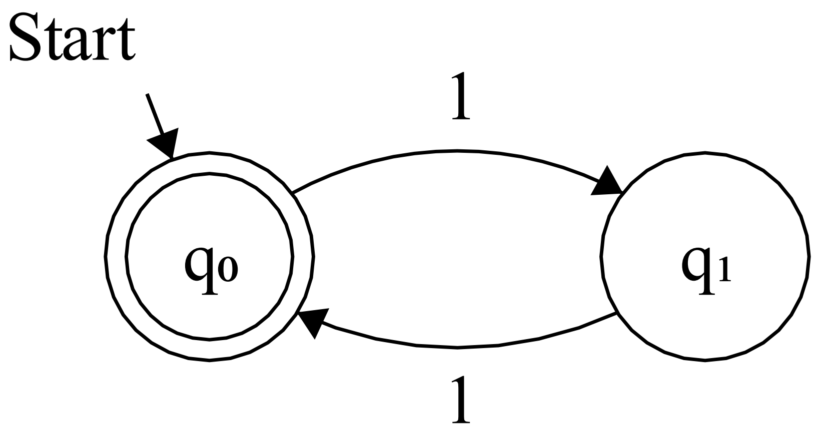
Finite Automata

COMP 455 – 002, Spring 2019

Example: Detect Even Number of 1s

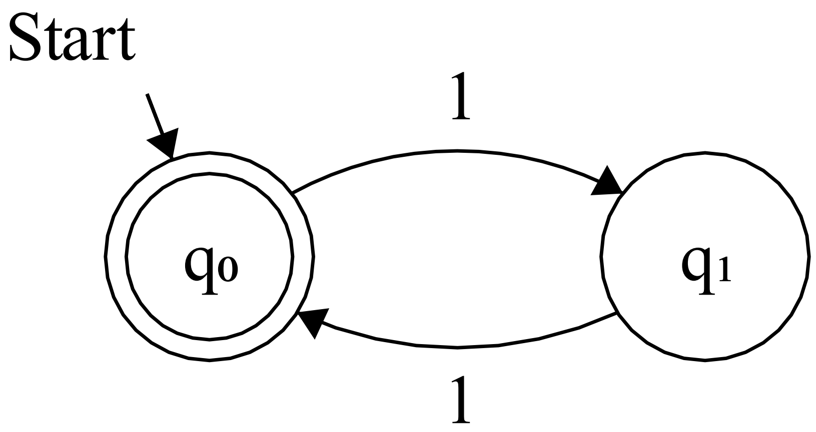
This is a "transition diagram" for a *deterministic finite automaton*.

Diagrams like this visualize automata like a simple game.



Example: Detect Even Number of 1s

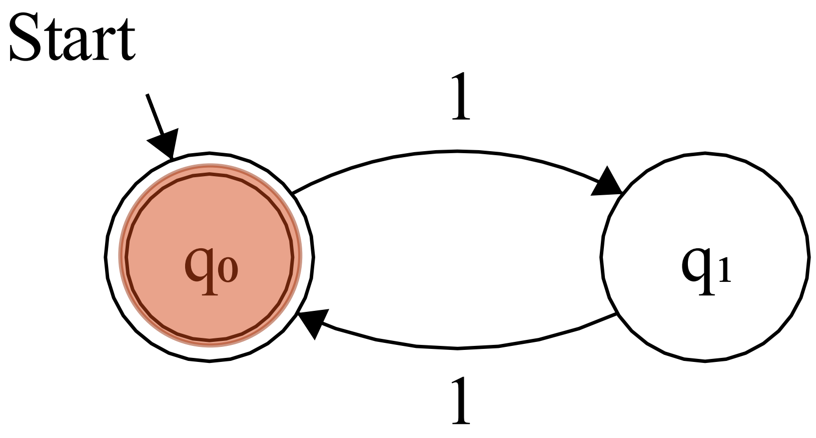
In this "game", we will move from circle to circle, following the instructions given by an input string.



