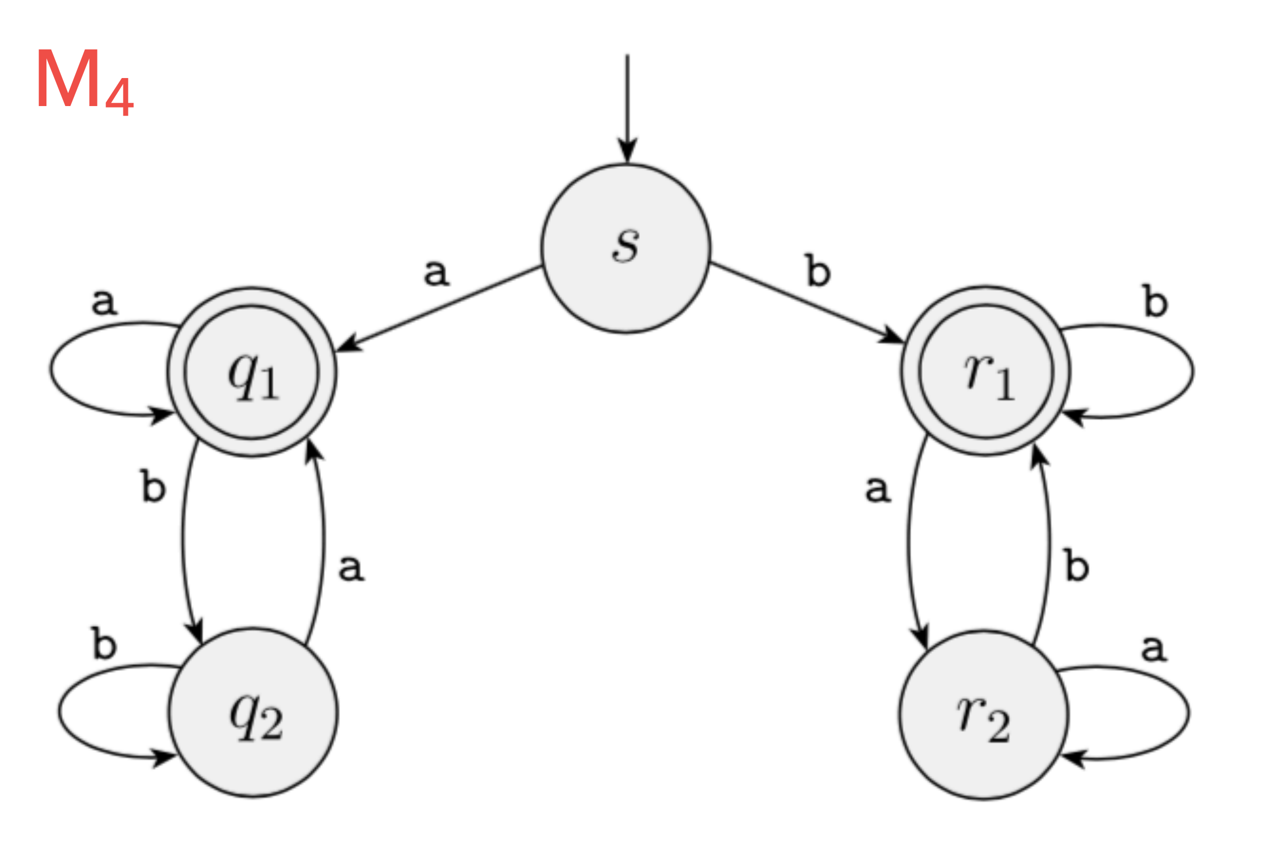
Q: A finite set called states.

Σ: A finite set called the alphabet.

$\delta: Q \times \Sigma \rightarrow Q$ is the transition function.

q_0: is the start state.

F: is the set of accepting states where $F \subseteq Q$.



