Nondeterminism

CSCI 338

DFA vs NFA

DFA:

??????

DFA vs NFA

DFA: Model of a computer that determines (accept or reject) if a string has a specific format.

DFA vs NFA

NFA

~~DFA~~: Model of a computer that determines (accept or reject) if a string has a specific format.

With some more bells and whistles.

DFA vs NFA

Deterministic Finite Automaton (DFA):

Nondeterministic Finite Automaton (NFA):

DFA vs NFA

Deterministic Finite Automaton (DFA):

- Every state has exactly one transition for every character ($e \in \Sigma$).

Nondeterministic Finite Automaton (NFA):

DFA vs NFA

Deterministic Finite Automaton (DFA):

- Every state has exactly one transition for every character ($e \in \Sigma$).

Nondeterministic Finite Automaton (NFA):

- Allowed to have multiple (or 0) transitions for each $e \in \Sigma$.

DFA vs NFA

Deterministic Finite Automaton (DFA):

- Every state has exactly one transition for every character ($e \in \Sigma$).

Nondeterministic Finite Automaton (NFA):

- Allowed to have multiple (or 0) transitions for each $e \in \Sigma$.
- Allowed to have transitions that happen without input.

DFA vs NFA

Deterministic Finite Automaton (DFA):

- Every state has exactly one transition for every character ($e \in \Sigma$).
- For the same input, everyone takes the same path to the same final state.

Nondeterministic Finite Automaton (NFA):

- Allowed to have multiple (or 0) transitions for each $e \in \Sigma$.
- Allowed to have transitions that happen without input.

DFA vs NFA

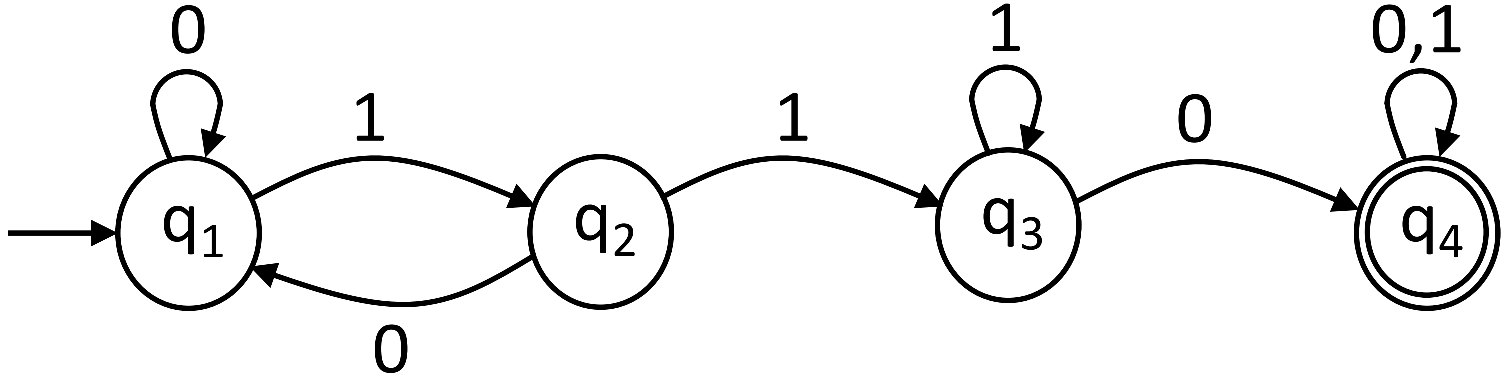
Deterministic Finite Automaton (DFA):

- Every state has exactly one transition for every character ($e \in \Sigma$).
- For the same input, everyone takes the same path to the same final state.

Nondeterministic Finite Automaton (NFA):

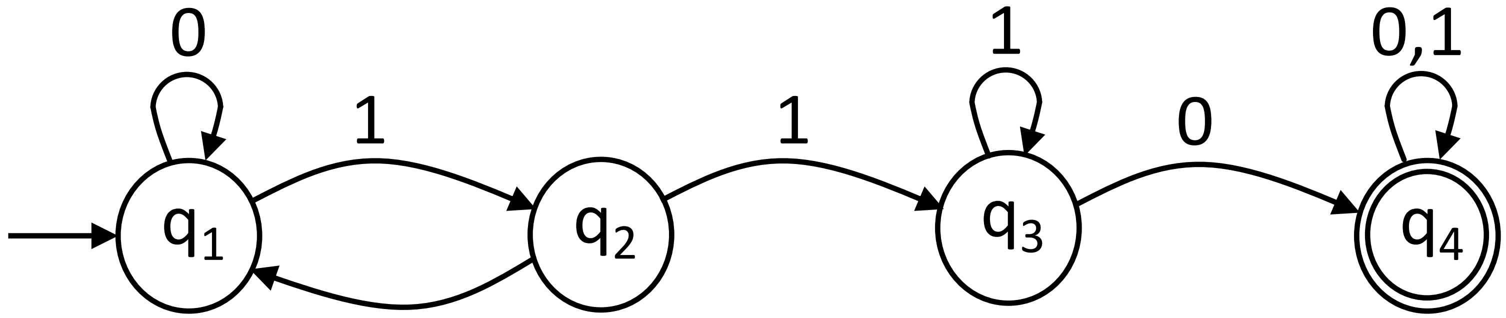
- Allowed to have multiple (or 0) transitions for each $e \in \Sigma$.
- Allowed to have transitions that happen without input.
- Processing strings is…different.

DFA Processing



$\{\omega\text{: } \omega$ contains the substring $110\}$

DFA Processing

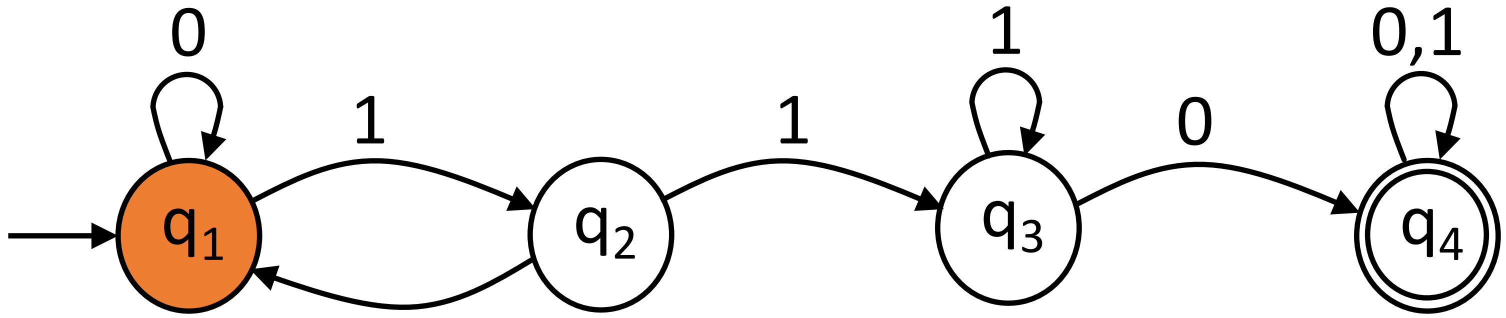


$\{\omega\text{: } \omega$ contains the substring $110\}$

$$\omega = 1110$$

DFA Processing

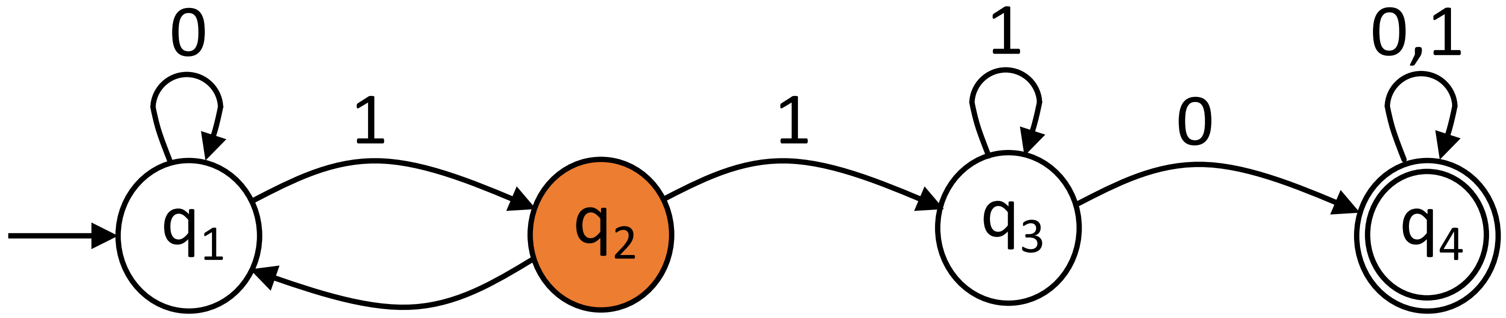
q_1



$\{\omega\colon\ \omega$ contains the substring $110\}$

$$\omega = 1110$$

DFA Processing



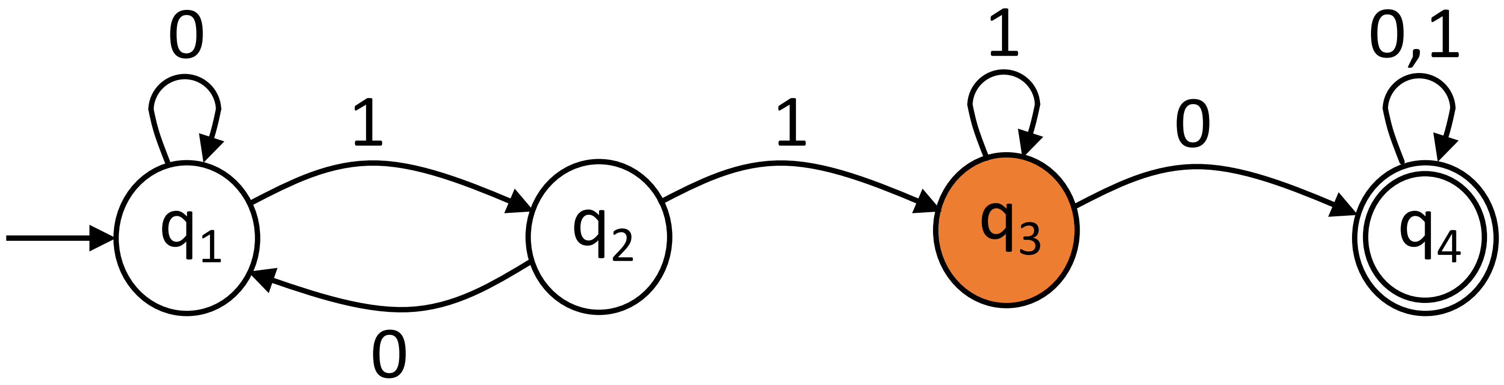
$\{\omega\colon\ \omega$ contains the substring $110\}$

$$q_1 \xrightarrow{} q_2 \quad 1$$

$$\omega = 1110$$

DFA Processing

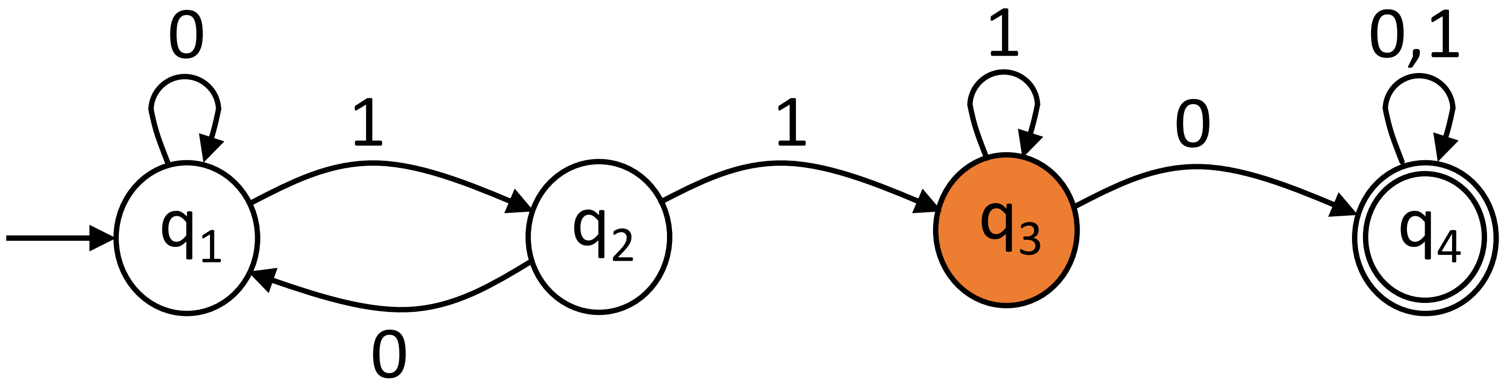
$$\begin{tikzcd}
q_1 \arrow[d, "1"] \\
q_2 \arrow[d, "1"] \\
q_3
\end{tikzcd}$$



$\{\omega\colon\ \omega$ contains the substring $110\}$

$$\omega = 1110$$

DFA Processing

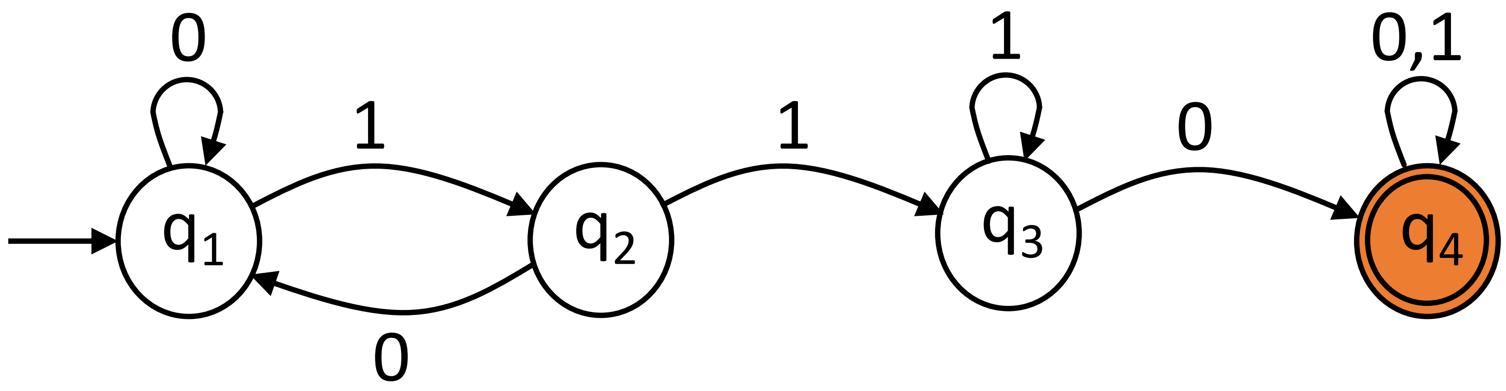


$\{\omega\text{: } \omega \text{ contains the substring } 110\}$

q_1
↓ 1
q_2
↓ 1
q_3
↓ 1
q_3

$\omega = 1110$

DFA Processing

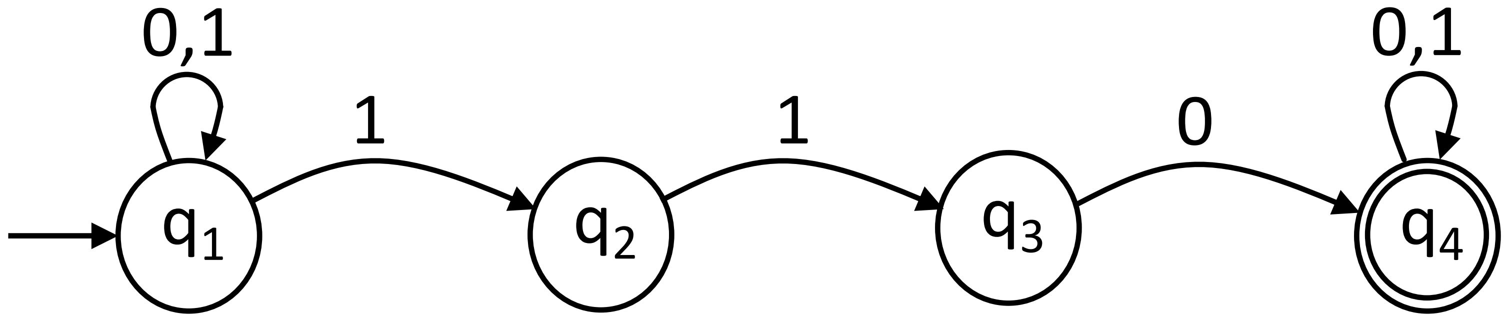


$\{\omega\text{: } \omega \text{ contains the substring } 110\}$

$$\begin{tikzcd}
q_1 \arrow[d, "1"] \\
q_2 \arrow[d, "1"] \\
q_3 \arrow[d, "1"] \\
q_3 \arrow[d, "0"] \\
q_4
\end{tikzcd}$$

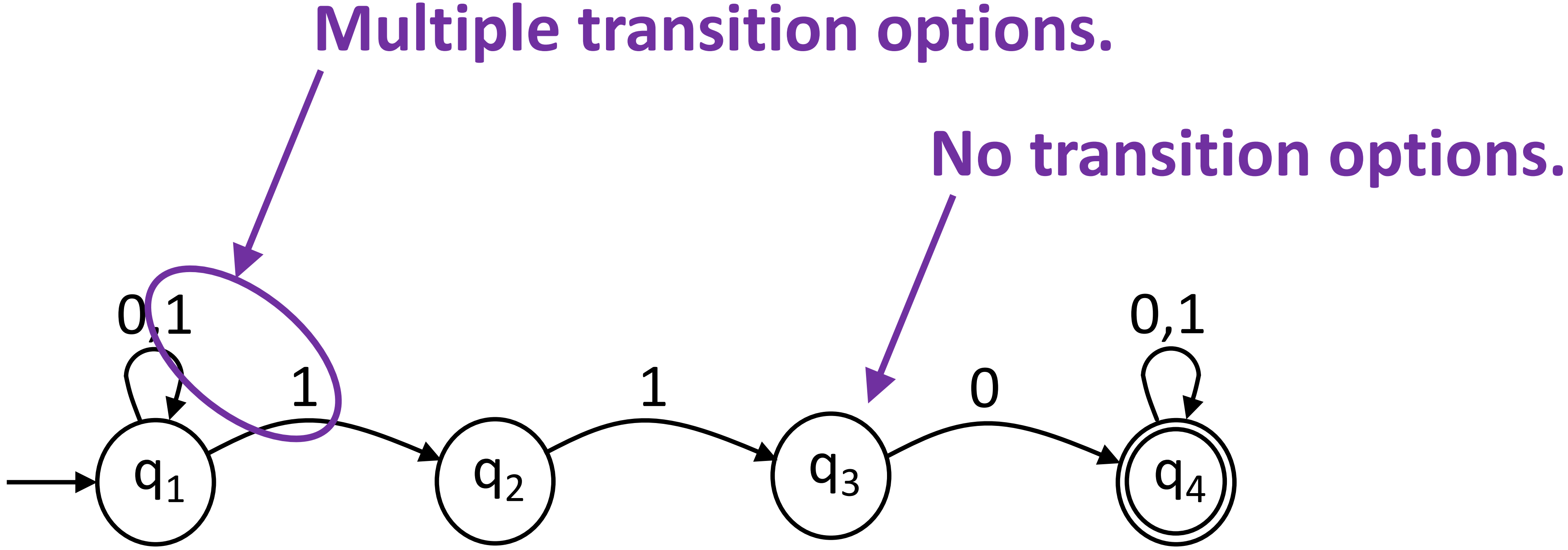
$$\omega = 1110$$

NFA Processing



$\{\omega$: ω contains the substring $110\}$

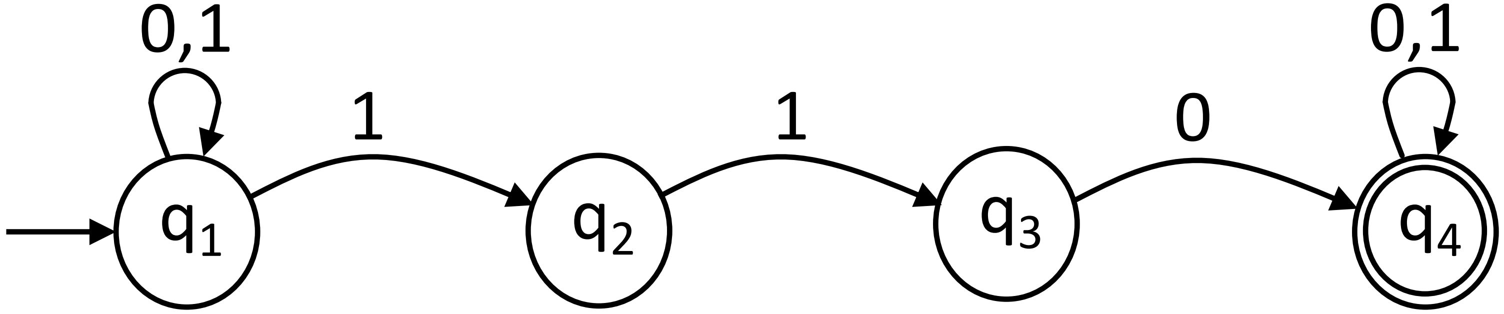
NFA Processing



$\{\omega\colon\ \omega$ contains the substring $110\}$

NFA Processing

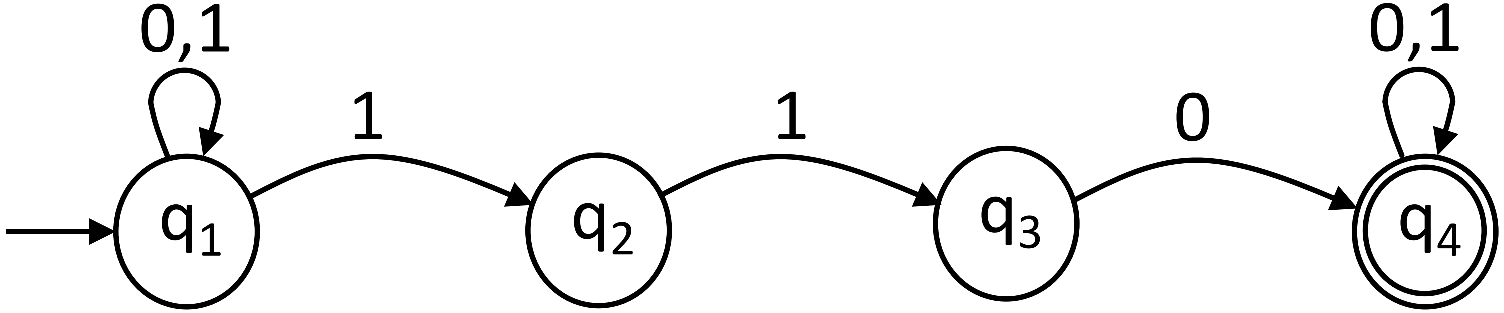
1. If a “decision” is encountered, split and take all options.