Input string: 1111

Example: Detect Even Number of 1s

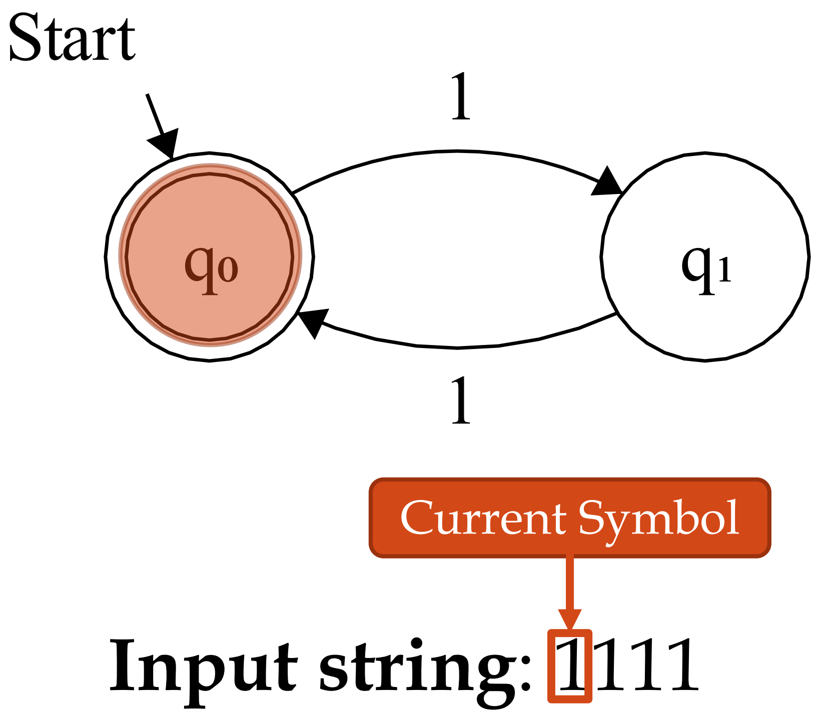
The “player” starts at the indicated circle:



Input string: 1111

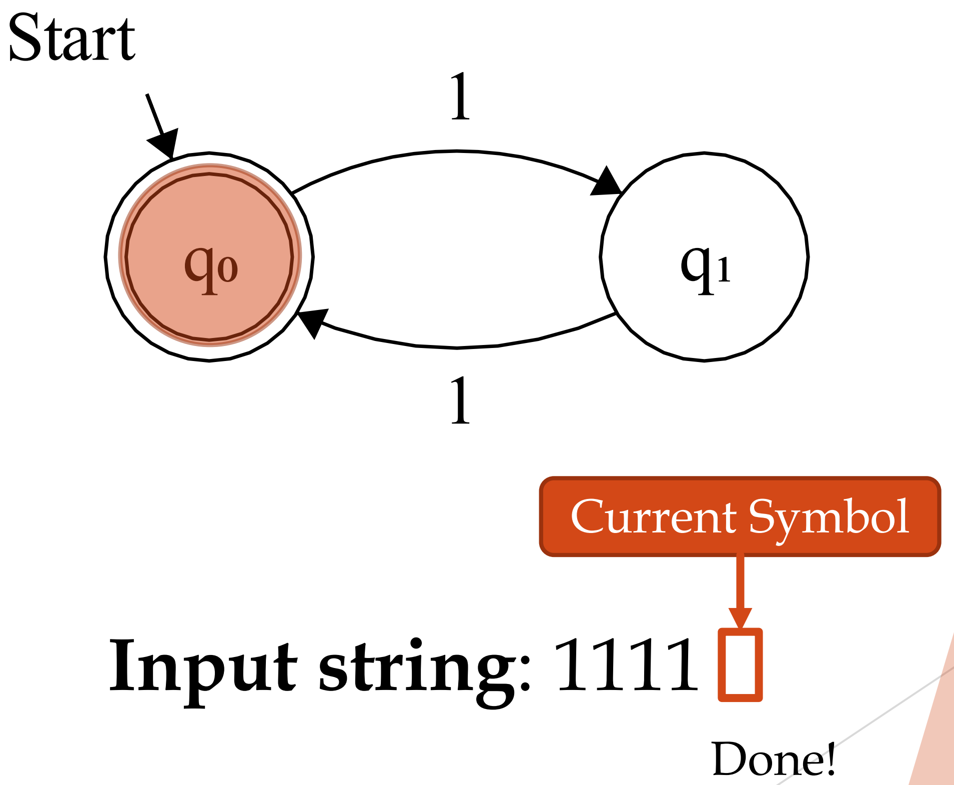
Example: Detect Even Number of 1s

The "game" proceeds by reading one character at a time from the input string and following the path labeled with the character.



Example: Detect Even Number of 1s

The "game" ends when all input symbols have been read.



Defining a Deterministic Finite Automaton

We define a deterministic finite automaton (DFA) as a 5-tuple: $(Q, \Sigma, \delta, q_0, F)$

- Q: A set of states
- Σ: A set of input symbols (the *alphabet*)
- q_0: The initial state. $q_0 \in Q$.
- F: A set of accepting ("final") states. $F \subseteq Q$.
- δ: The "transition function" mapping $Q \times \Sigma \rightarrow Q$.