$M_4 = \{\{s, q_1, q_2, r_1, r_2\}, \{a, b\}, \delta, s, \{q_1, r_1\}\}$

$\mathscr{L}(M_4) = \{ w \mid w$ start and end with the same symbol$\}$

δ	a	b
s	q_1	r_1
q_1	q_1	q_2
q_2	q_1	q_2
r_1	r_1	r_2
r_2	r_2	r_1

SOME DEFINITIONS

Character: A single symbol.

Alphabet (Σ): A finite, non-empty set of characters.

String Over Alphabet Σ: A finite, sequence of characters drawn from Σ.

Empty String (ε): A string containing no characters.

A Formal Language: A set of strings.

LANGUAGE OF AN AUTOMATA

The Language of an Automata: The set of strings accepted by the automata.

$$\mathscr{L}(M) = A$$

The $\mathscr{L}$anguage of automata M

A is the set of all strings accepted by M.

"M *Recognizes A*"

Regular Language: A language that is recognized by a DFA.

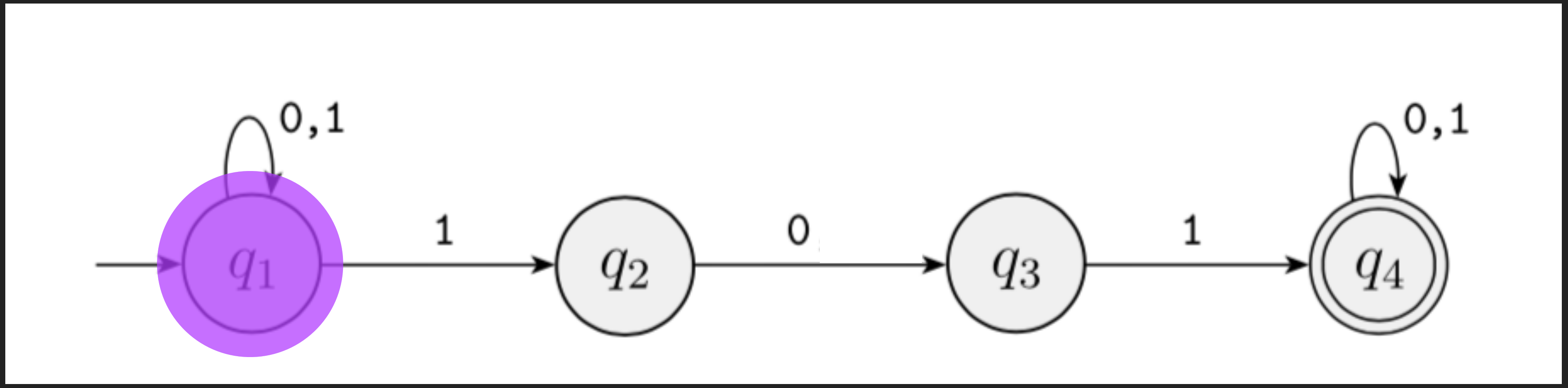
$$\mathscr{L}(M) = A$$

"A is a Regular Language"

NON-DETERMINISTIC FINITE AUTOMATA (NFA)

- Defined relative to an alphabet.
- Each state has zero or more transitions for each symbol in the alphabet.
- Has a unique Start State.
- Has zero or more more accepting states.