$\{\omega\colon\ \omega$ contains the substring $110\}$

NFA Processing

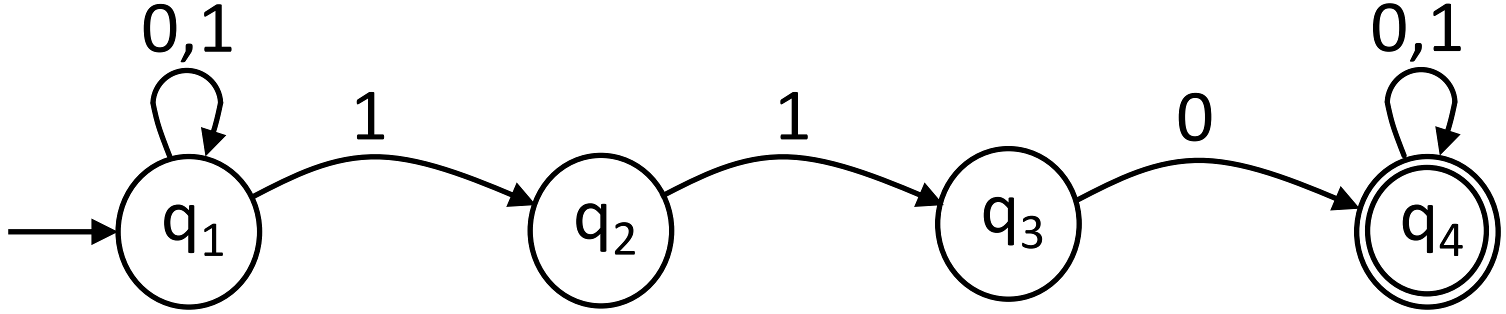
1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.



$\{\omega\colon\ \omega$ contains the substring $110\}$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.

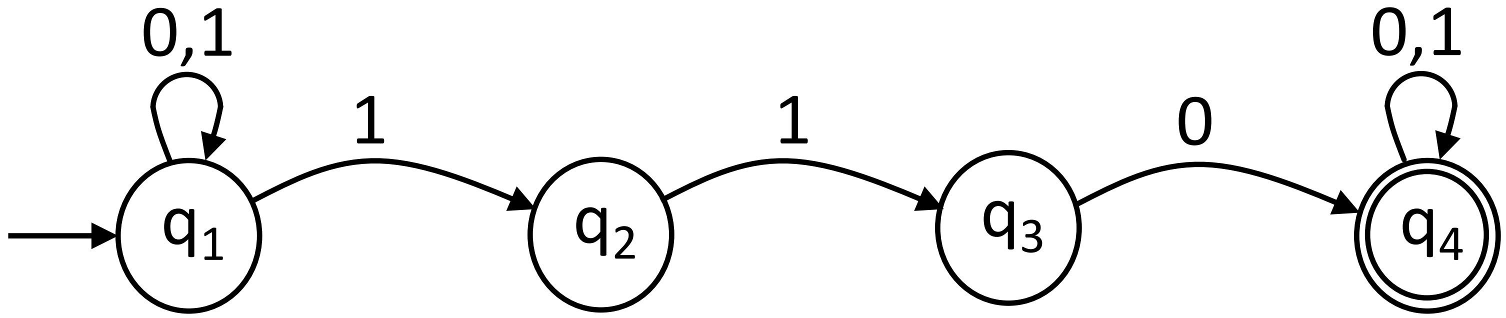


$\{\omega\colon\ \omega$ contains the substring $110\}$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.

q_1

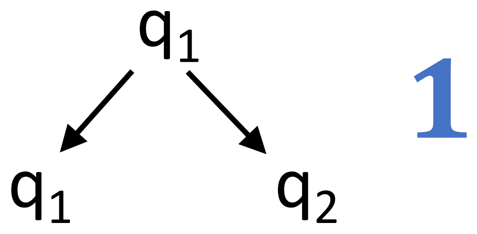


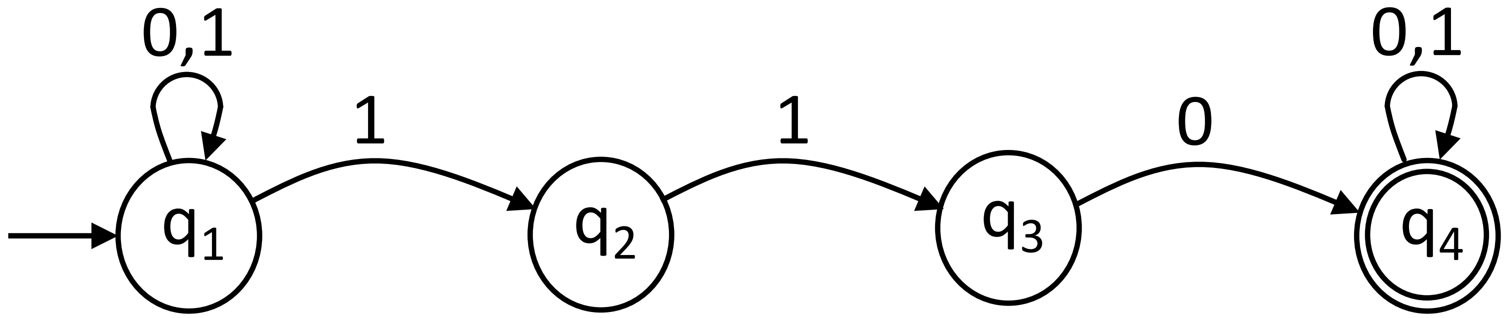
$\{\omega\colon\ \omega$ contains the substring $110\}$

$$\omega = 1110$$

NFA Processing

1. **If a "decision" is encountered, split and take all options.**
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



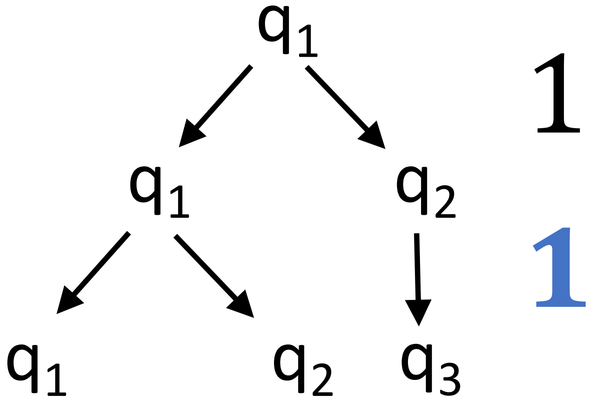


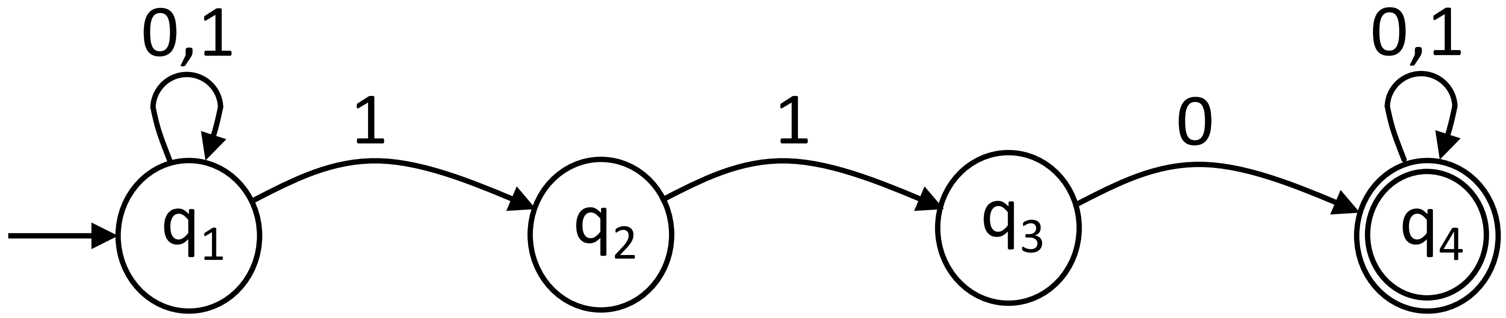
$\{\omega\colon\ \omega$ contains the substring $110\}$

$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



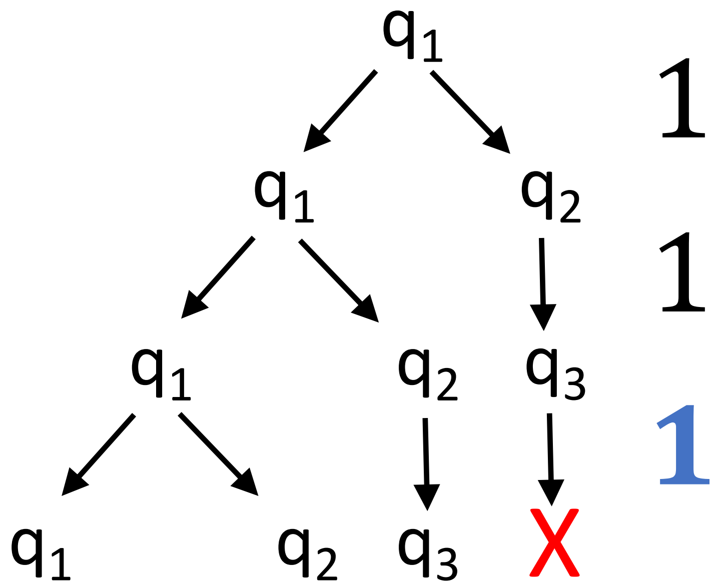


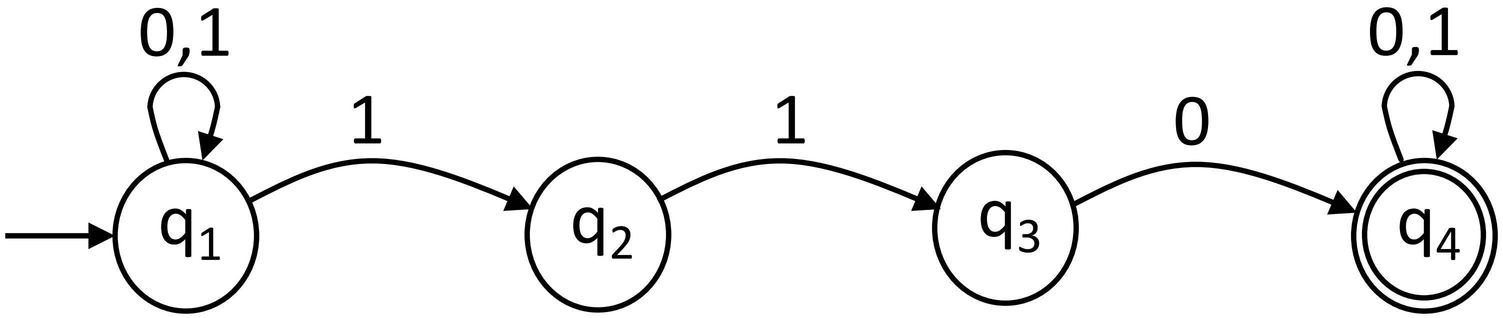
$\{\omega\colon \omega$ contains the substring $110\}$

$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. **If input symbol does not match any outgoing transitions, that branch dies.**
3. If any branch ends in an accept state, accept. If not, reject.



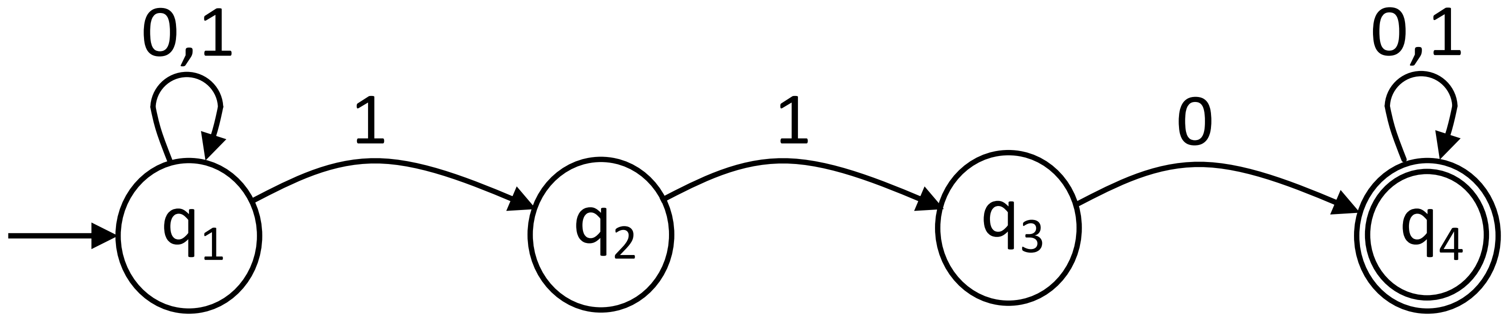


$\{\omega\colon\ \omega$ contains the substring $110\}$

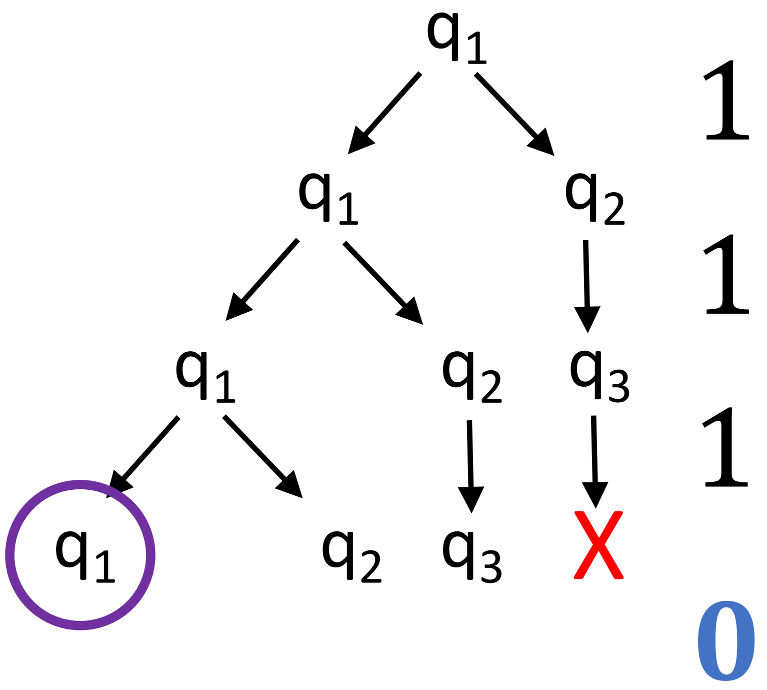
$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



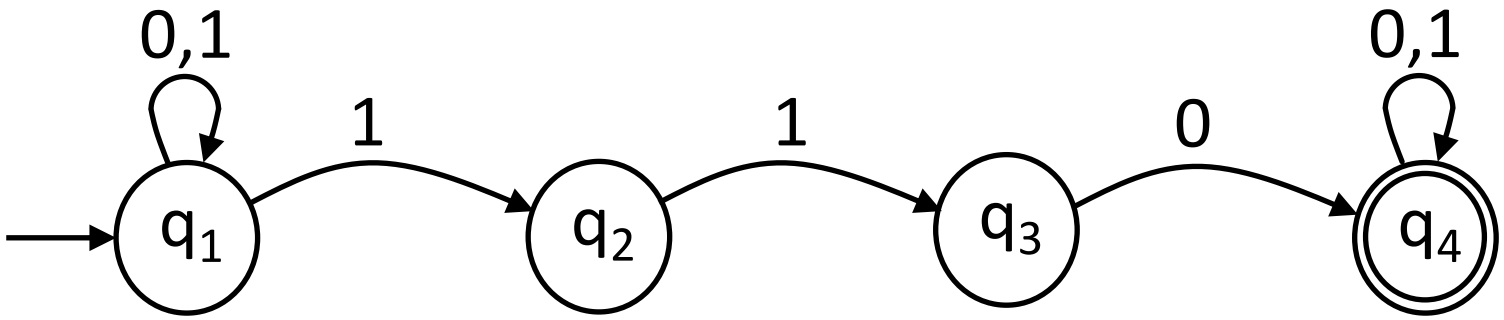
$\{\omega\colon\ \omega$ contains the substring $110\}$



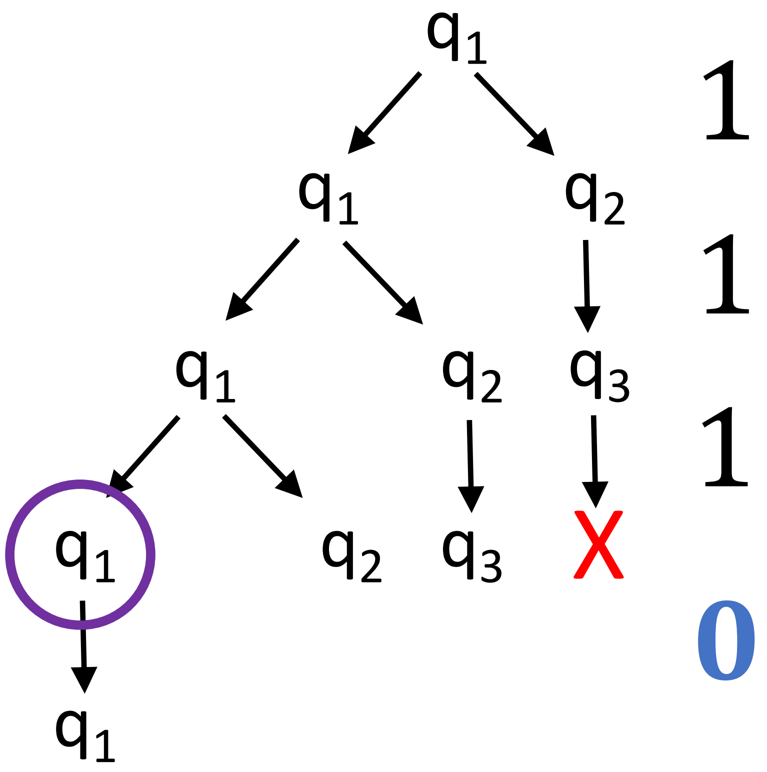
$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



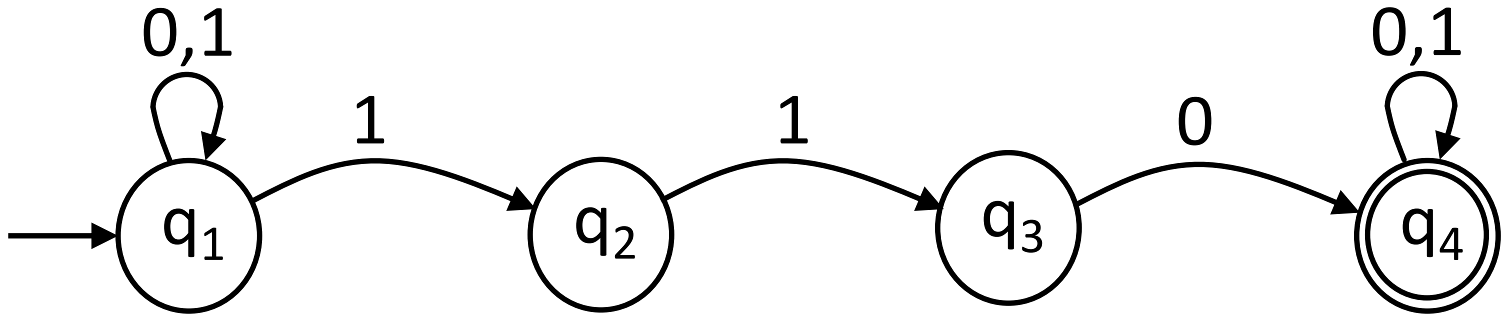
$\{\omega\colon\ \omega$ contains the substring $110\}$