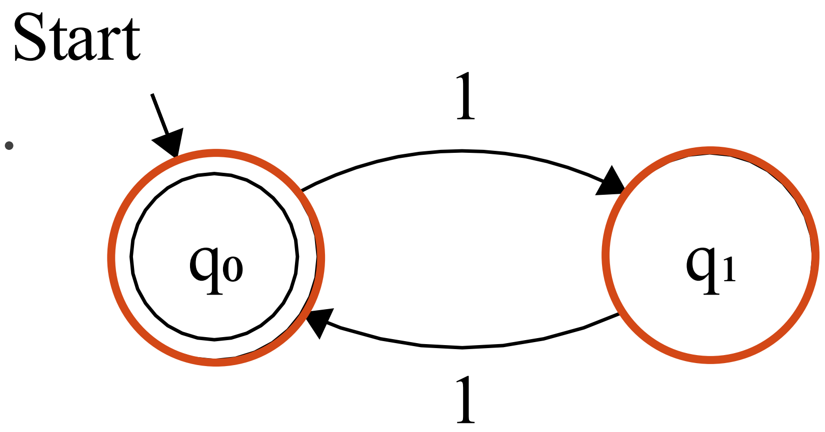
δ *must* be defined for all symbols in all states.

δ returns a single state.

From the Diagram to $(Q, \Sigma, \delta, q_0, F)$

The circles represent states.

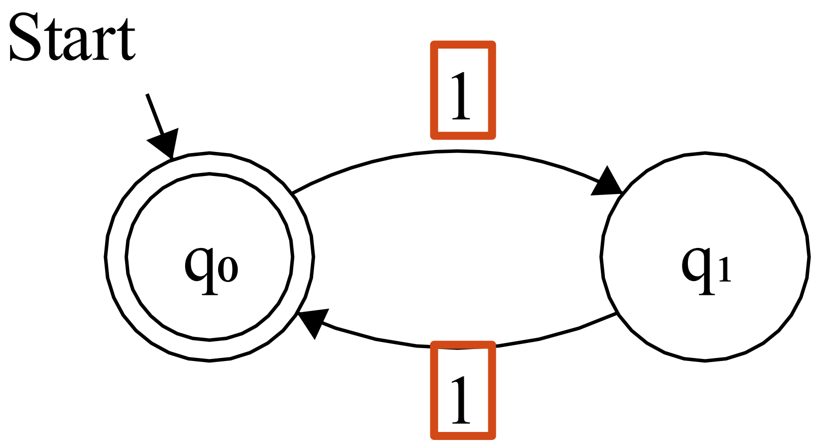
- $Q = \{q_0, q_1\}$



From the Diagram to $(Q, \Sigma, \delta, q_0, F)$

This automaton only handles strings of 1s

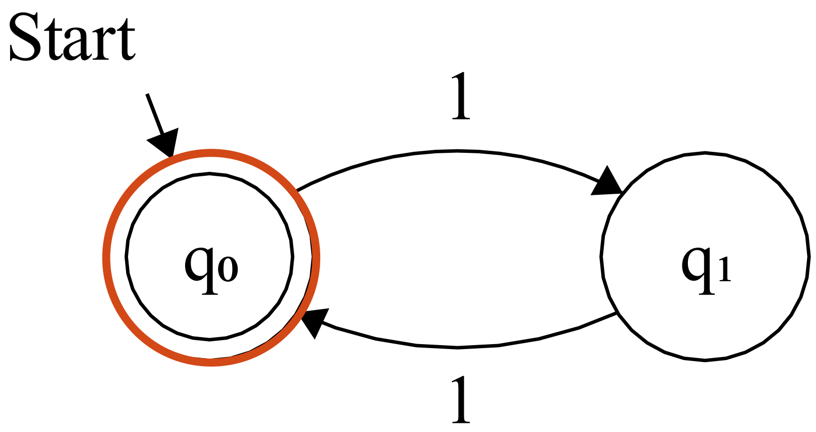
- $\Sigma = \{1\}$



From the Diagram to $(Q, \Sigma, \delta, q_0, F)$

The initial state is labeled with “Start”

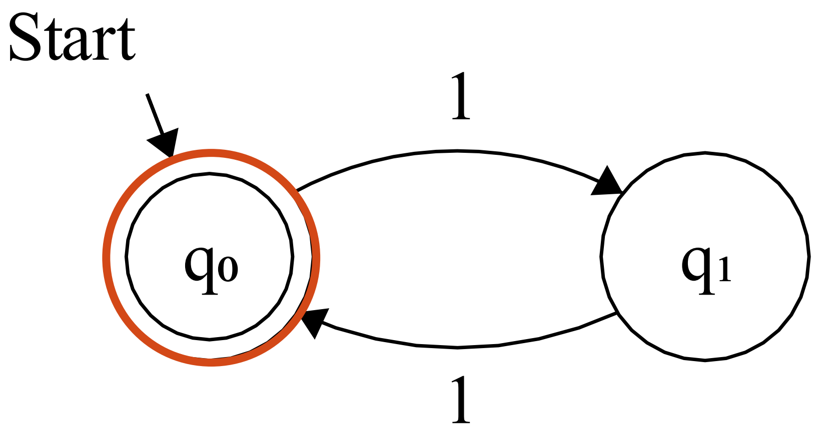
- $q_0 = q_0$



From the Diagram to $(Q, \Sigma, \delta, q_0, F)$

Accepting states have double circles. This automaton only has one.