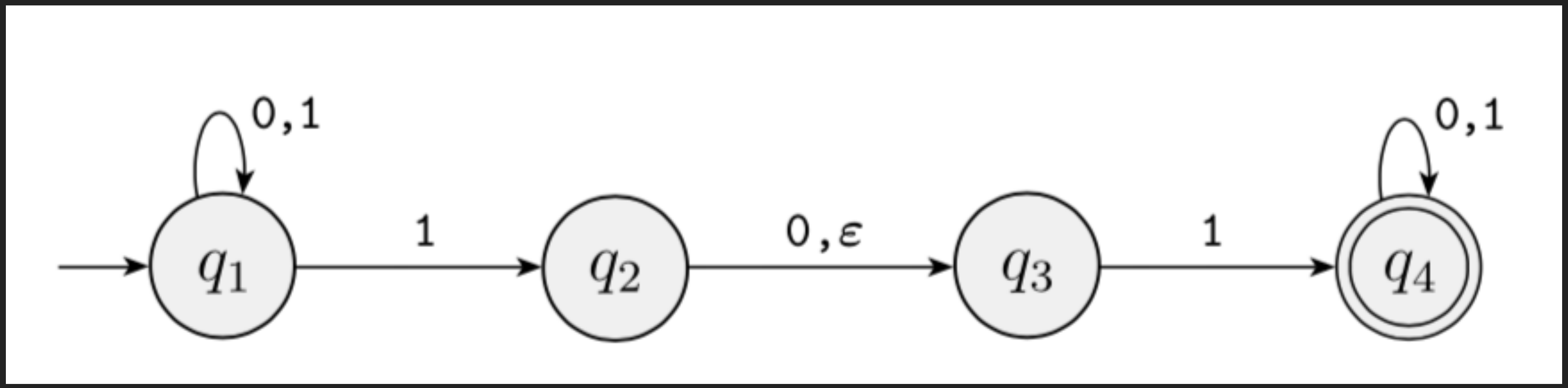
NON-DETERMINISTIC FINITE AUTOMATA (NFA)



NFAs have multiple transitions they could make at each state.

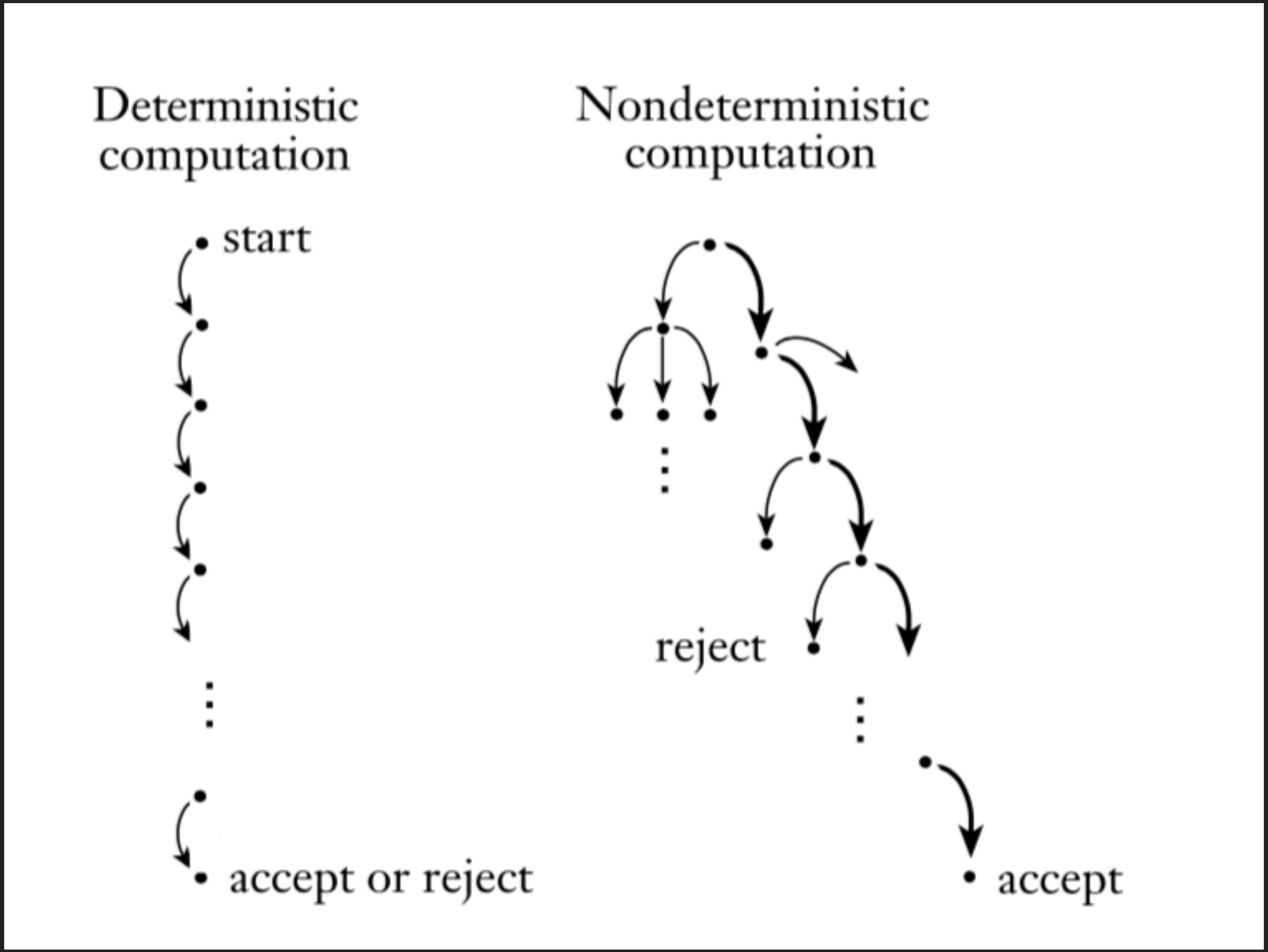
NFAs accept an input if *any* set of valid transitions lead to an accept state.