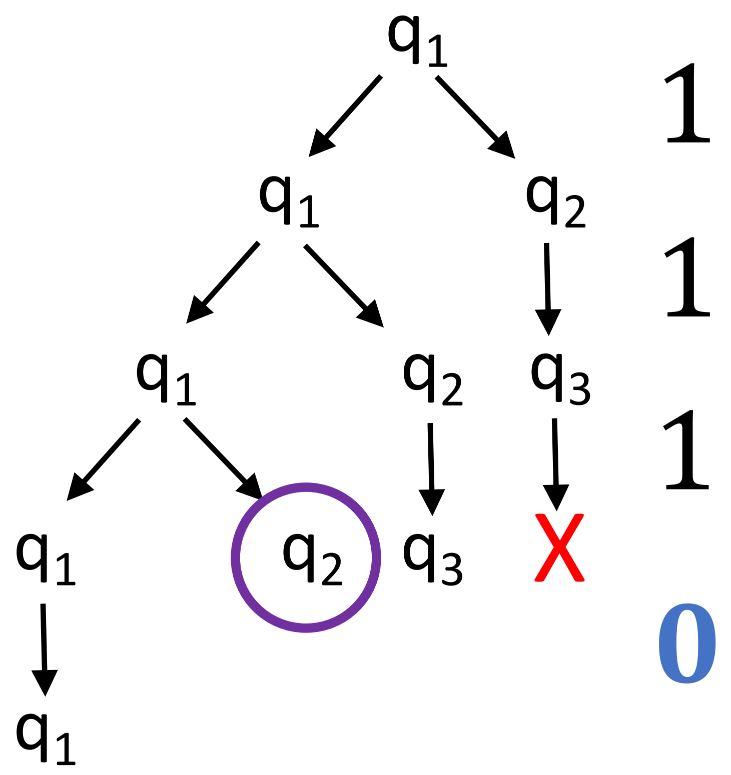
$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



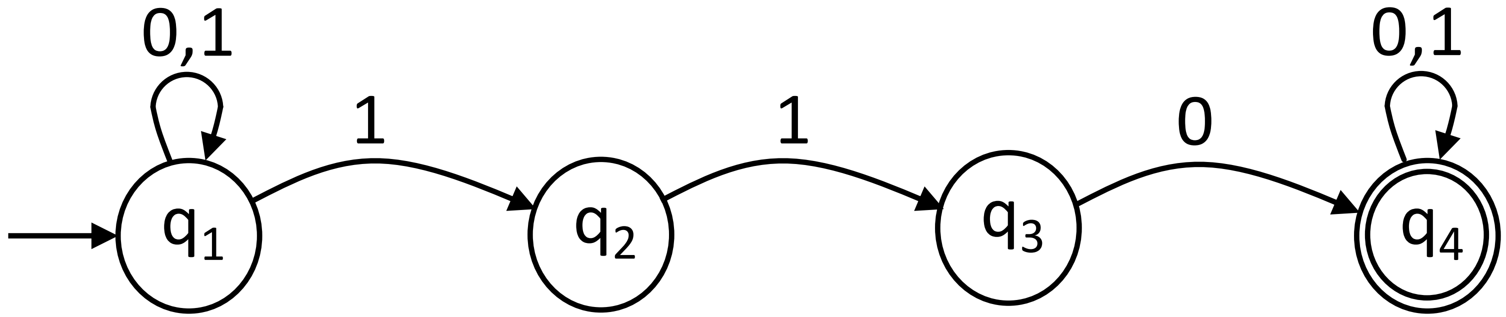
$\{\omega\colon\ \omega$ contains the substring $110\}$



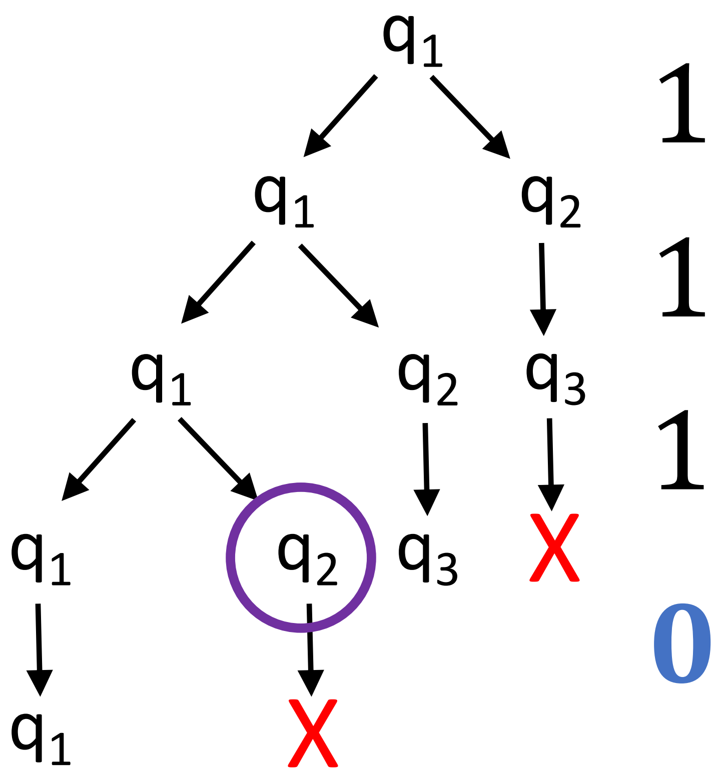
$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



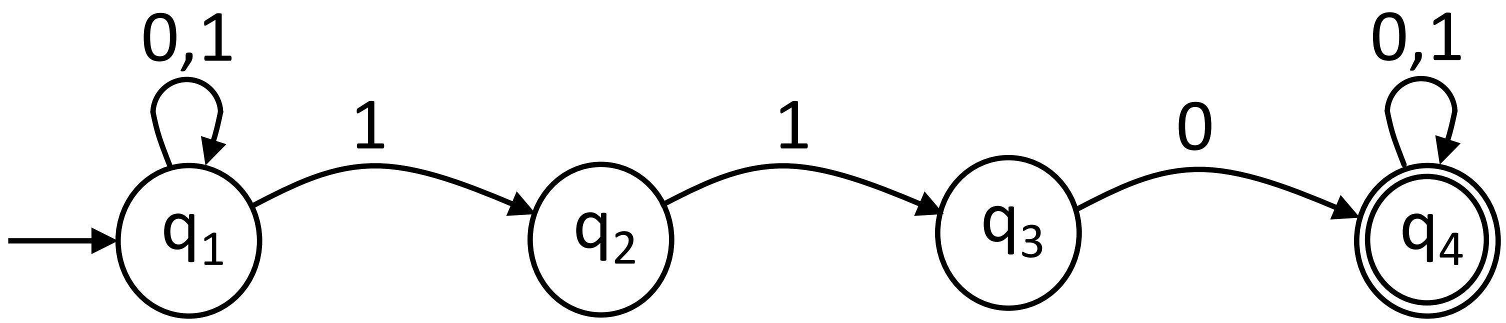
$\{\omega\colon\ \omega$ contains the substring $110\}$



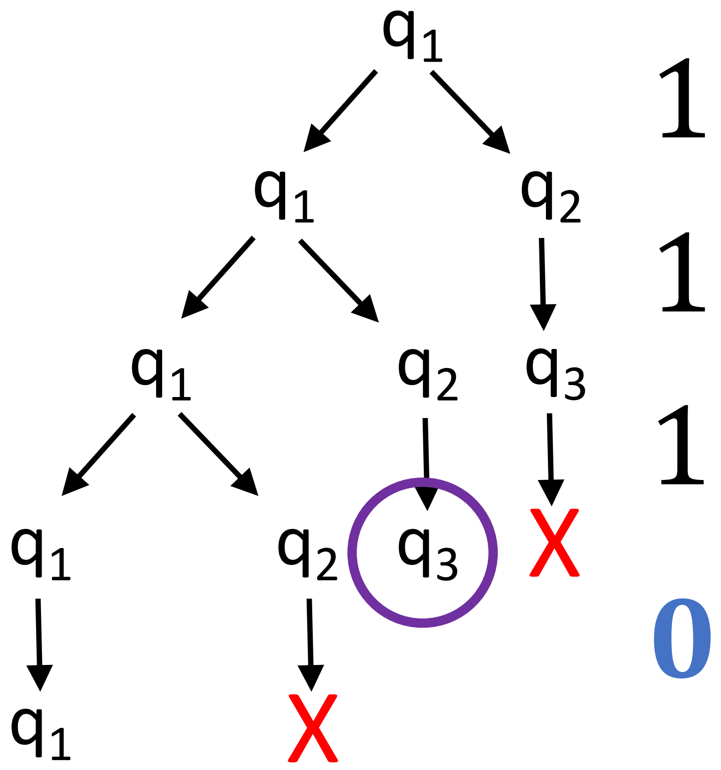
$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



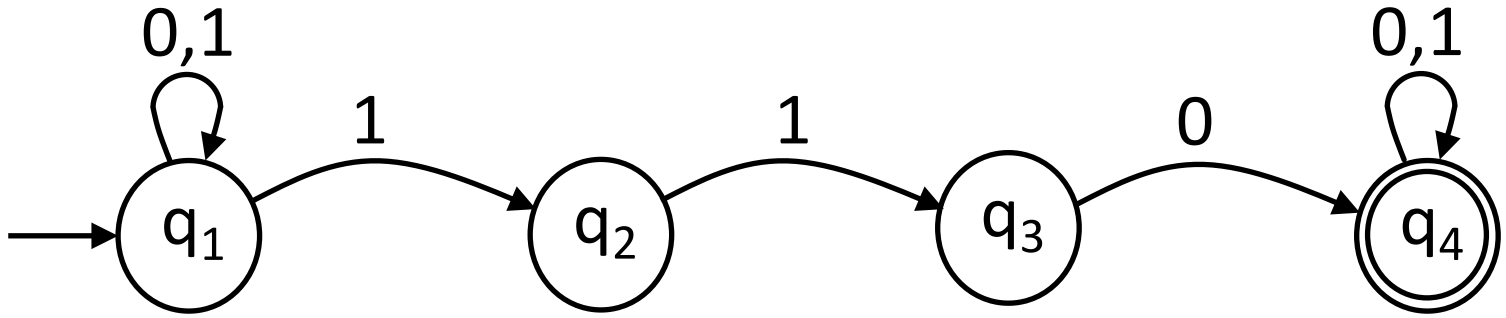
$\{\omega\colon\ \omega$ contains the substring $110\}$



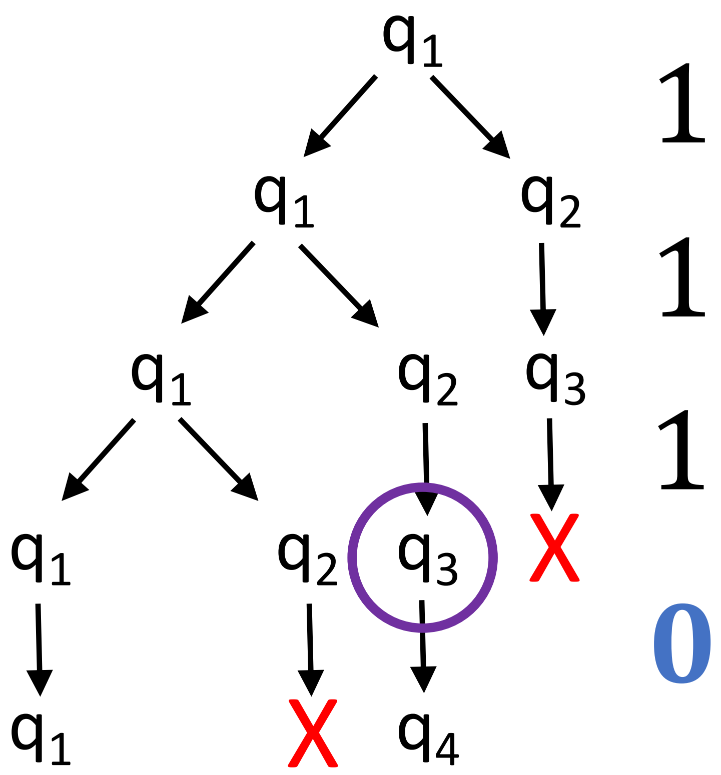
$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



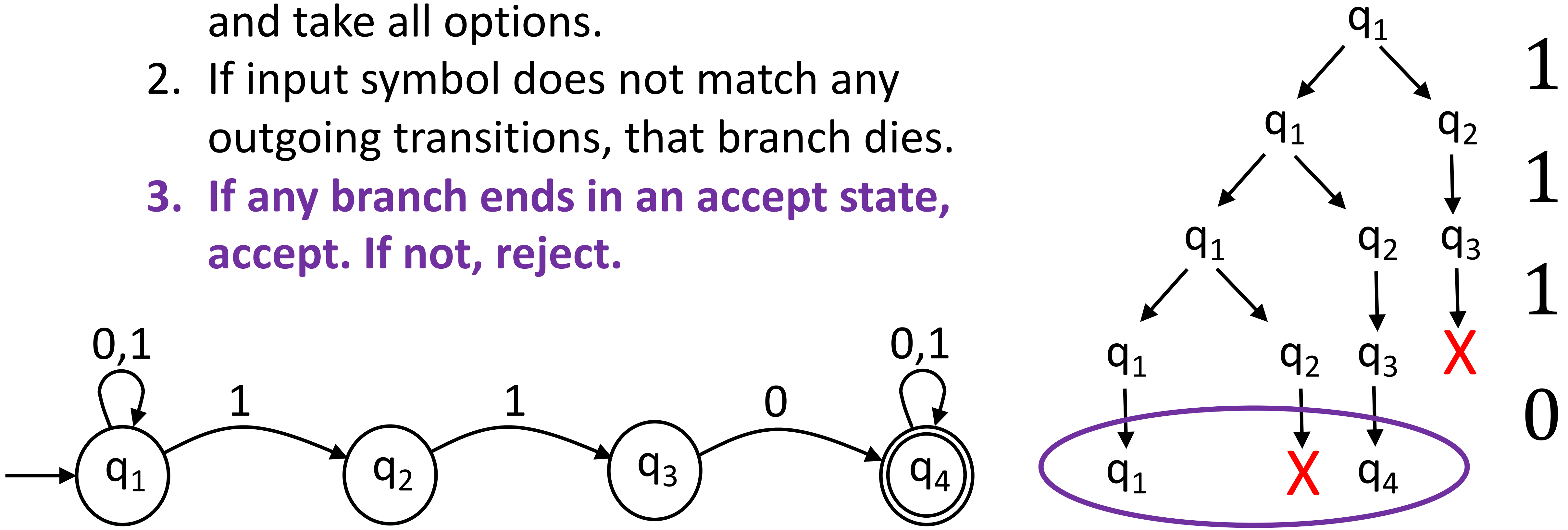
$\{\omega\colon\ \omega$ contains the substring $110\}$



$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. **If any branch ends in an accept state, accept. If not, reject.**



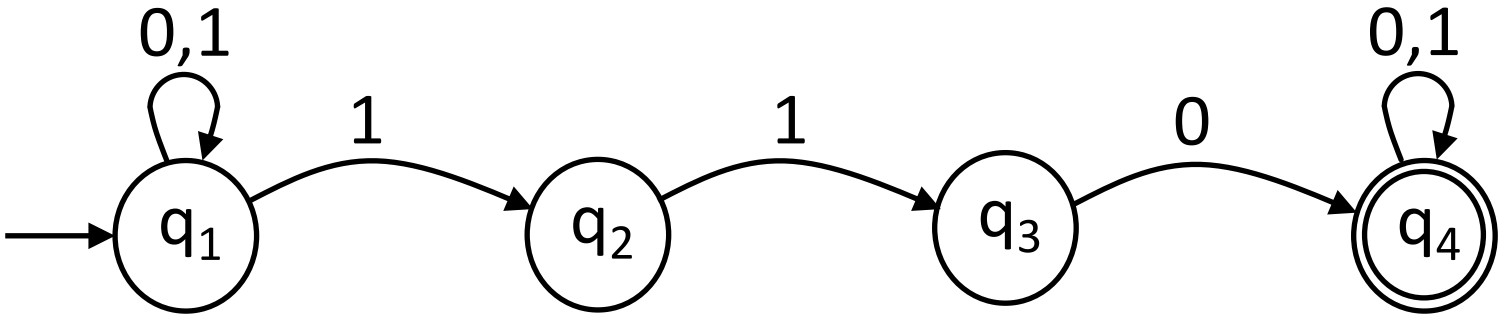
{ω: ω contains the substring 110}

$\omega = 1110$

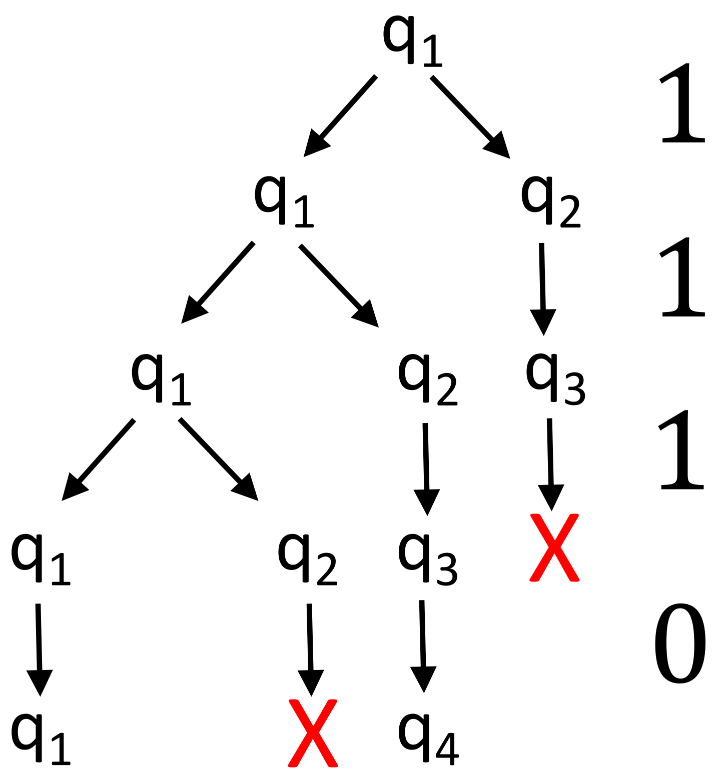
NFA Processing

1. If a "decision" is encountered, split
   ar
2. If
   ou
3. If
   accept. If not, reject.

Alternate approach: If it is possible to end on an accept state, accept.



$\{\omega\colon\ \omega$ contains the substring $110\}$

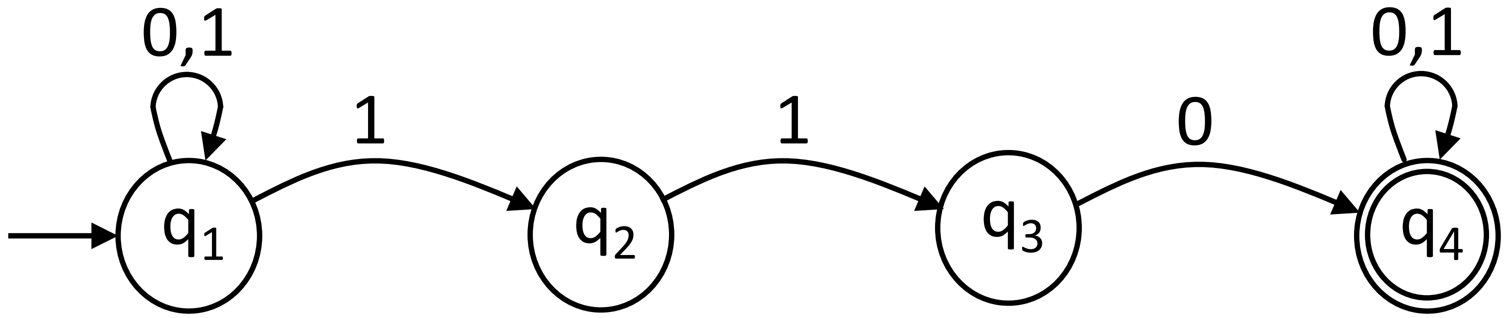


$\omega = 1110$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.

?

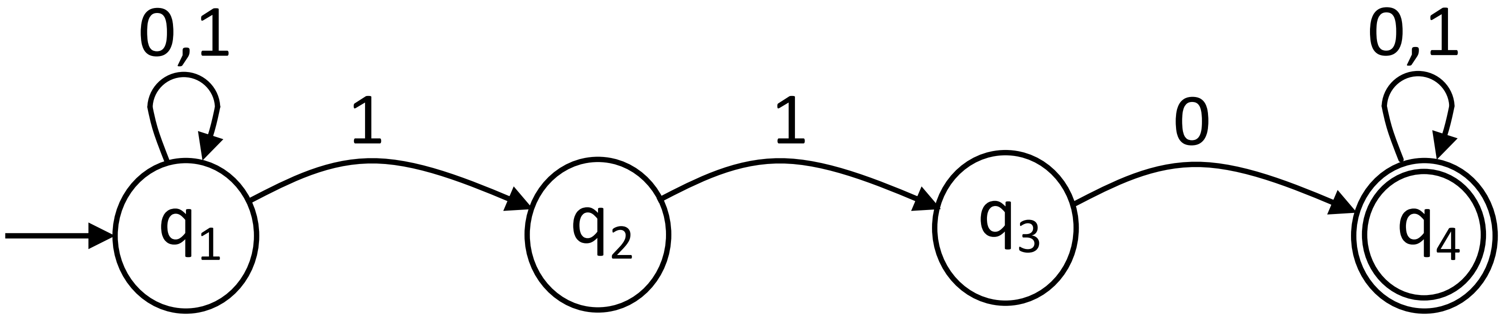


$\{\omega\text{: } \omega$ contains the substring $110\}$

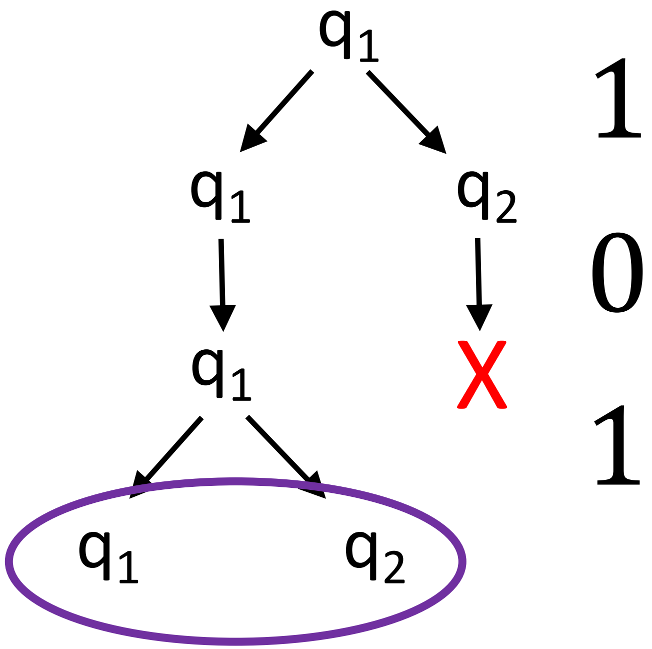
$$\omega = 101$$

NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.