- $F = \{q_0\}$

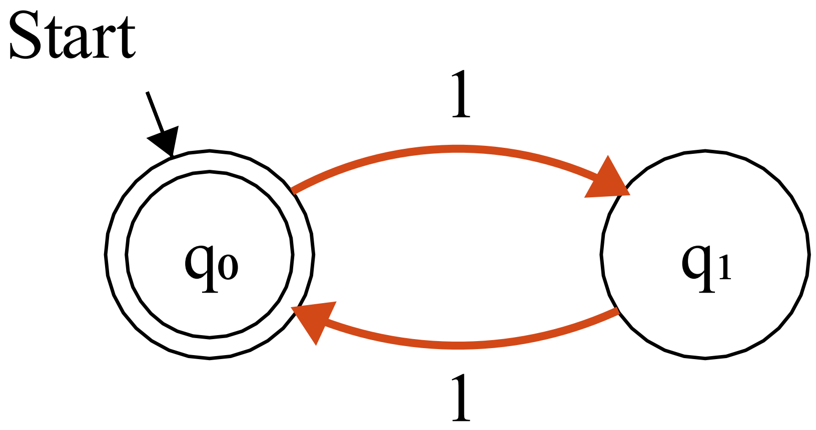


From the Diagram to $(Q, \Sigma, \delta, q_0, F)$

The arrows connecting the states represent possible transitions.

δ is the automaton's transition function:

$\delta(current\ state, current\ symbol)$
$= destination\ state$



δ can be represented as a table:

δ	1
q_0	q_1
q_1	q_0

Current input symbol

Current state

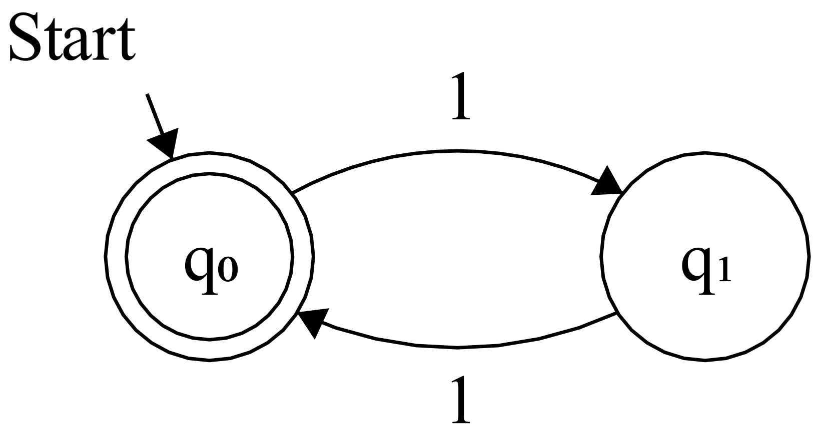
Destination state

From the Diagram to $(Q, \Sigma, \delta, q_0, F)$

This automaton in tuple notation:

- $Q = \{q_0, q_1\}$
- $\Sigma = \{1\}$
- $q_0 = q_0$
- $F = \{q_0\}$
- $\delta =$

δ	1
q_0	q_1
q_1	q_0



We use a table here, but δ can also be described using words, mathematical formulas, *etc*.

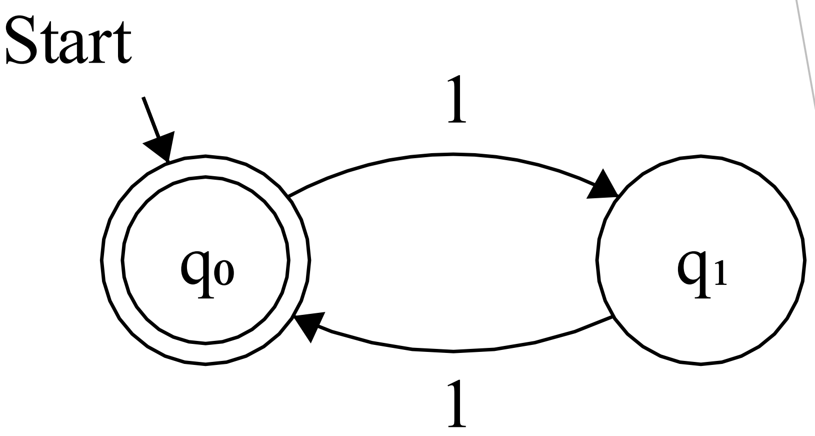
"Transition Table" Representation

The table for δ implicitly lists the states and alphabet.