Input: 1101 Accepted

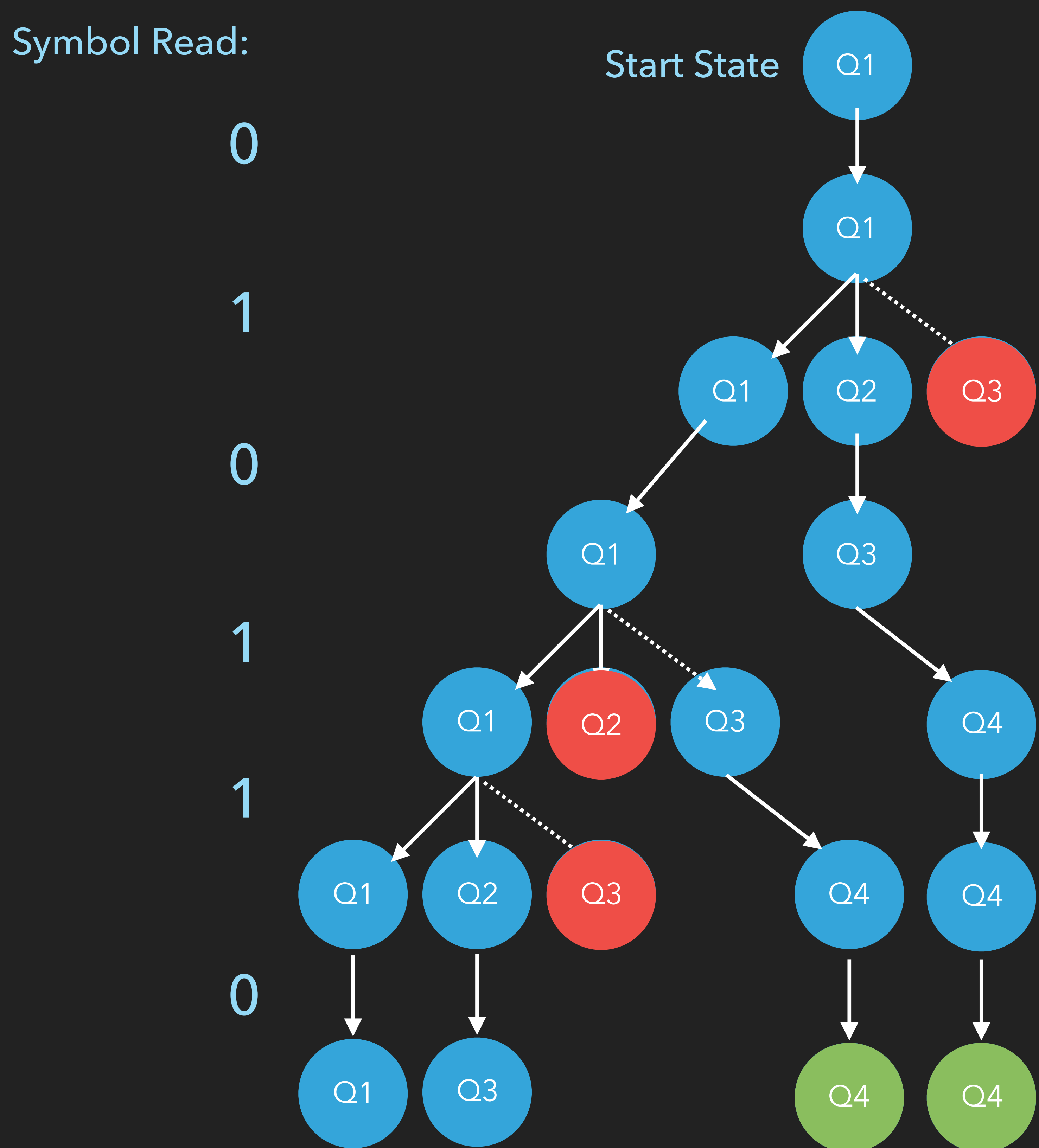


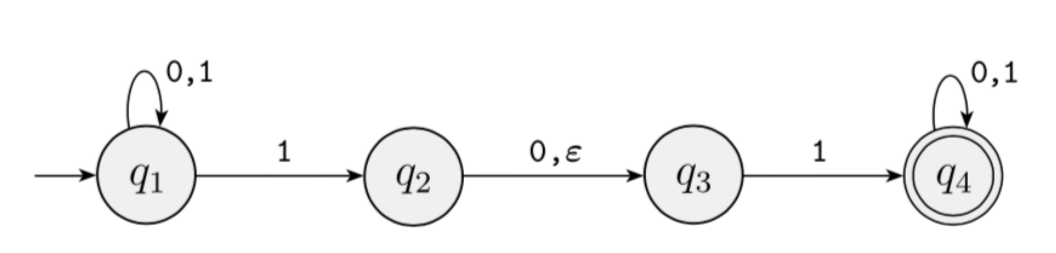
ε transition: A transition that doesn't consume input.

DFA Computation vs. NFA Computation