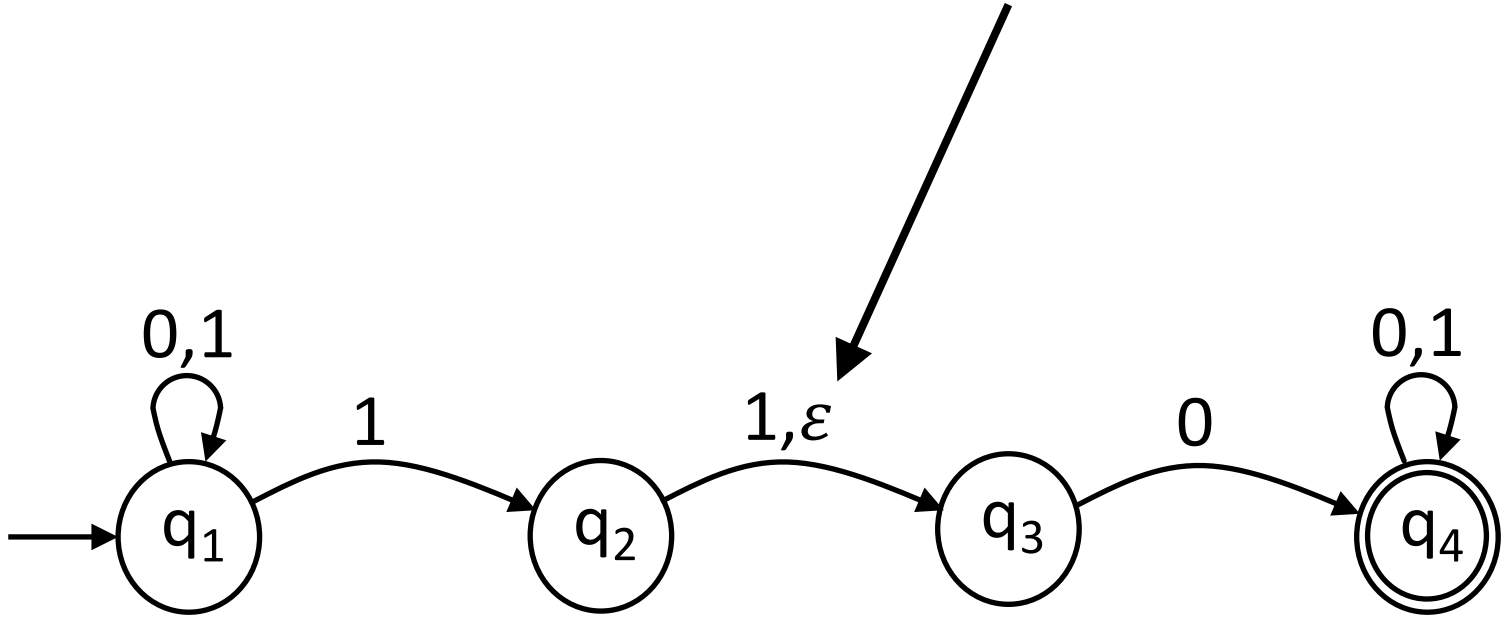
$\{\omega\!: \ \omega$ contains the substring $110\}$



$\omega = 101$

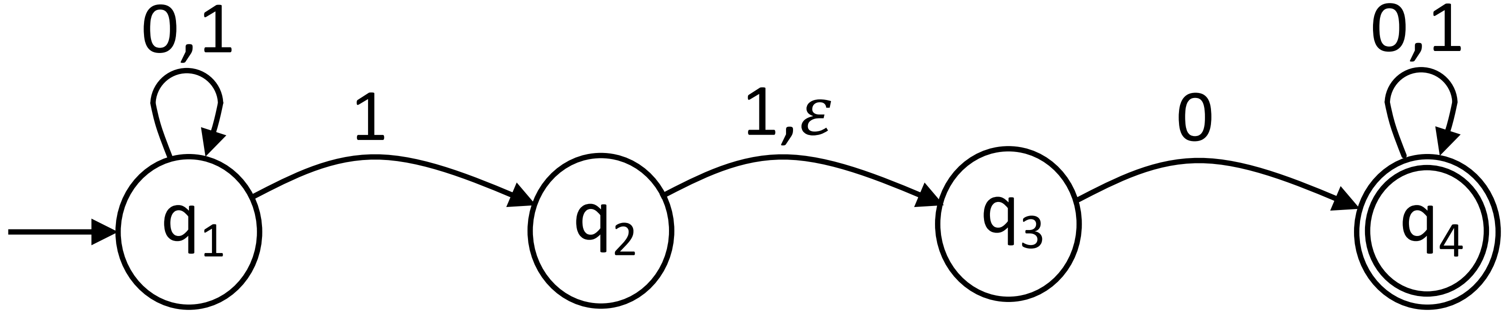
ε-Transitions

ε-transitions can be taken without consuming any characters from the string being processed.



ε-Transitions

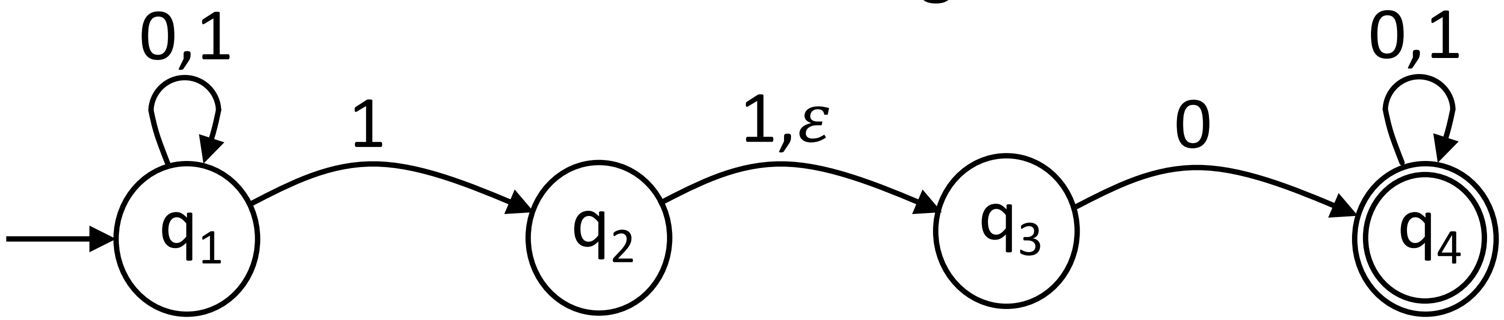
1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. **If ε is encountered, split and take all options without consuming a character from string.**



ε-Transitions

q_1

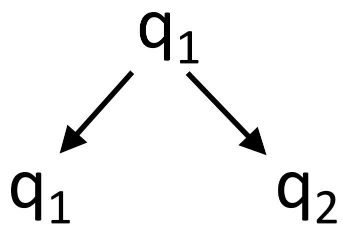
1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If ε is encountered, split and take all options without consuming a character from string.



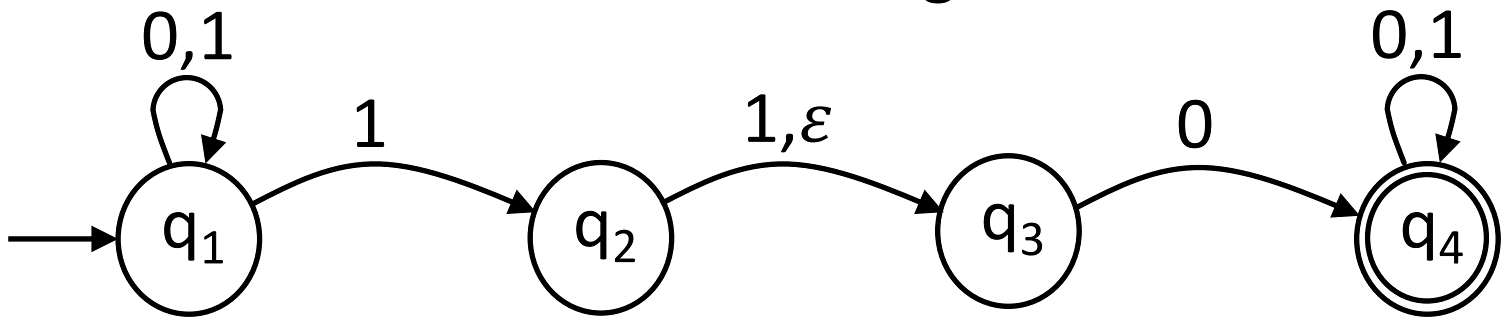
$$\omega = 101$$

ε-Transitions

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If ε is encountered, split and take all options without consuming a character from string.

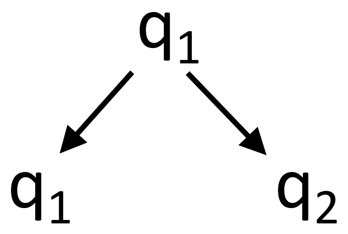


1



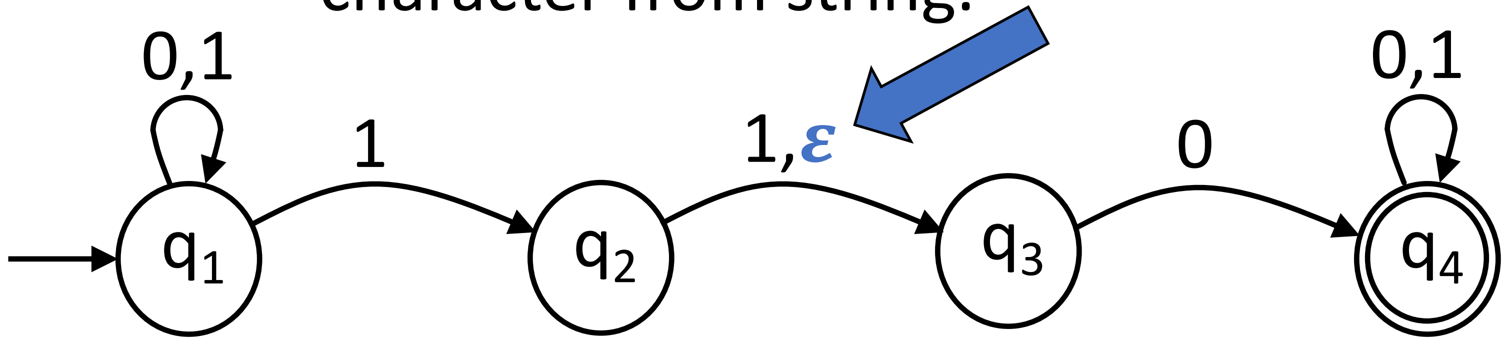
$\omega = 101$

ε-Transitions



1

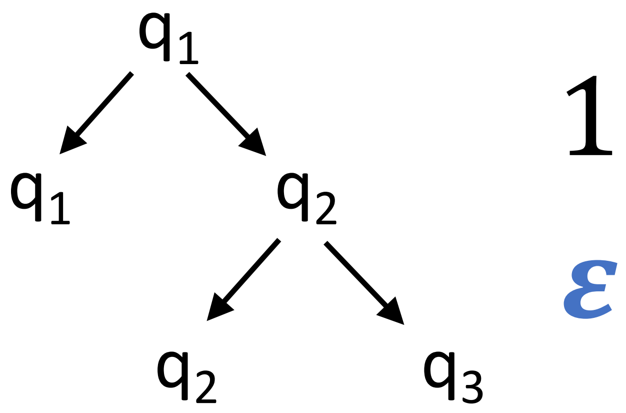
1. If a "decision" is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If ε is encountered, split and take all options without consuming a character from string.

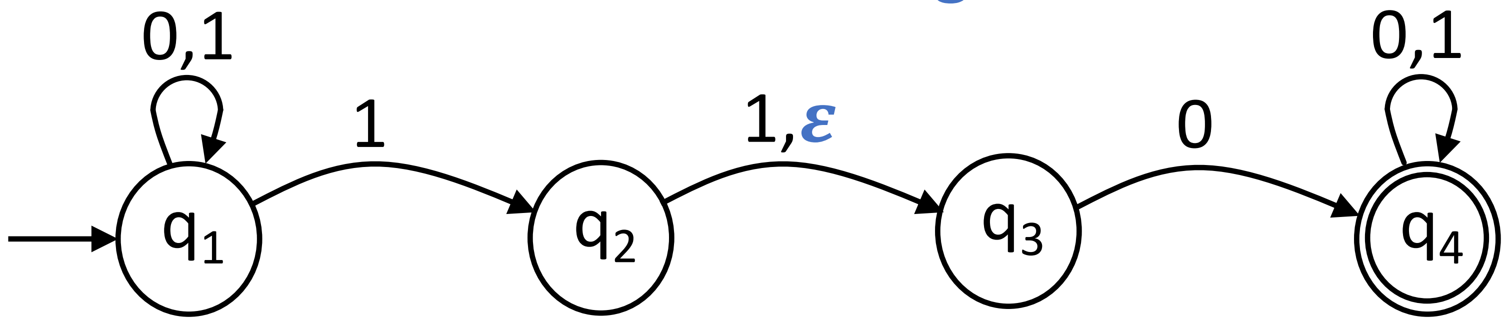


$$\omega = 101$$

ε-Transitions

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. **If ε is encountered, split and take all options without consuming a character from string.**

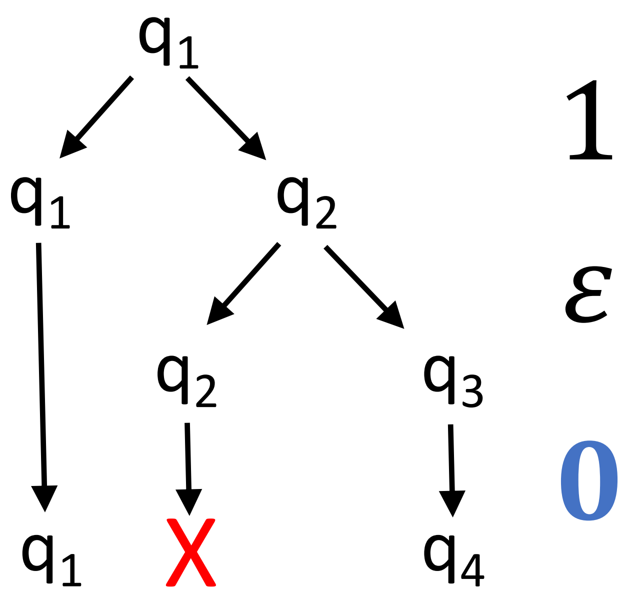


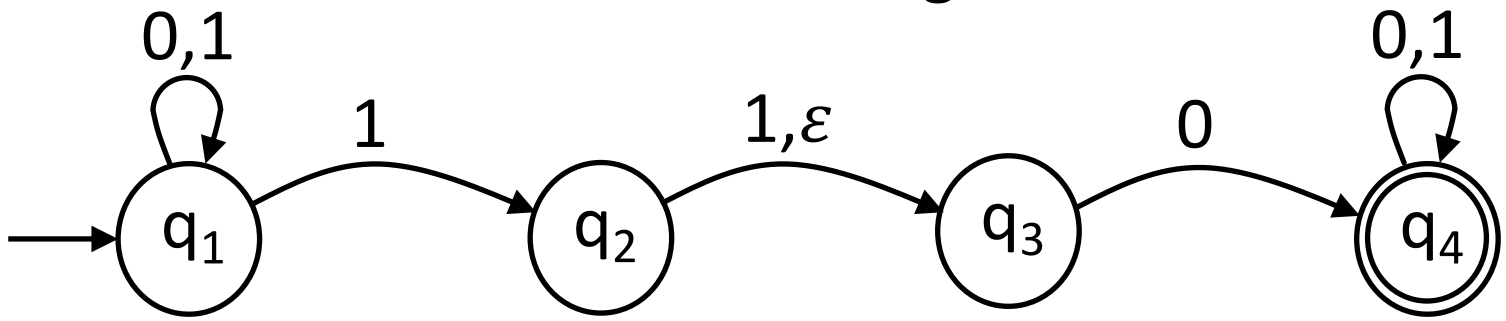


$$\omega = 101$$

ε-Transitions

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If ε is encountered, split and take all options without consuming a character from string.

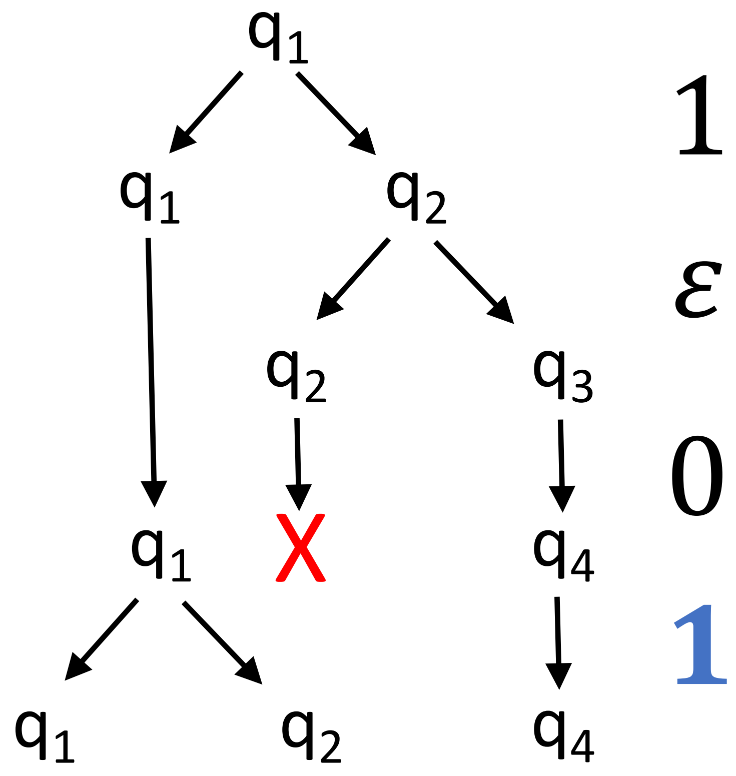


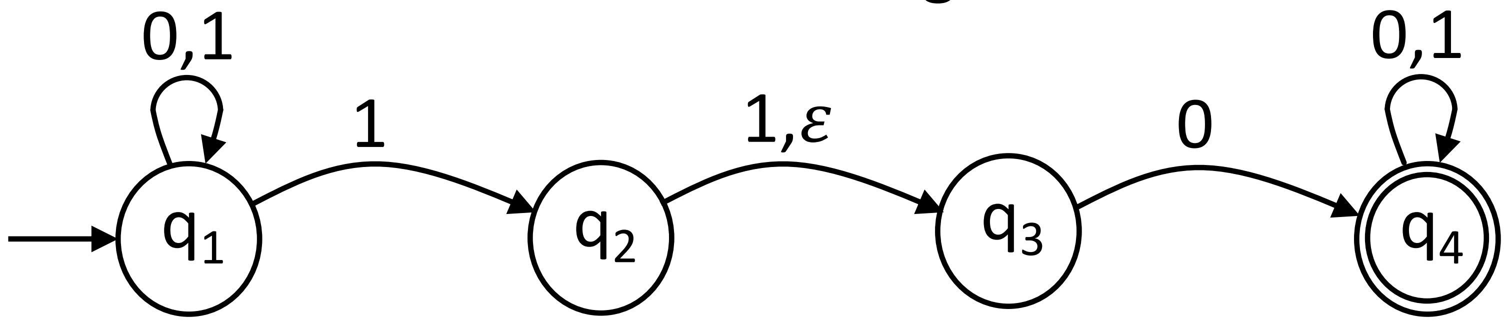


$$\omega = 101$$

ε-Transitions

1. If a "decision" is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If ε is encountered, split and take all options without consuming a character from string.