A "transition table" just adds the remaining needed information:

- Indicate the start state with $\rightarrow$
- Indicate accepting states with $*$
- Example:

δ	1
$*\!\rightarrow q_0$	q_1
q_1	q_0



Notation: Extending δ to Strings

- An automaton's standard transition function, δ, takes two parameters: a state and a *symbol*.
- The "extended transition function", $\hat{\delta}$, takes a state and a *string*.
- $\hat{\delta}$ can be defined in terms of δ:
  - Assume that w is a string, a is a symbol in Σ, and q is a state.
  - Recursively, $\hat{\delta}(q, wa) = \delta\left(\hat{\delta}(q, w), a\right)$.
- Examples from the previous automaton:
  - $\hat{\delta}(q_0, \varepsilon) = q_0$
  - $\hat{\delta}(q_0, 111) = q_1$
  - $\hat{\delta}(q_0, 1111) = q_0$

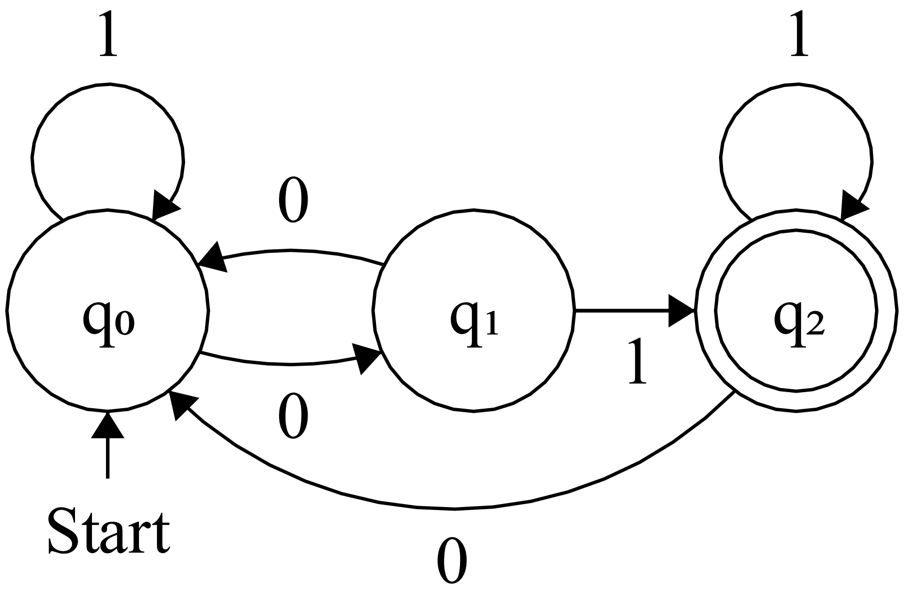
$\delta(q, a) = \hat{\delta}(q, a)$ for DFAs.

Example: Proofs About Automata

Here is a more complex automaton.

Transition table representation:

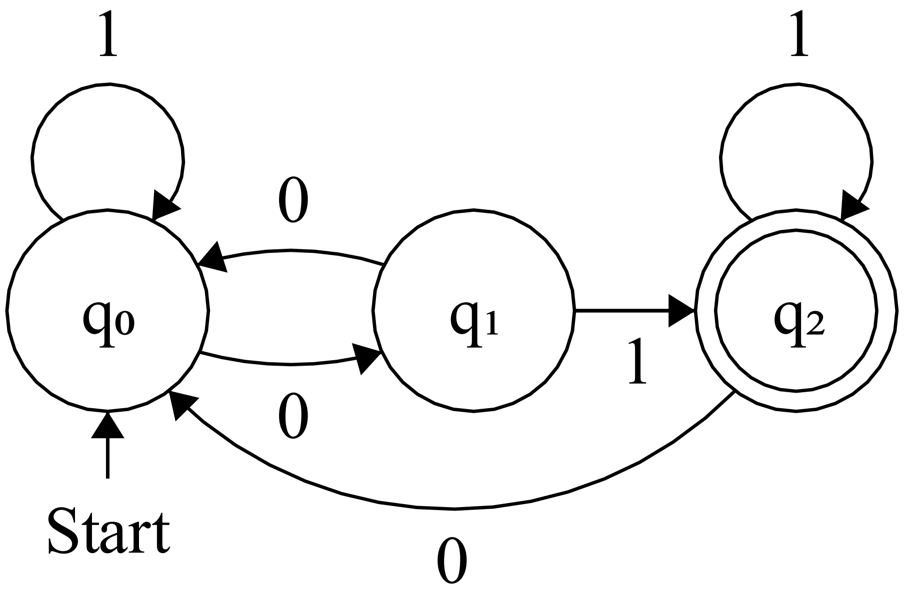
δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_2
$* q_2$	q_0	q_2