NON-DETERMINISTIC FINITE AUTOMATA (NFA)



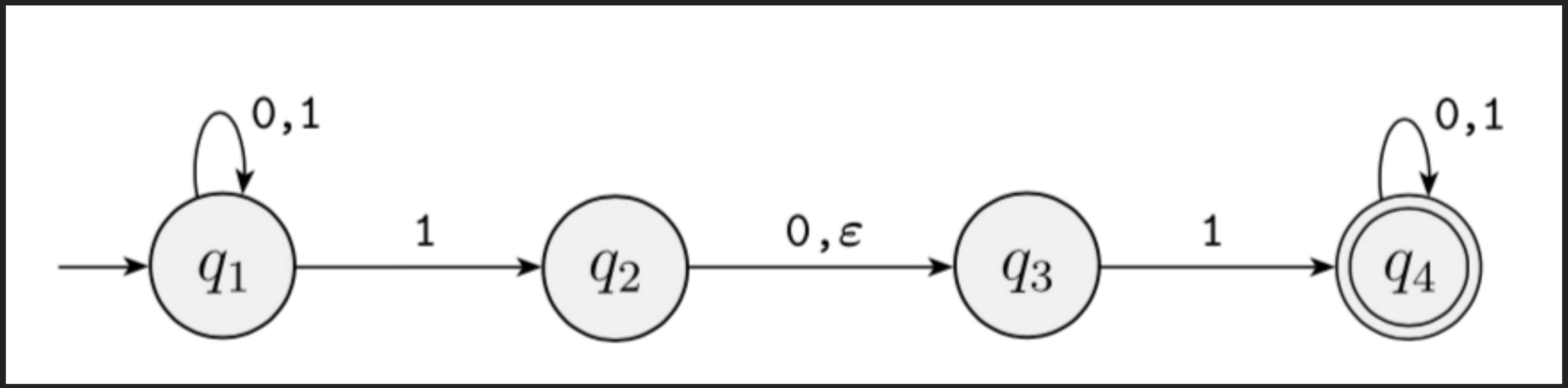


Input: 010110

Accepted

NFAs accept an input if *any* set of valid transitions lead to an accept state.