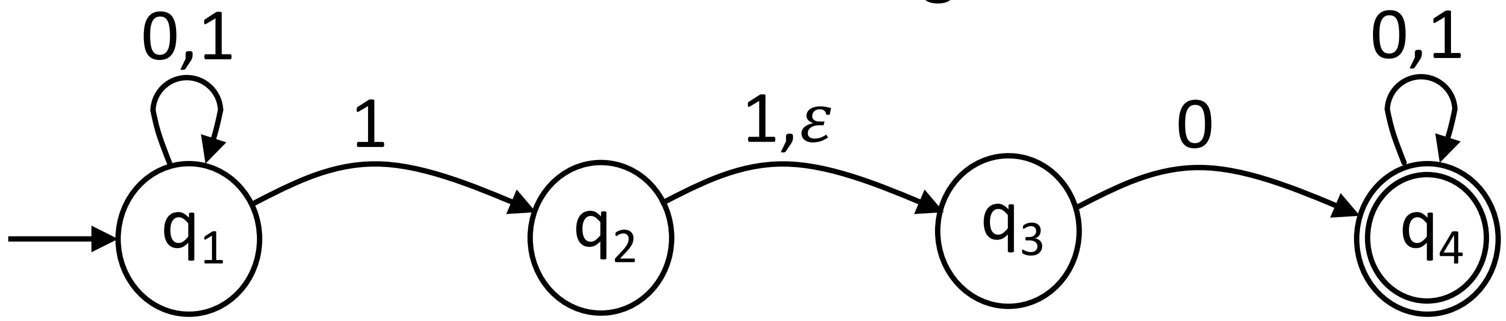


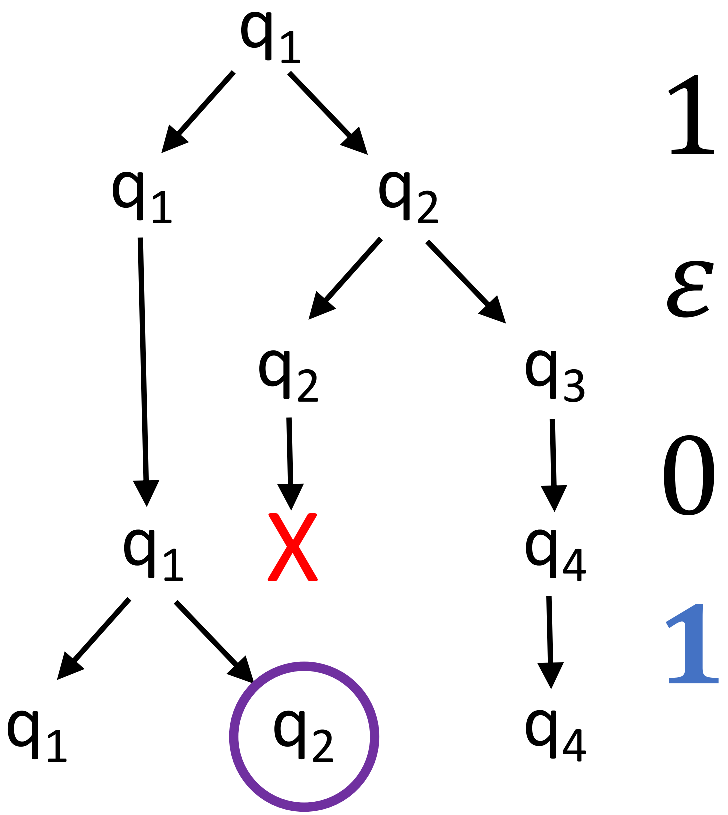


$\omega = 101$

ε-Transitions

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If ε is encountered, split and take all options without consuming a character from string.

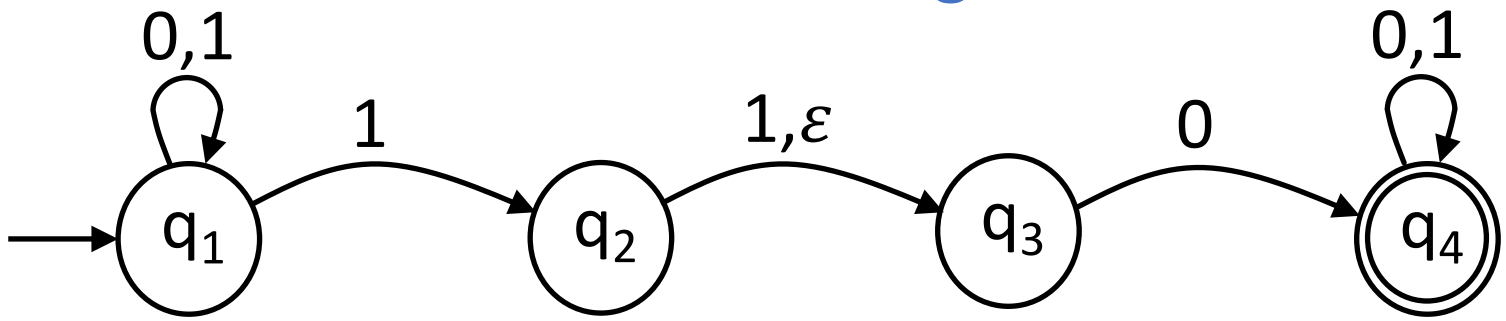


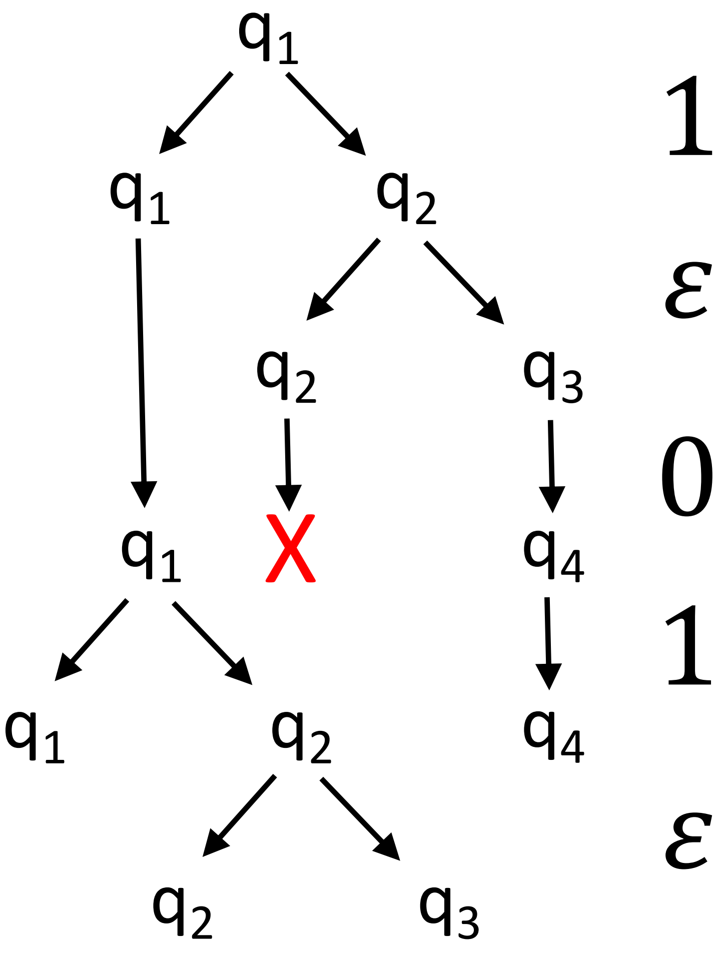


$$\omega = 101$$

ε-Transitions

1. If a "decision" is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. **If ε is encountered, split and take all options without consuming a character from string.**

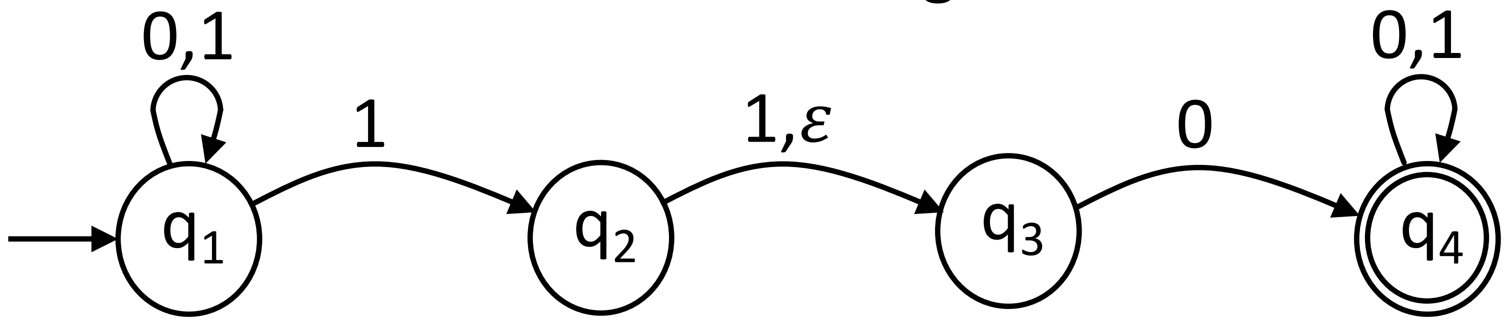


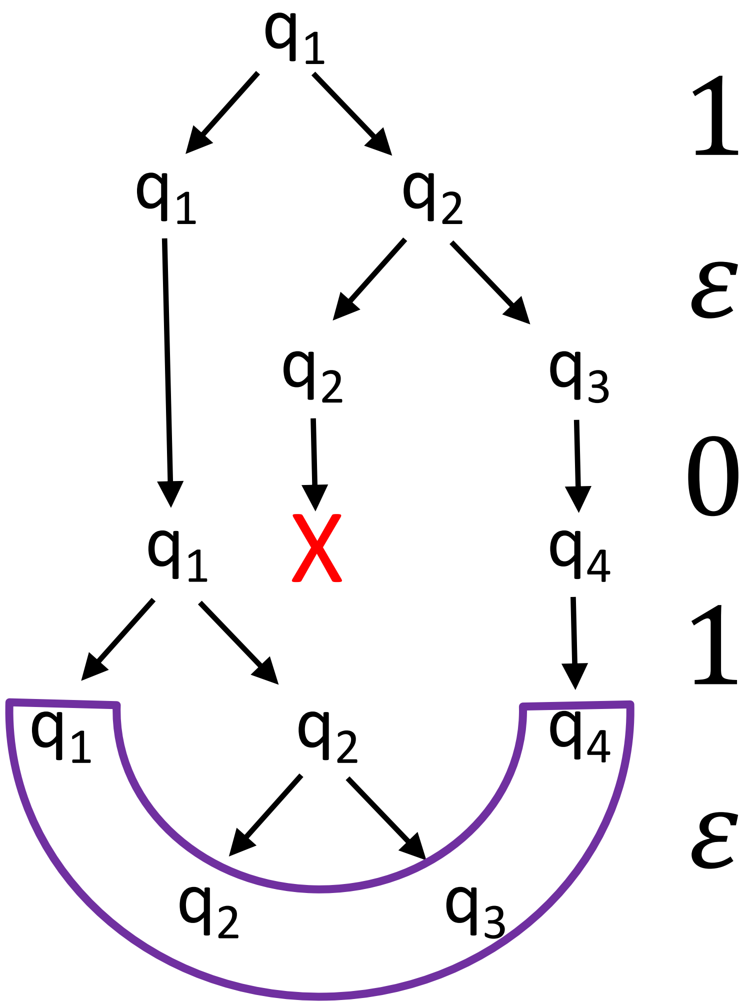


$$\omega = 101$$

ε-Transitions

1. If a "decision" is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If ε is encountered, split and take all options without consuming a character from string.

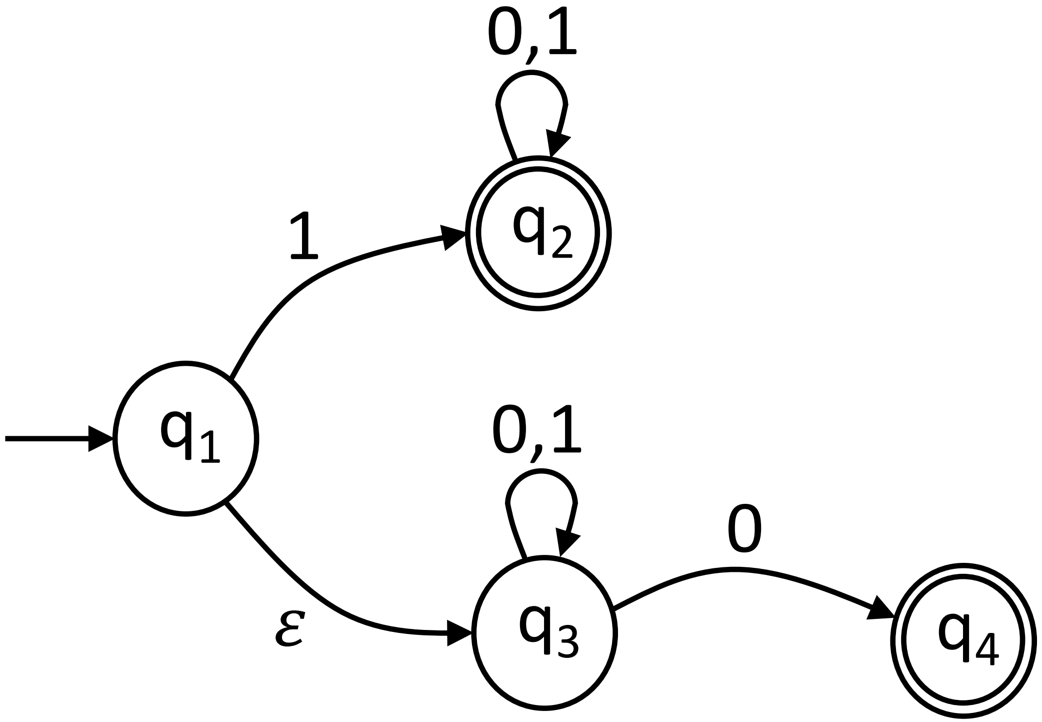




$$\omega = 101$$

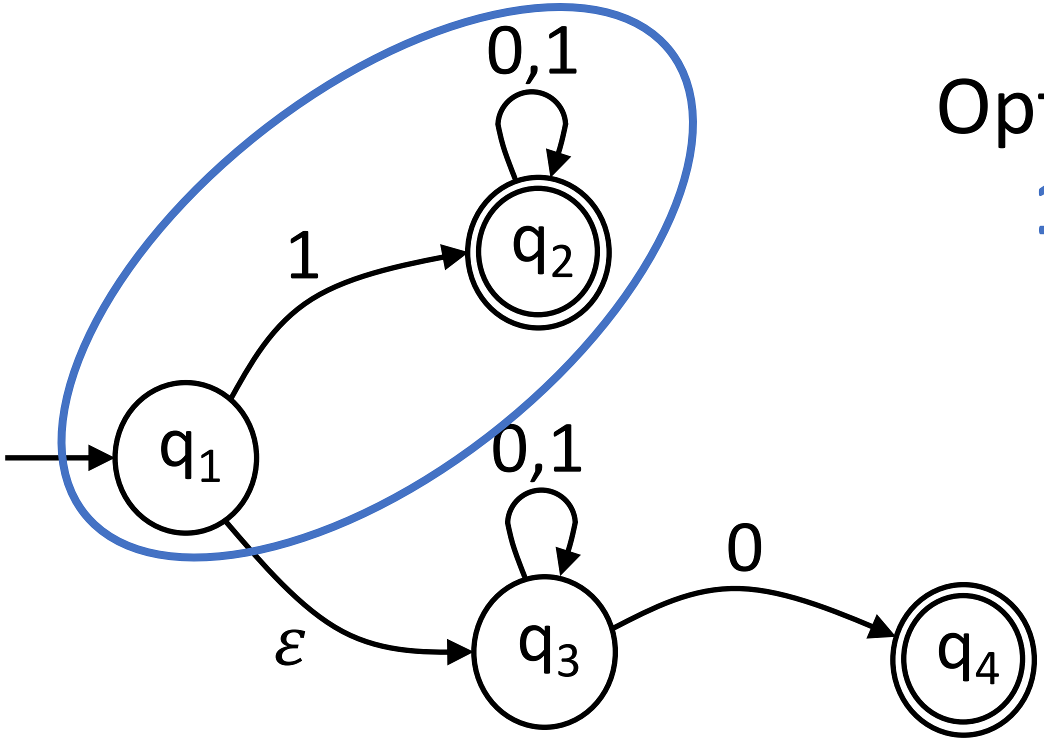
NFA Practice

What is the language of the following NFA?



NFA Practice

What is the language of the following NFA?

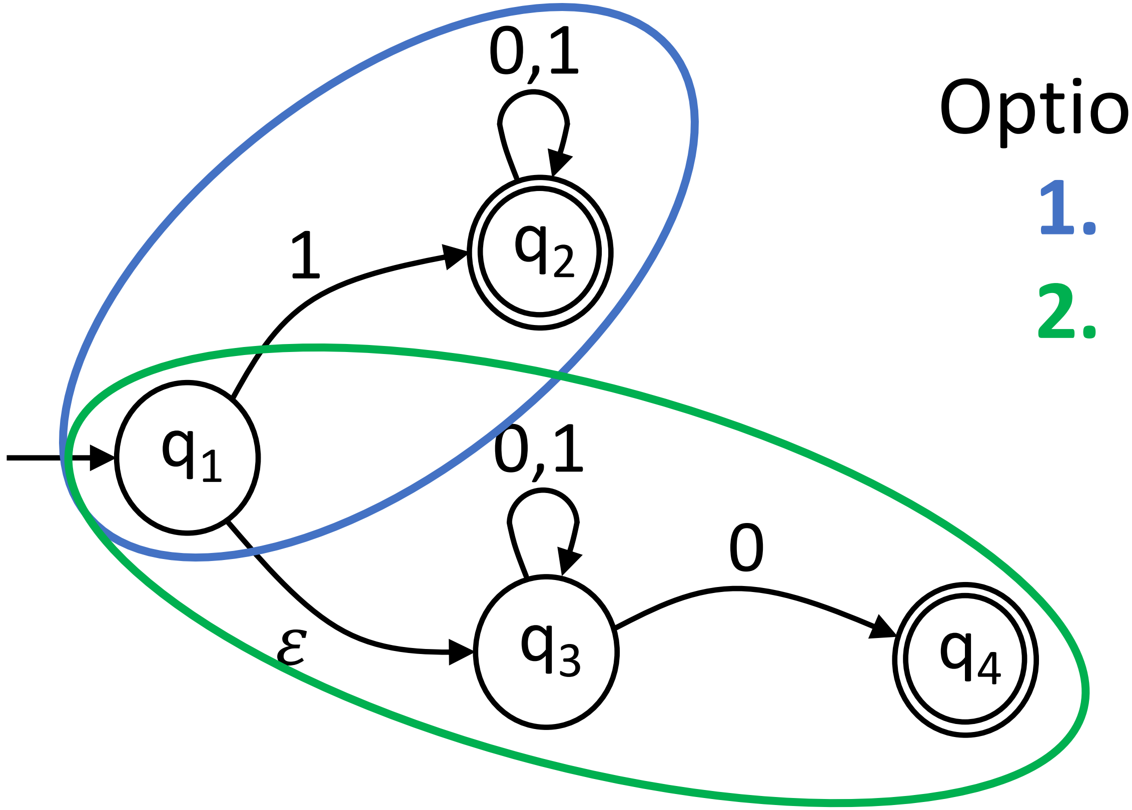


Options for string to be accepted:

1. **Start with 1, followed by anything.**

NFA Practice

What is the language of the following NFA?

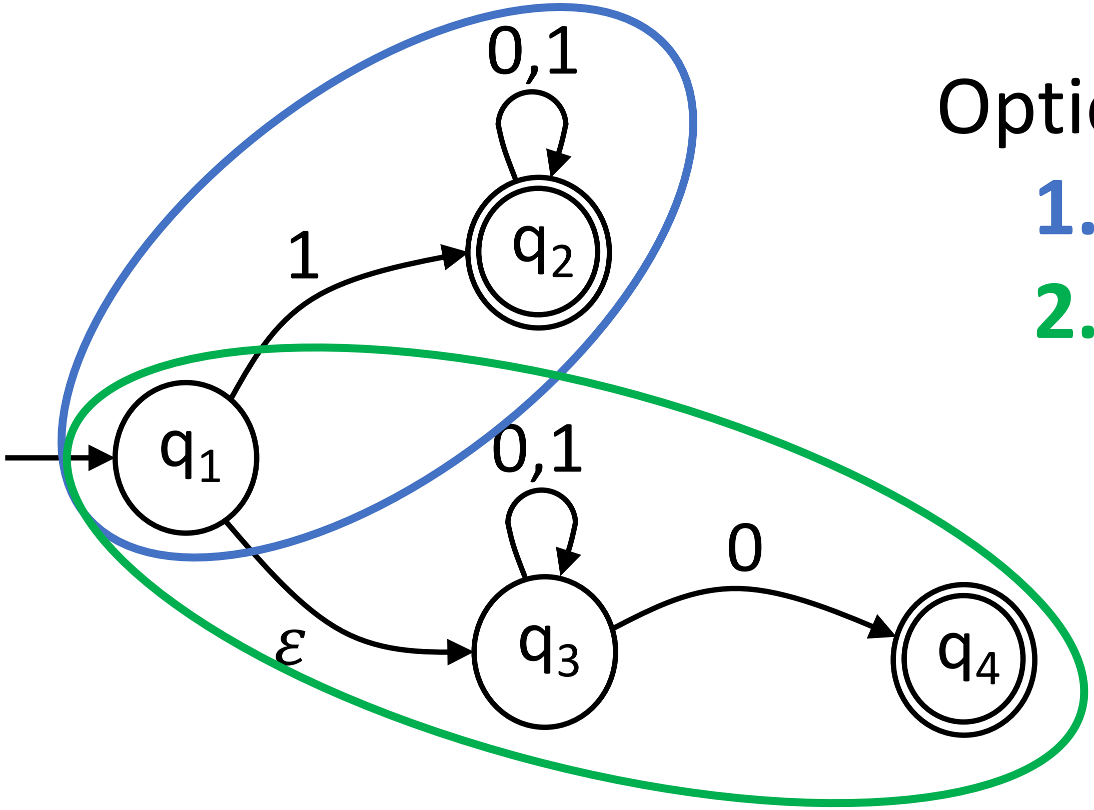


Options for string to be accepted:

1. **Start with 1, followed by anything.**
2. **Start with anything, ending with 0.**

NFA Practice

What is the language of the following NFA?