Example: Proofs About Automata

Questions:

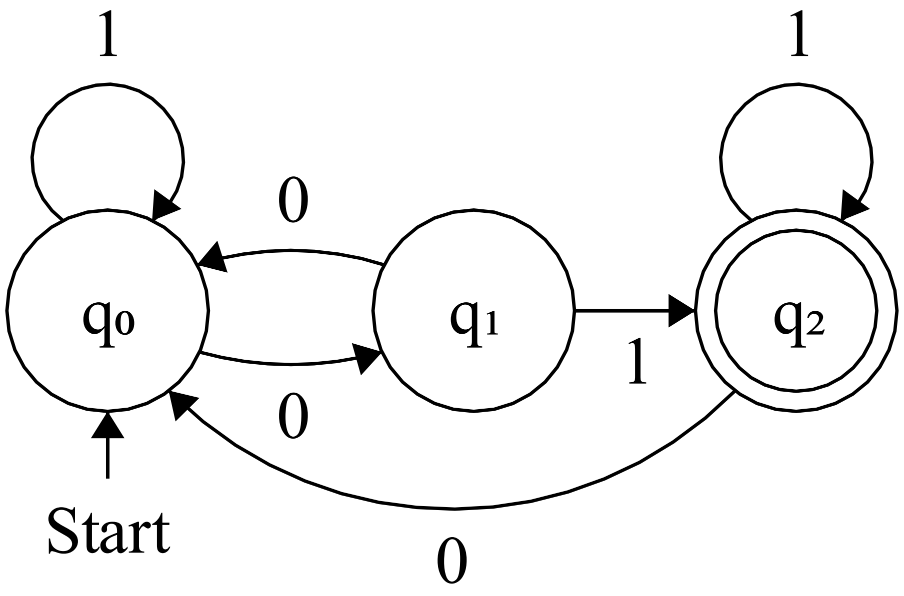
- What language does this automaton represent?
- How should we prove it's correct?



Example: Proofs About Automata

$L = \{x \mid x$ has an odd number of 0s and ends with a 1$\}$.

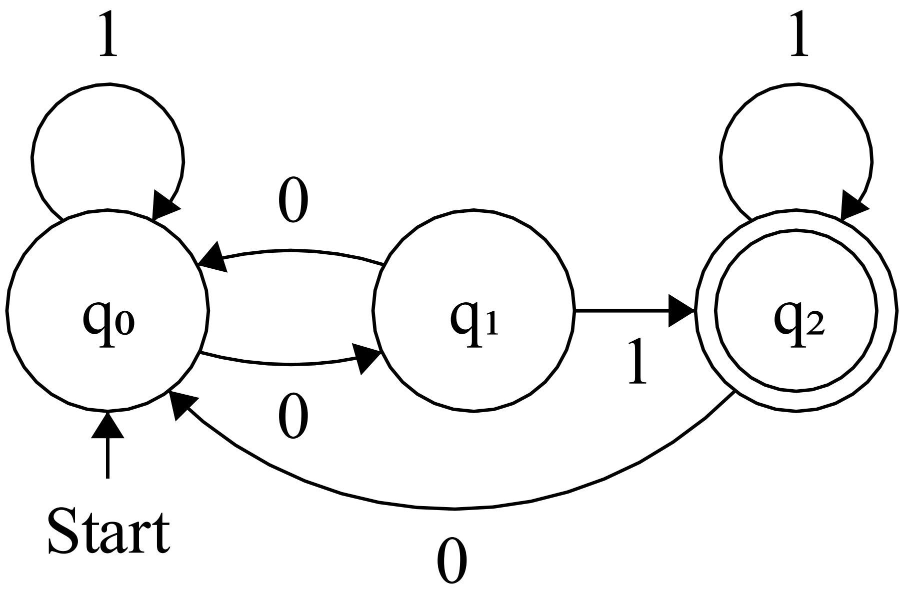
We will prove this by induction on the length of an input string, x.