NON-DETERMINISTIC FINITE AUTOMATA (NFA)



Two ways to think about NFAs:

1. NFAs are able to guess the correct sequence of transitions to use in order to reach an accepted state.
2. NFAs attempt all possible sequences of transitions simultaneously.

Questions:

1. If a language is accepted by a DFA, does an NFA exist that will accept it?

Yes! Because every DFA *is* an NFA already, it just has a single path.

2. If a language is accepted by an NFA, does an DFA exist that will accept it?

Yes! Find the proof of this idea on page 55 of the Sipser text.

DFA: Defined by a 5-tuple: $(Q, \Sigma, \delta, q_0, F)$

NFA: Defined by a 5-tuple: $(Q, \Sigma, \delta, q_0, F)$

Q: A finite set called states.

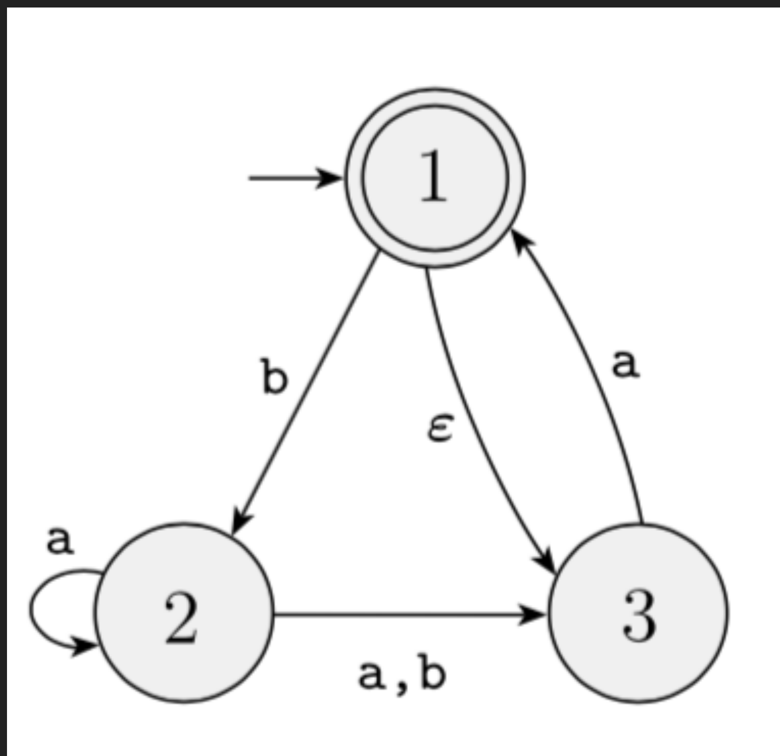
Σ: A finite set called the alphabet.

$\delta: Q \times \Sigma_\varepsilon \rightarrow \mathscr{P}(Q)$ is the transition function.

q_0: is the set of start states.

F: is the set of accepting states where $F \subseteq Q$.

CREATING A DFA FROM AN NFA - THE SUBSET (OR POWERSET) CONSTRUCTION



Q: {1, 2, 3}

$\mathscr{P}$(Q): {∅, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3} }

Σ: {a, b}

q_0: {1}

F: {1}

Accepted: ε, a, baba, and baa

Rejected: b, bb, and babba

δ	a	b	
{1}	{1, 3}	{2}	*
{1, 3}	{1, 3}	{2}	*
{2}	{2, 3}	{3}	
{2, 3}	{1, 2, 3}	{3}	
{3}	{1}	∅	
{1, 2, 3}	{1, 2, 3}	{2, 3}	*