Options for string to be accepted:

1. **Start with 1, followed by anything.**
2. **Start with anything, ending with 0.**

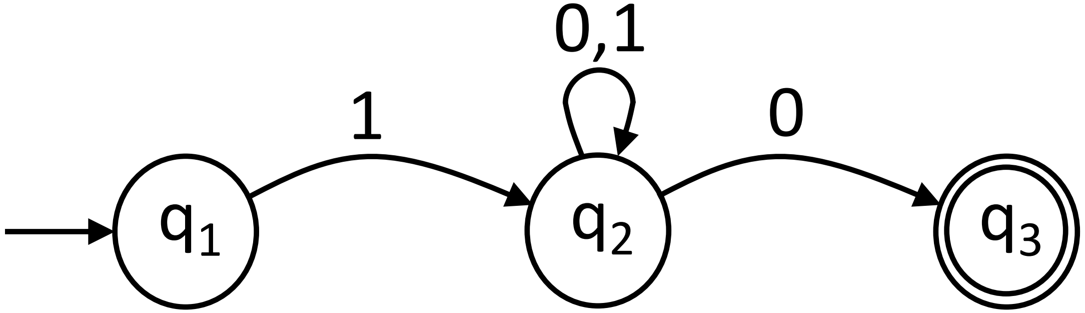
$\{\omega\colon\ \omega$ starts with 1 or ends with $0\}$

NFA Practice

What is the NFA that accepts $\{\omega\colon\ \omega$ starts with 1 **and** ends with $0\}$?

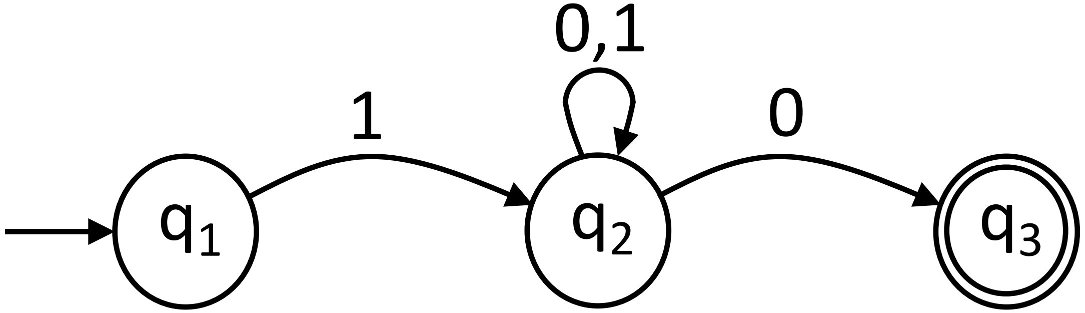
NFA Practice

What is the NFA that accepts $\{\omega\colon\ \omega$ starts with 1 **and** ends with $0\}$?



NFA Practice

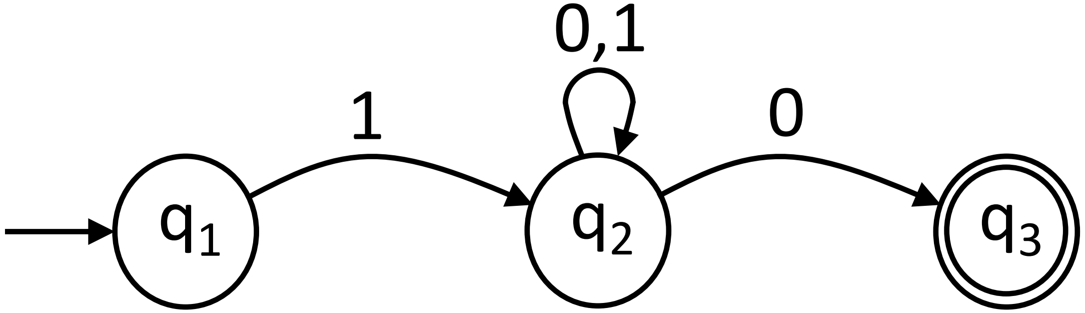
What is the NFA that accepts $\{\omega\colon\ \omega$ starts with 1 **and** ends with $0\}$?



Only ω's that start with 1 get to q_2.

NFA Practice

What is the NFA that accepts $\{\omega\colon\ \omega$ starts with 1 **and** ends with $0\}$?

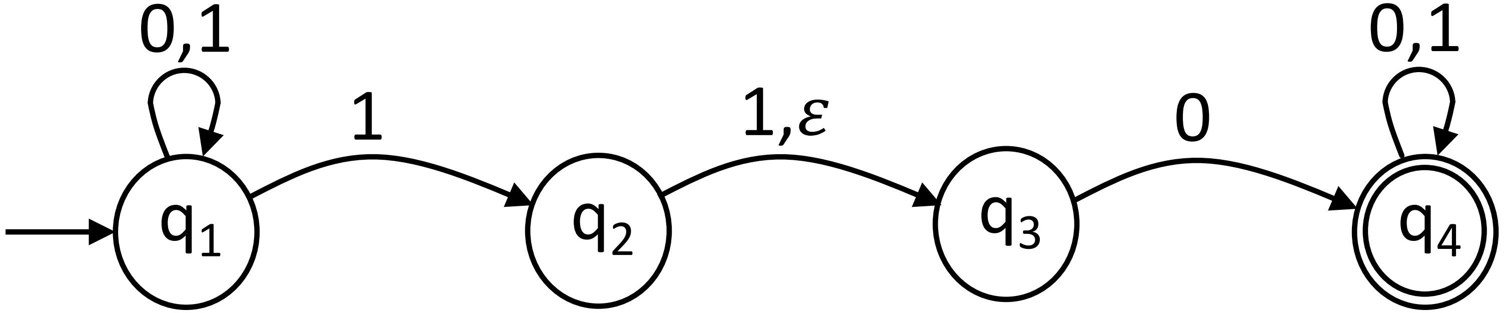


Only ω's that start with 1 get to q_2.
Any string that gets to q_2, can get to q_3 <u>and terminate</u>, if it ends with 0.

NFA Formal Definition

NFAs consist of:

1. Finite set of states, Q.
2. Finite alphabet, Σ.
3. Transition function, $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$.
4. Start state, $q_0 \in Q$.
5. Set of accept states, $F \subseteq Q$.



NFA Formal Definition

Power set of Q.

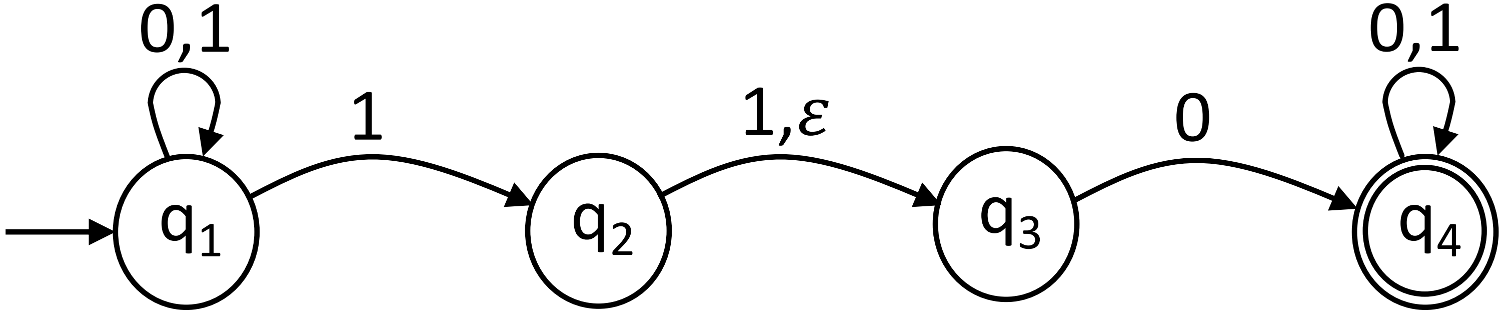
I.e. set of all subsets.

E.g. $Q = \{q_1, q_2\}$

$\Rightarrow \mathcal{P}(Q) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\}$

NFAs consist of:

1. Finite set of states, Q.
2. Finite alphabet, Σ.
3. Transition function, $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$.
4. Start state, $q_0 \in Q$.
5. Set of accept states, $F \subseteq Q$.



NFA Formal Definition

Power set of Q.

I.e. set of all subsets.

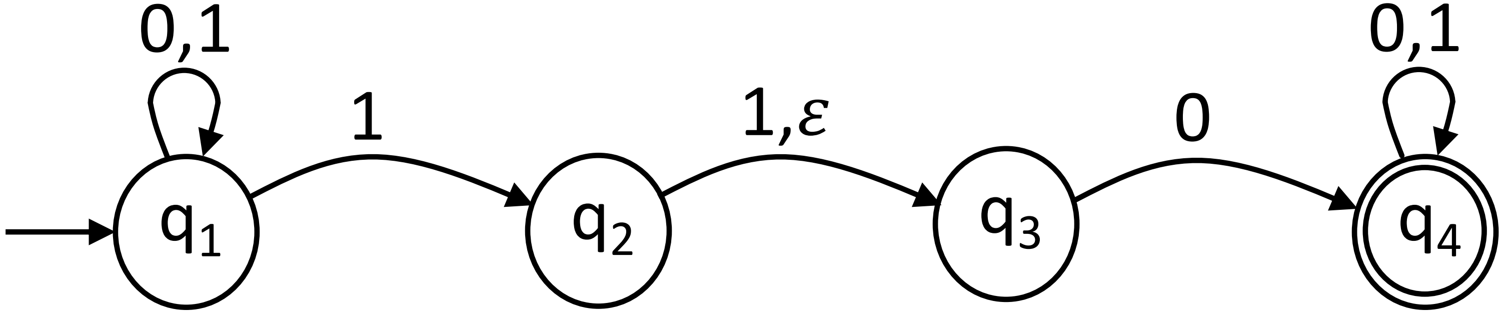
E.g. $Q = \{q_1, q_2\}$

$\Rightarrow \mathcal{P}(Q) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\}$

I.e. $\exists$ 0 or more transitions for each $e \in \Sigma \cup \{\varepsilon\}$ at each state

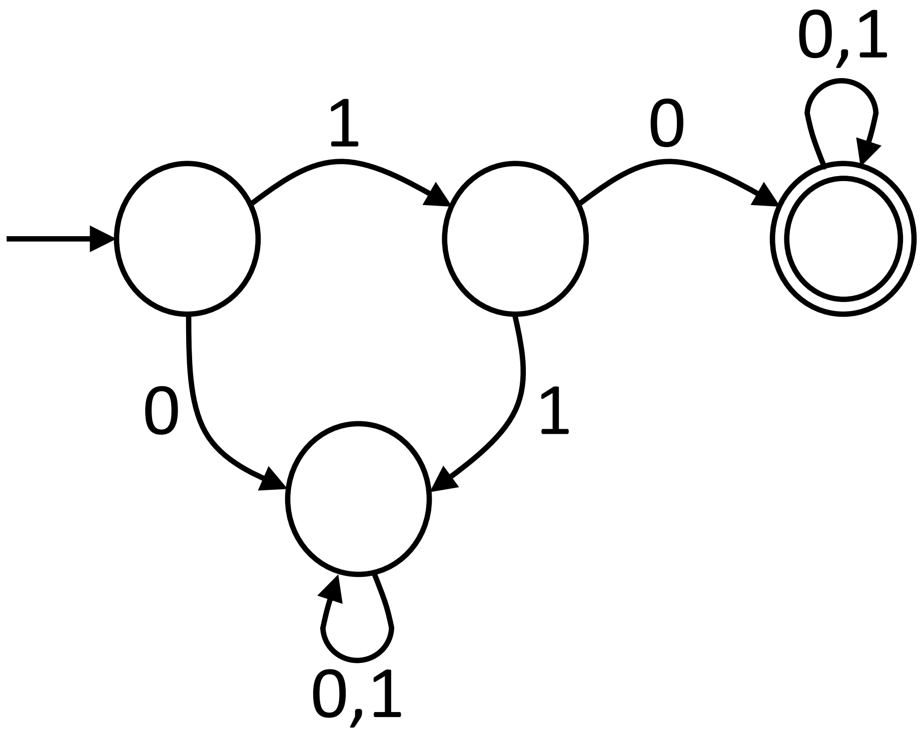
NFAs consist of:

1. Finite set of states, Q.
2. Finite alphabet, Σ.
3. Transition function, $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$.
4. Start state, $q_0 \in Q$.
5. Set of accept states, $F \subseteq Q$.



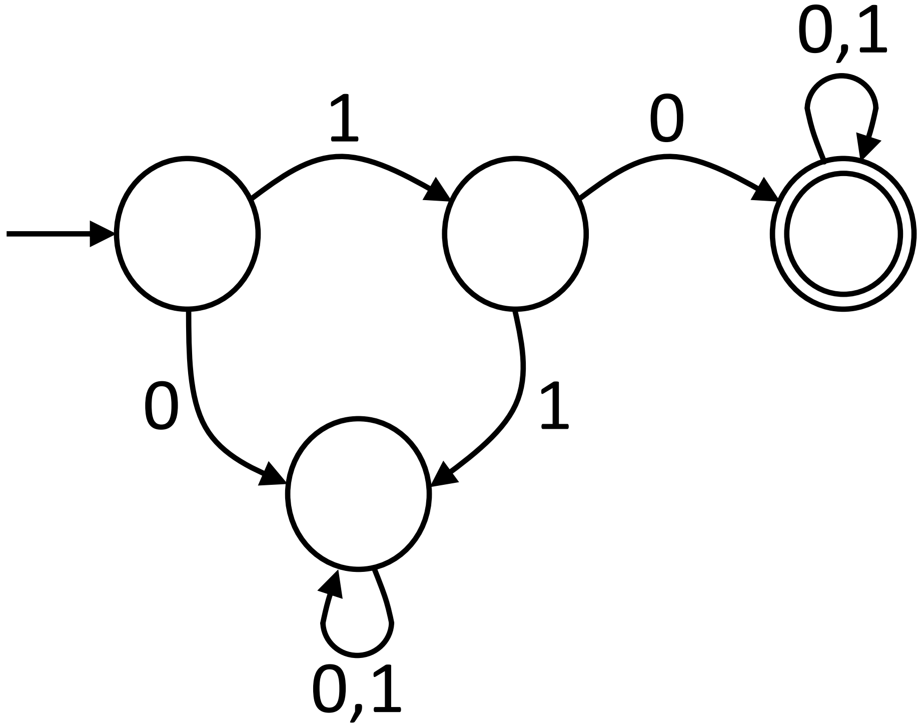
Build an NFA for the following language:

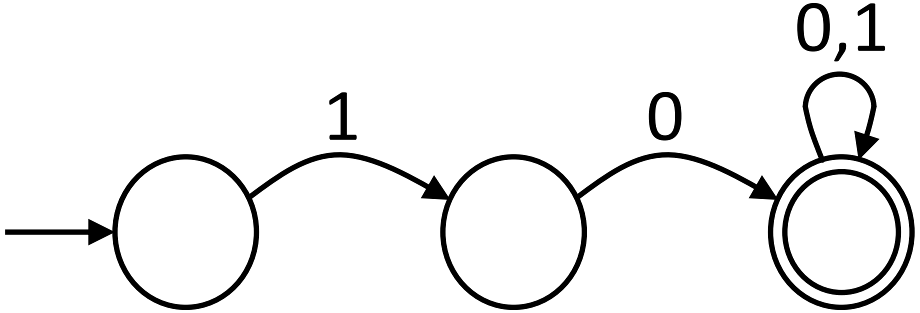
$\{\omega : \omega$ begins with sequence $10\}$.



Build an NFA for the following language:

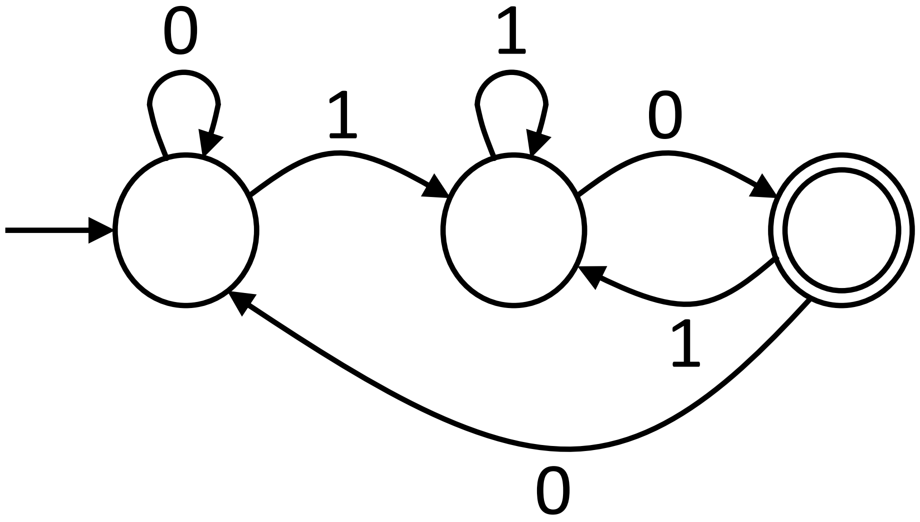
$\{\omega\colon \omega \text{ begins with sequence } 10\}$.





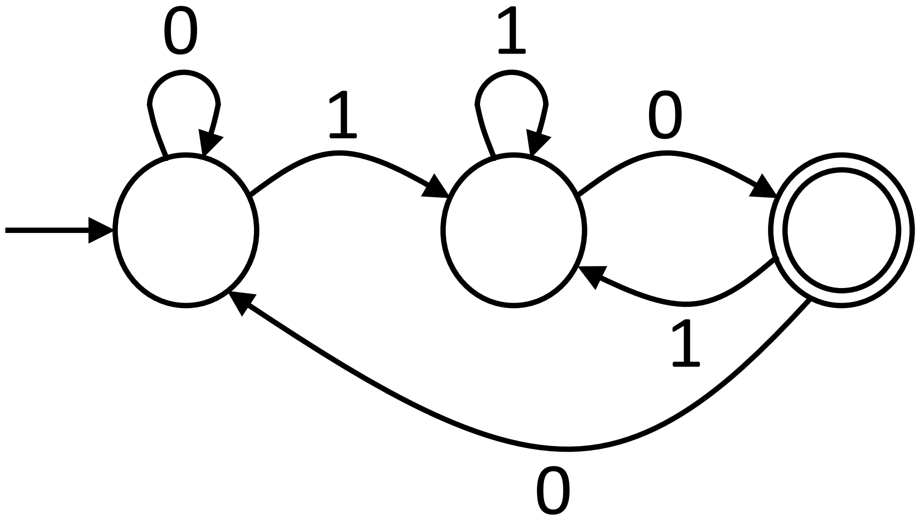
Build an NFA for the following language:

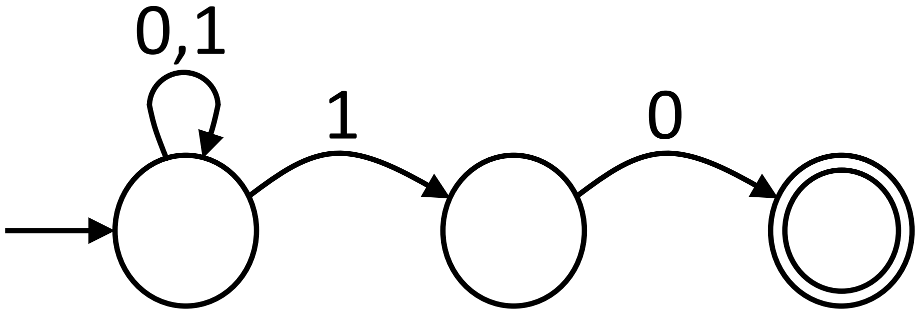
$\{\omega \colon \omega$ ends with sequence $10\}$.



Build an NFA for the following language:

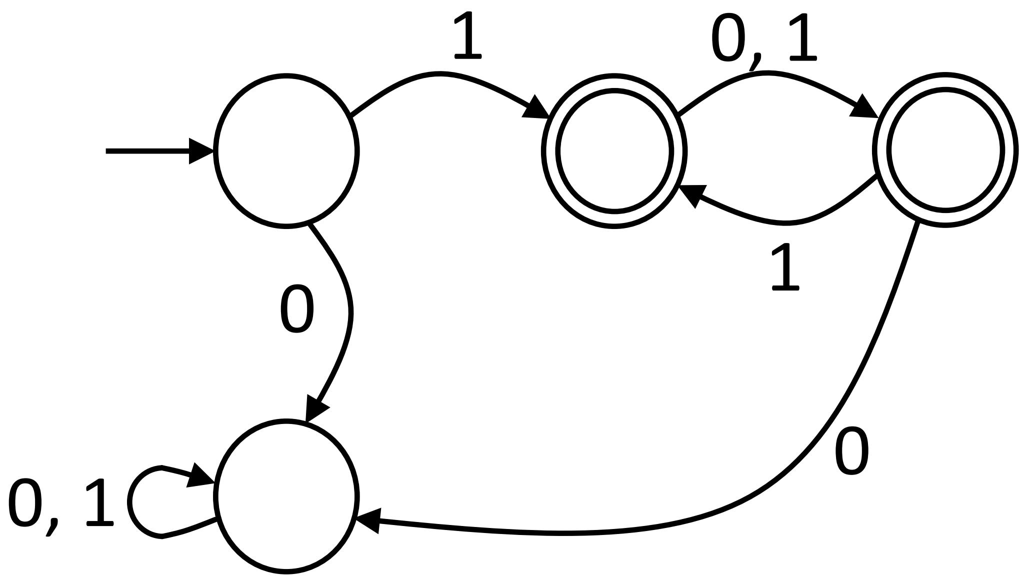
$\{\omega : \omega \text{ ends with sequence } 10\}$.