Example: Proofs About Automata

We start our proof by defining what each state means about the input read so far:

- q_0: Even # of 0s
- q_1: Odd # of 0s, and ends with a 0
- q_2: Odd # of 0s, and ends with a 1



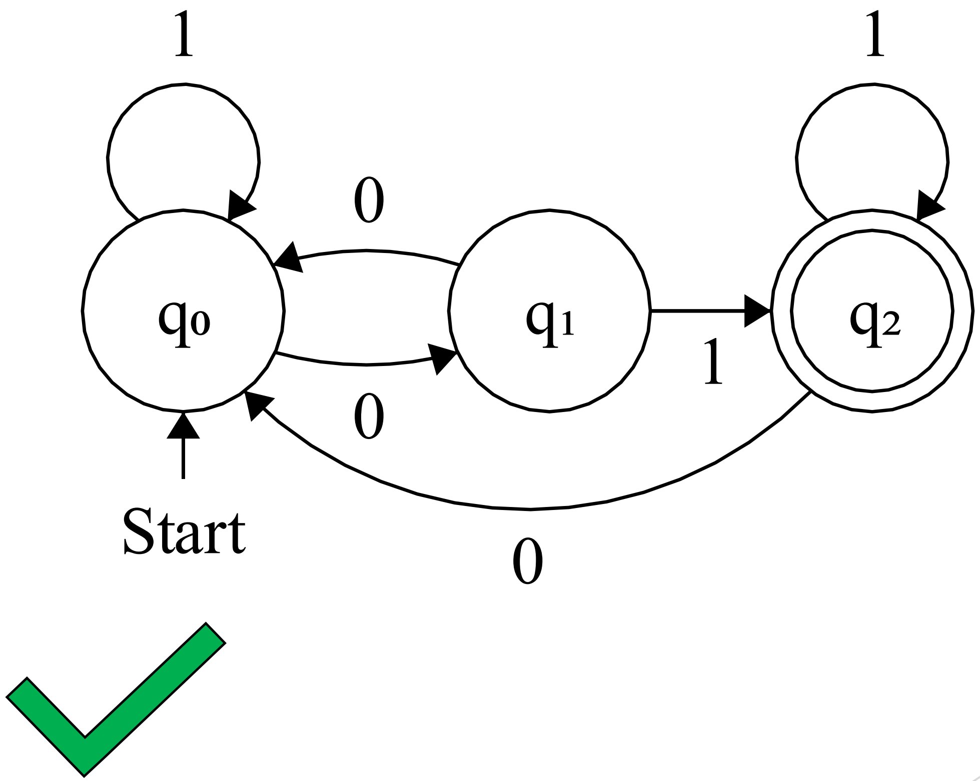
Proof obligation:
Show that these definitions are correct!

Example: Proofs About Automata

Base case: Prove the definition is correct for a string of length 0 (ε).

The automaton is in state q_0 after processing ε.

Since ε contains an even number of 0s, our definition of q_0 holds.

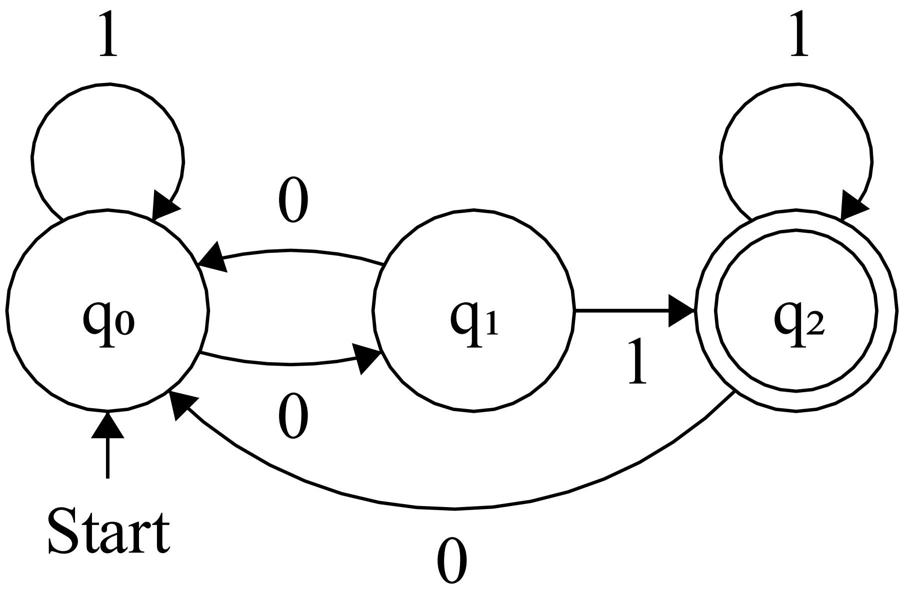


Example: Proofs About Automata

Inductive step: Assume that $\hat{\delta}(q_0, x)$ is correct for string x.