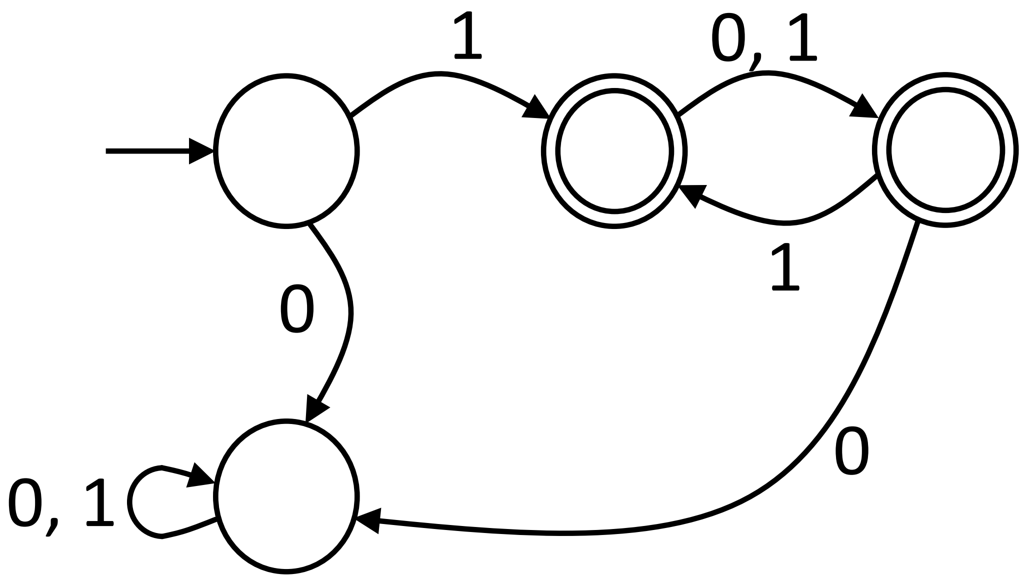
Build an NFA for the following language:

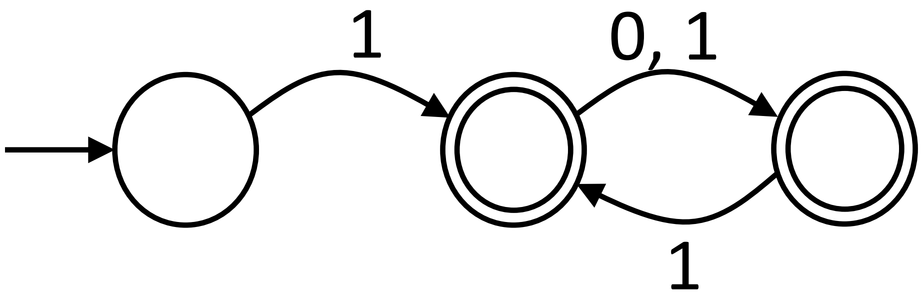
$\{\omega$: every odd symbol is a $1\}$.



Build an NFA for the following language:

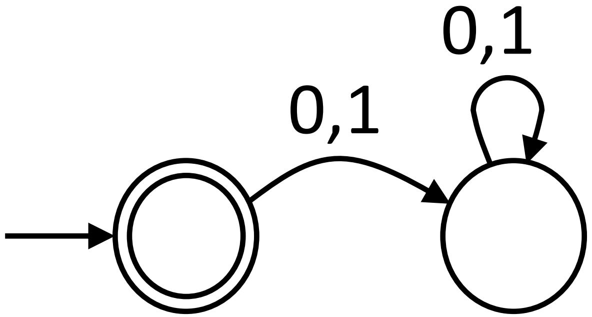
$\{\omega$: every odd symbol is a $1\}$.





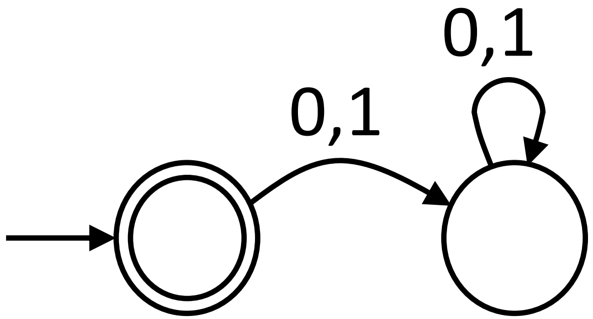
Build an NFA for the following language:

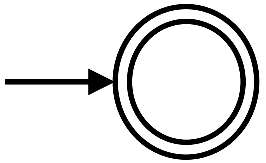
$\{\varepsilon\}$.



Build an NFA for the following language:

$\{\varepsilon\}$.



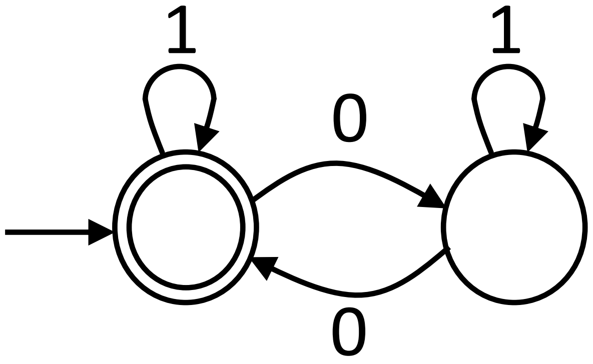


Build an NFA for the following language:

$\{\omega : \omega$ contains an even number of 0’s$\}$.

Build an NFA for the following language:

$\{\omega : \omega \text{ contains an even number of } 0\text{'s}\}$.

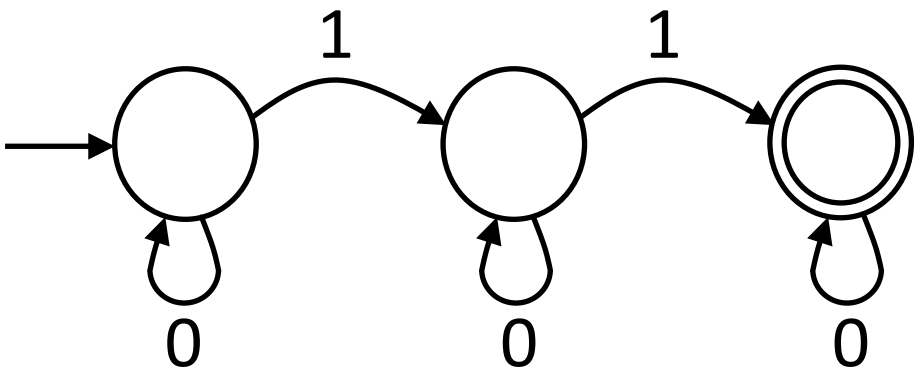


Build an NFA for the following language:

$\{\omega\colon \omega$ contains exactly two 1's$\}$.

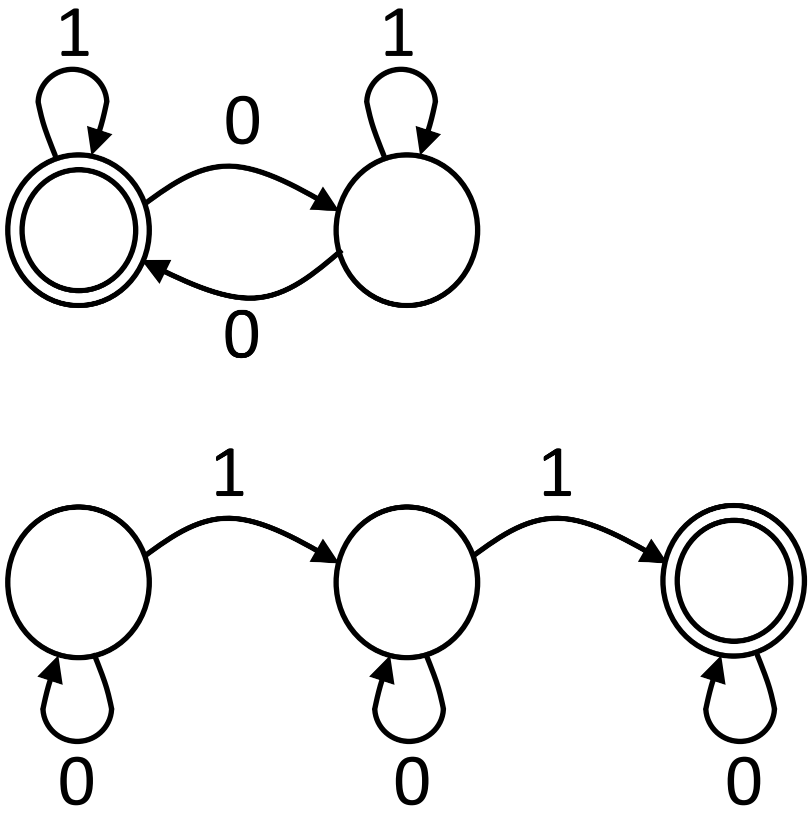
Build an NFA for the following language:

$\{\omega: \omega$ contains exactly two 1's$\}$.



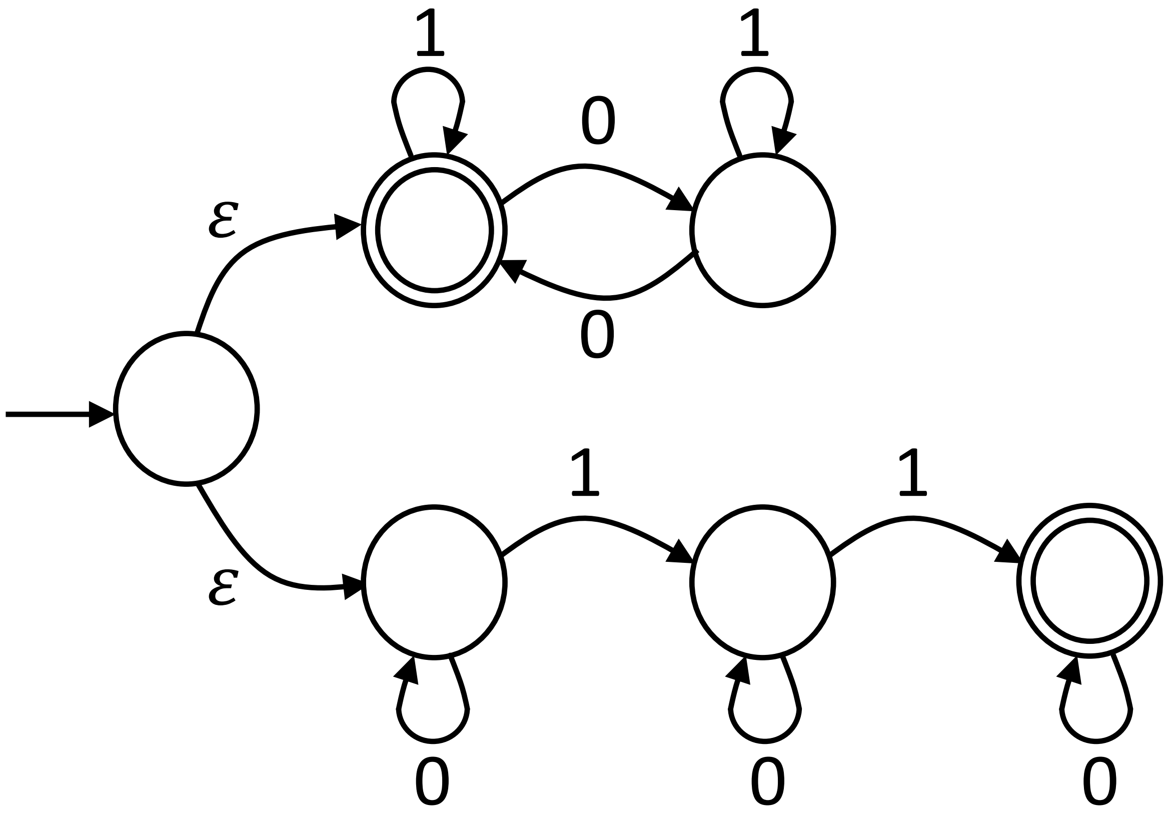
Build an NFA for the following language:

$\{\omega: \omega$ contains an even number of 0's or exactly two 1's$\}$.



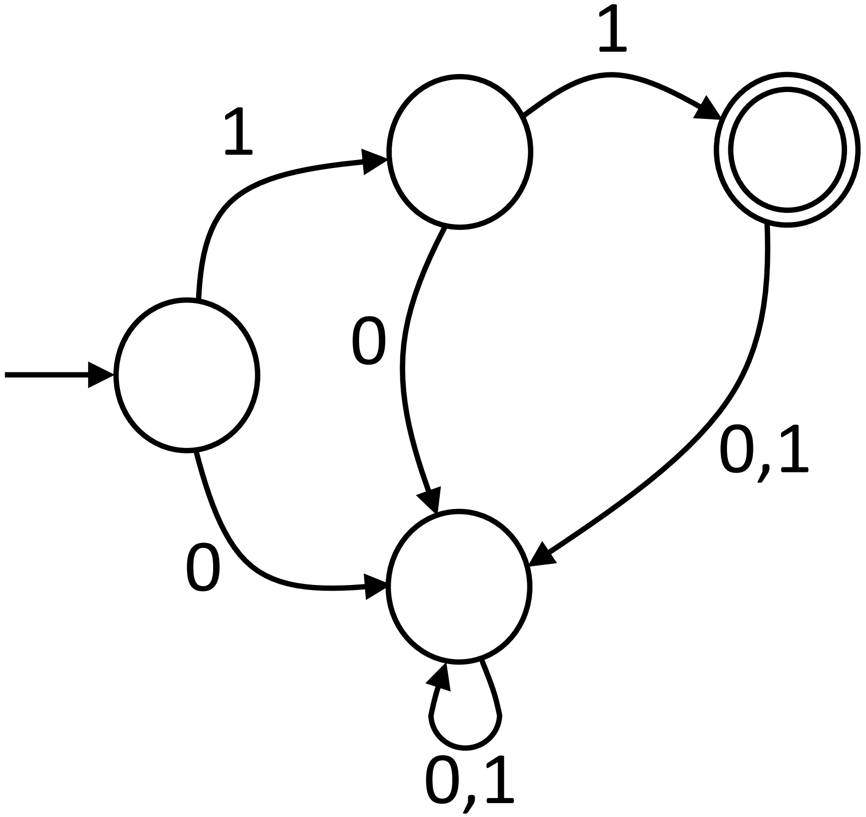
Build an NFA for the following language:

$\{\omega : \omega$ contains an even number of 0's or exactly two 1's$\}$.



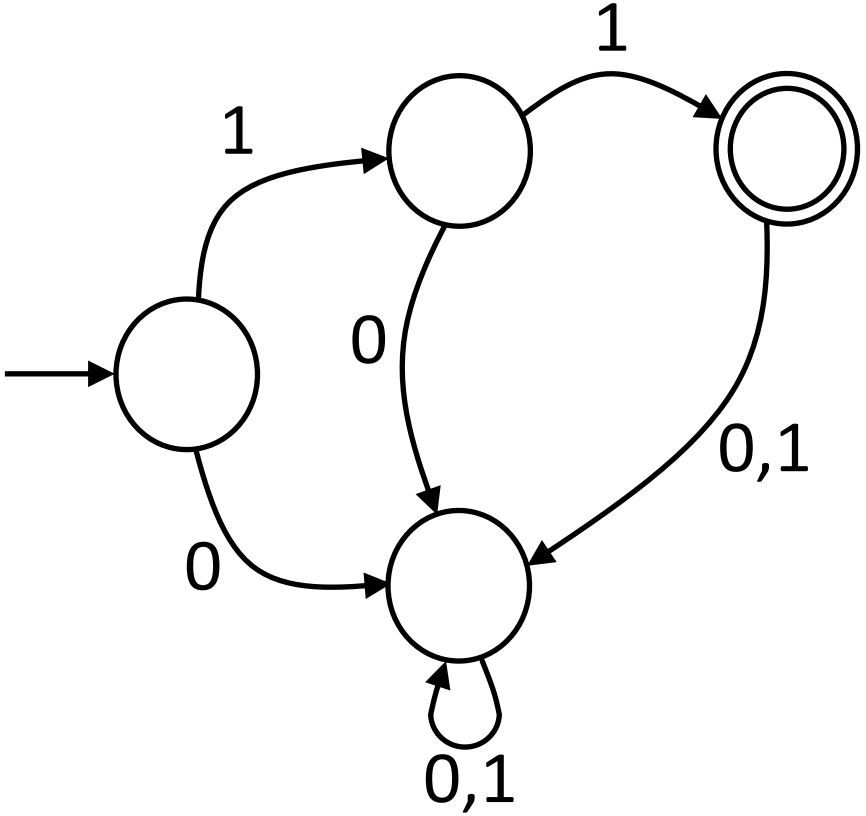
Build an NFA for the following language:

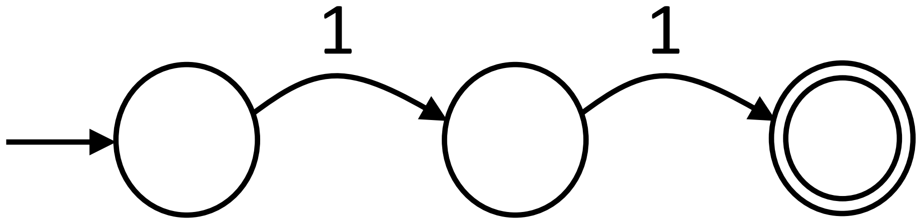
$\{11\}$.



Build an NFA for the following language:

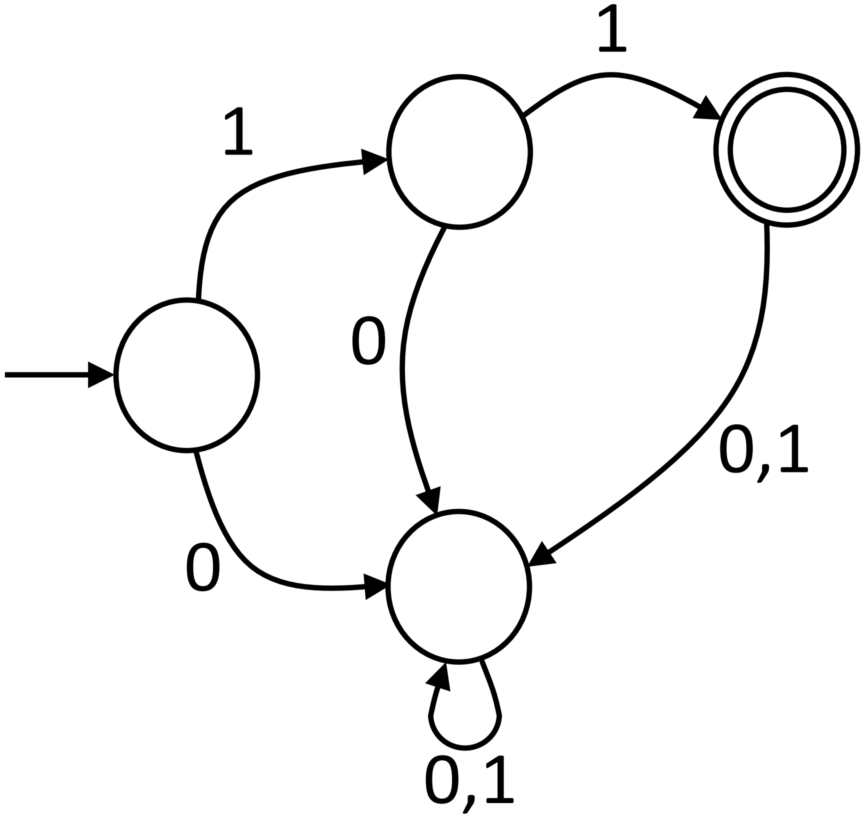
$\{11\}$.

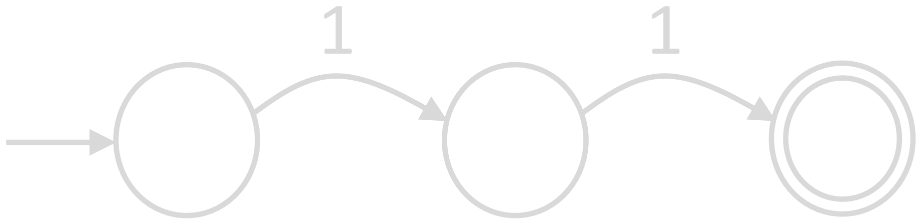




Build an NFA for the following language:

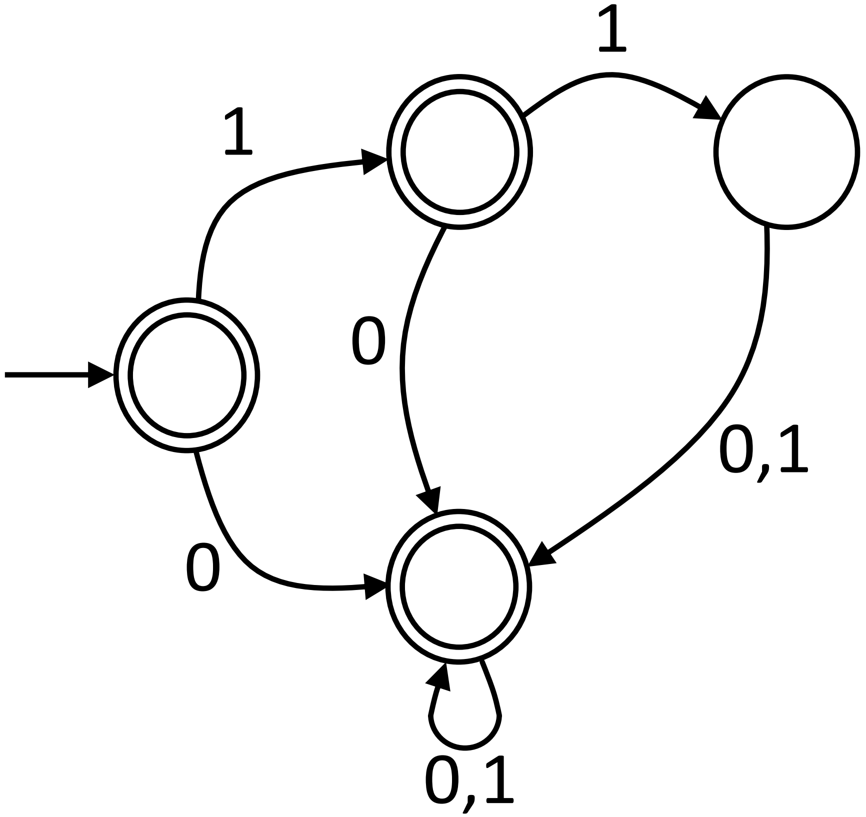
$\{\omega : \omega$ could be anything except $11\}$.

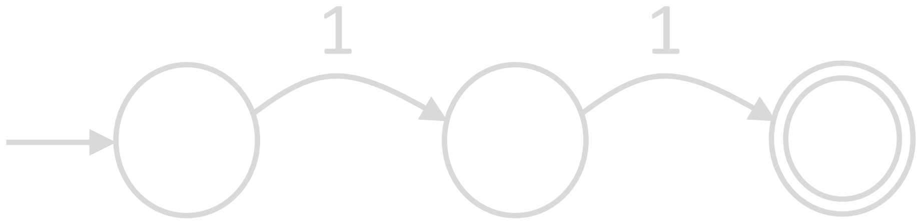




Build an NFA for the following language:

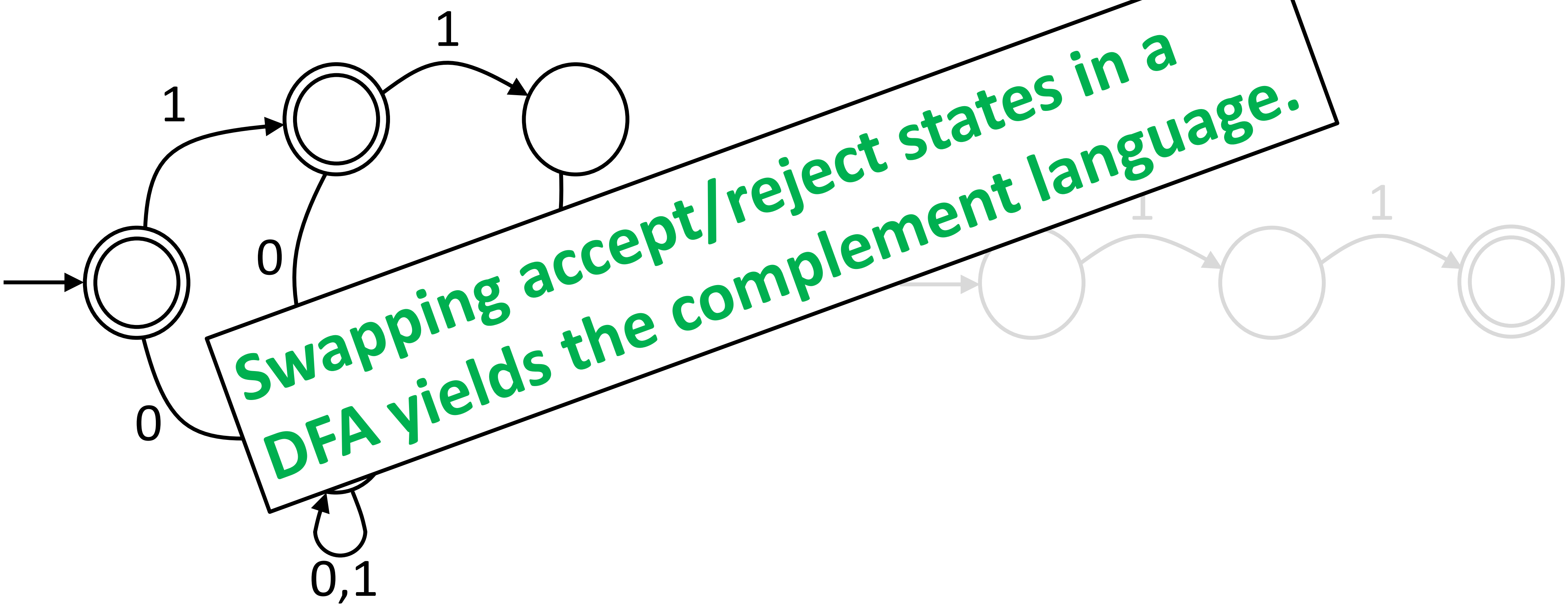
$\{\omega: \omega$ could be anything except $11\}$.





Build an NFA for the following language:

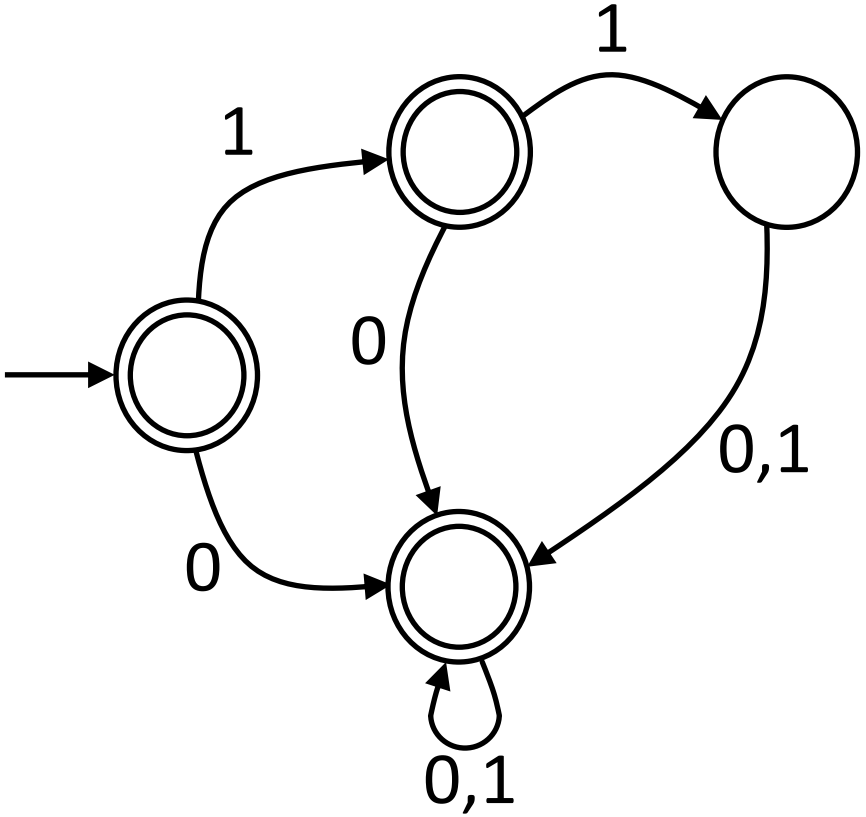
$\{\omega : \omega$ could be anything except $11\}$.

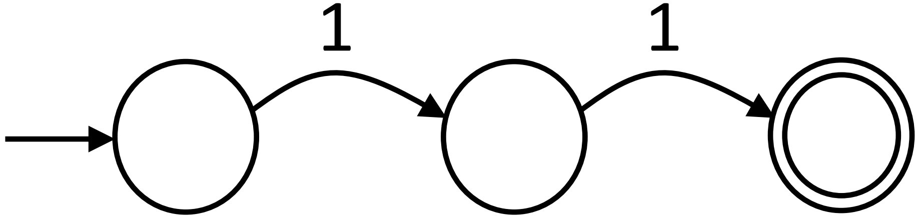


Swapping accept/reject states in a DFA yields the complement language.

Build an NFA for the following language:

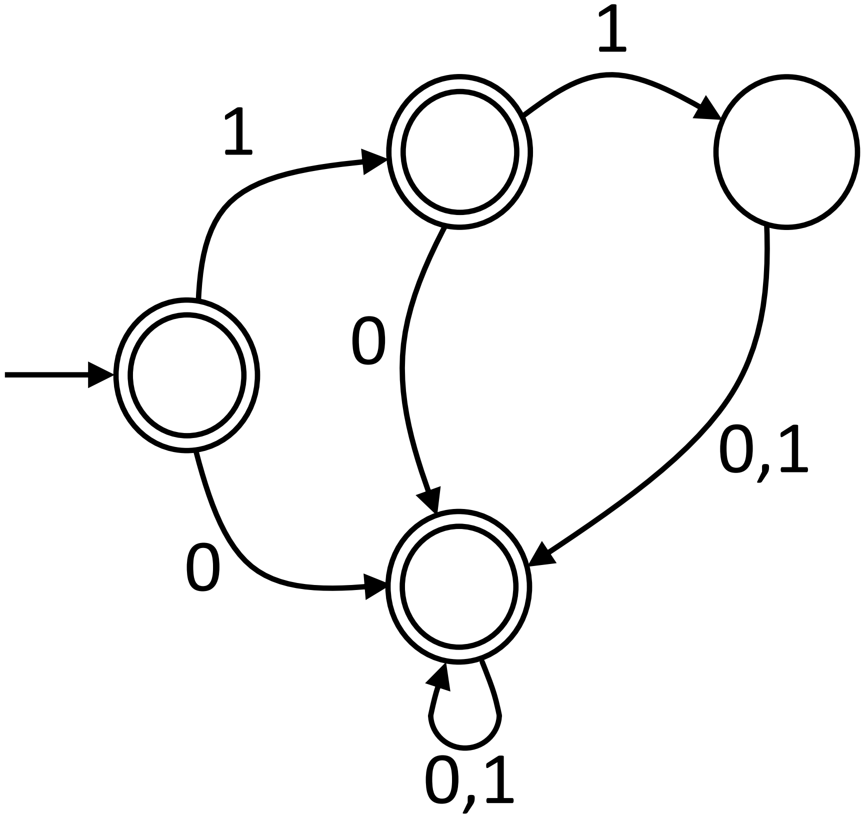
$\{\omega\colon \omega$ could be anything except $11\}$.

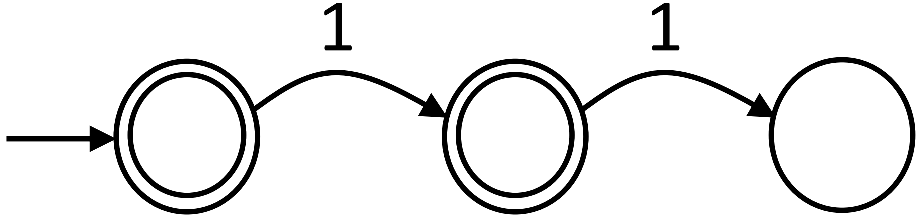




Build an NFA for the following language:

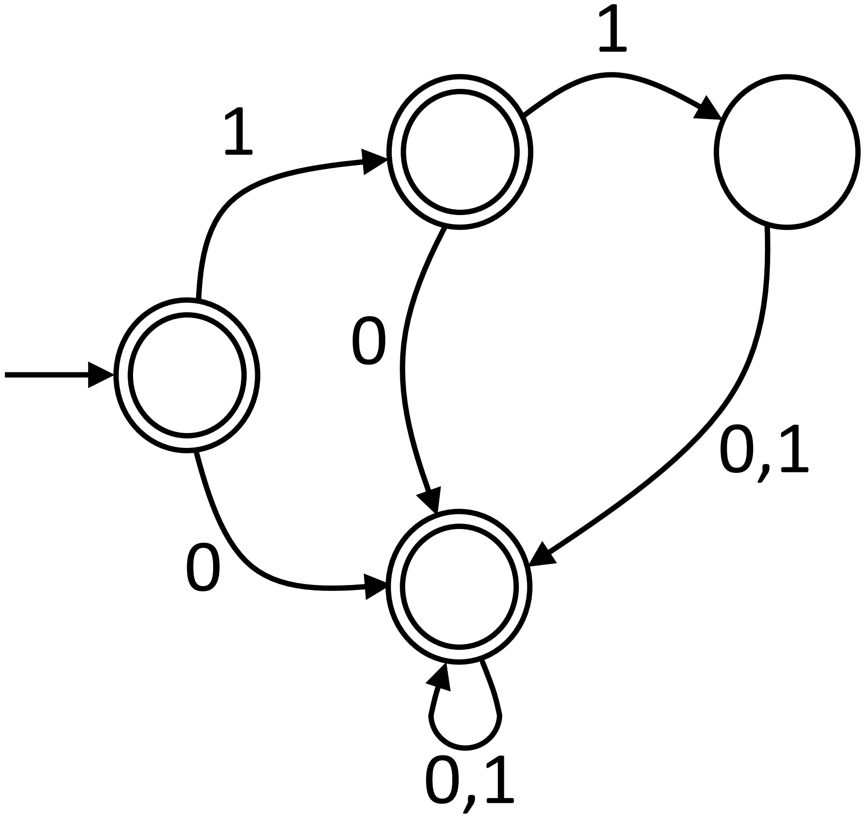
$\{\omega \colon \omega$ could be anything except $11\}$.

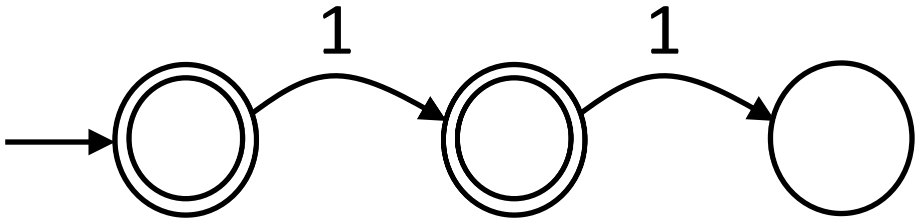




Build an NFA for the following language:

$\{\omega : \omega \text{ could be anything except } 11\}$.

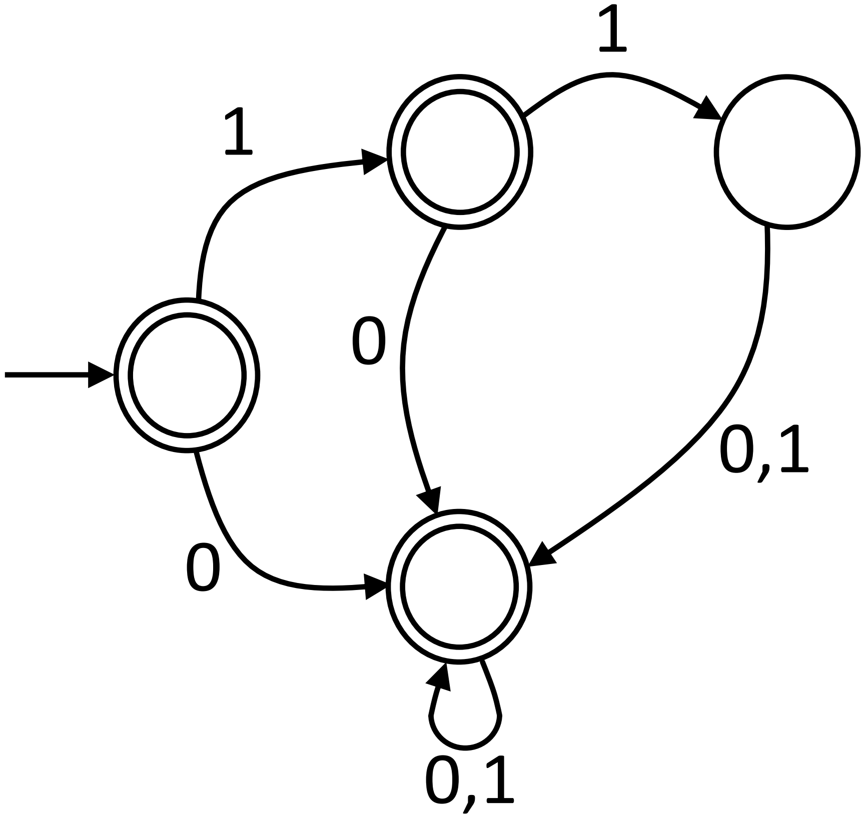


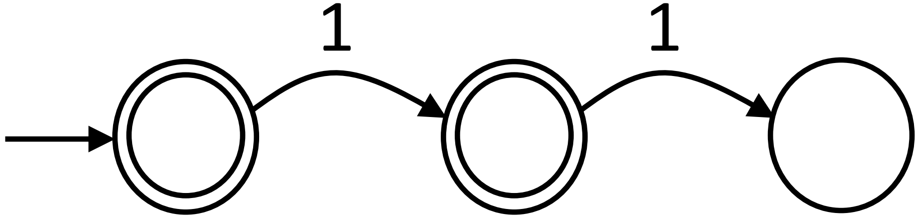


Language?

Build an NFA for the following language:

$\{\omega : \omega$ could be anything except $11\}$.

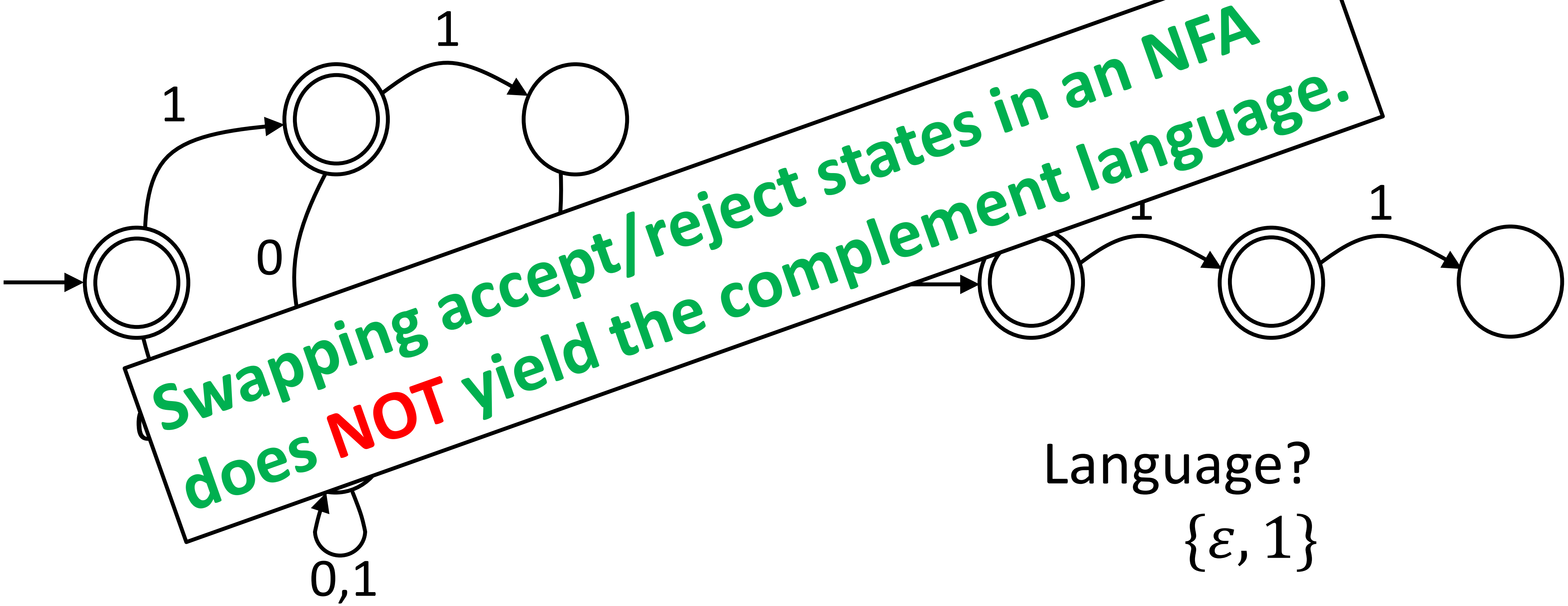




Language?

$\{\varepsilon, 1\}$

Build an NFA for the following language:
$\{\omega: \omega$ could be anything except $11\}$.



Swapping accept/reject states in an NFA does NOT yield the complement language.

Language?
$\{\varepsilon, 1\}$

Build an NFA for the following language:

$\{\omega\colon \omega$ contains the same number of 0s and $1s\}$.