We need to prove that $\hat{\delta}(q_0, xa)$ remains correct for any symbol a.

This requires proving correctness for all possible transitions from all three states (mutual induction).



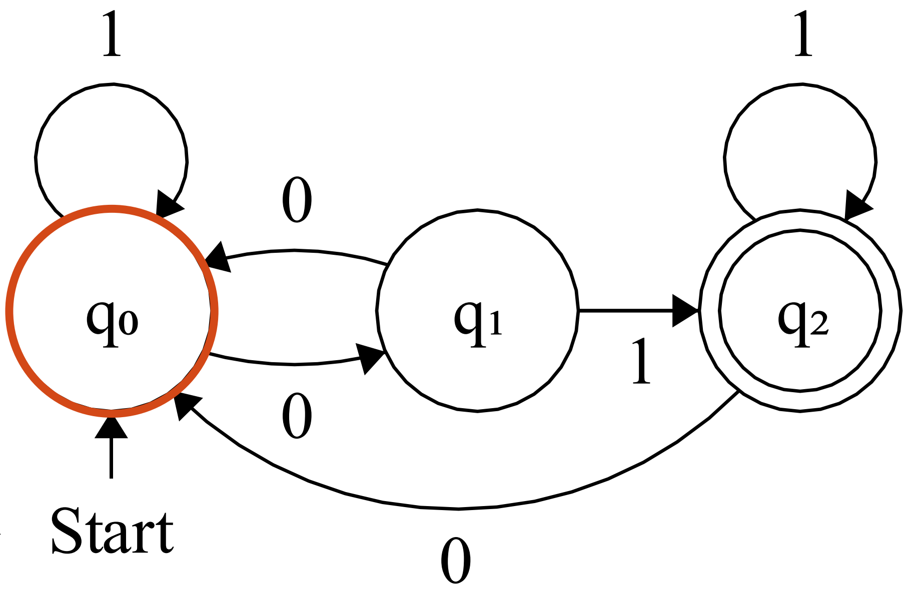
Example: Proofs About Automata

Induction part 1: state q_0

If $\hat{\delta}(q_0, x) = q_0$, then we can assume x contained an even number of 0s.

$\delta(q_0, 1) = q_0$. Reading a 1 doesn't change the # of 0s, so this is correct.

$\delta(q_0, 0) = q_1$. We've read an odd # of zeros, but the string doesn't end in 1 yet.



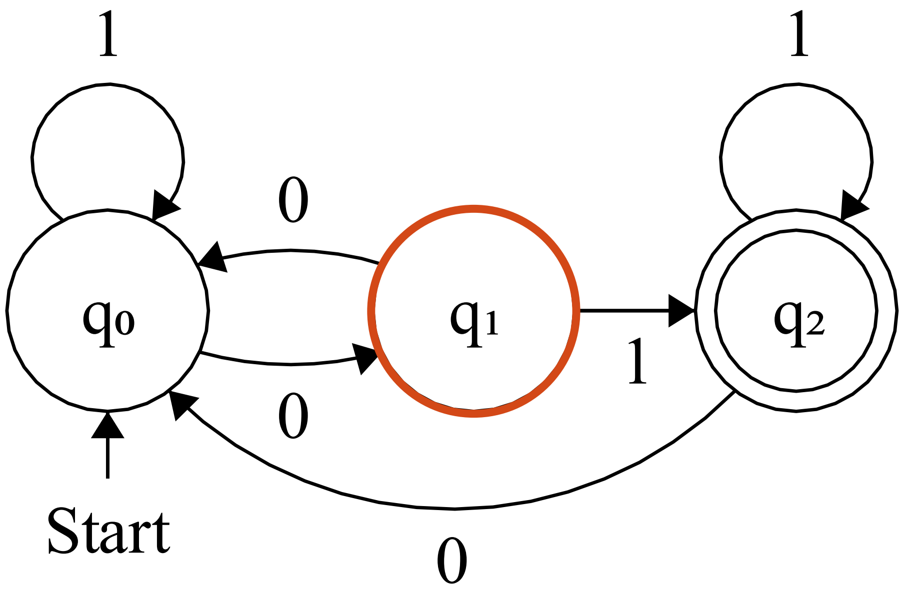
Example: Proofs About Automata

Induction part 2: state q_1

If $\hat{\delta}(q_0, x) = q_1$, then we can assume x contained an odd number of 0s and ends with a 0.

$\delta(q_1, 1) = q_2$. The string contains an odd # of 0s, but now ends with 1. ✓

$\delta(q_1, 0) = q_0$. Reading an additional 0 means that the string contains an even number of 0s. ✓



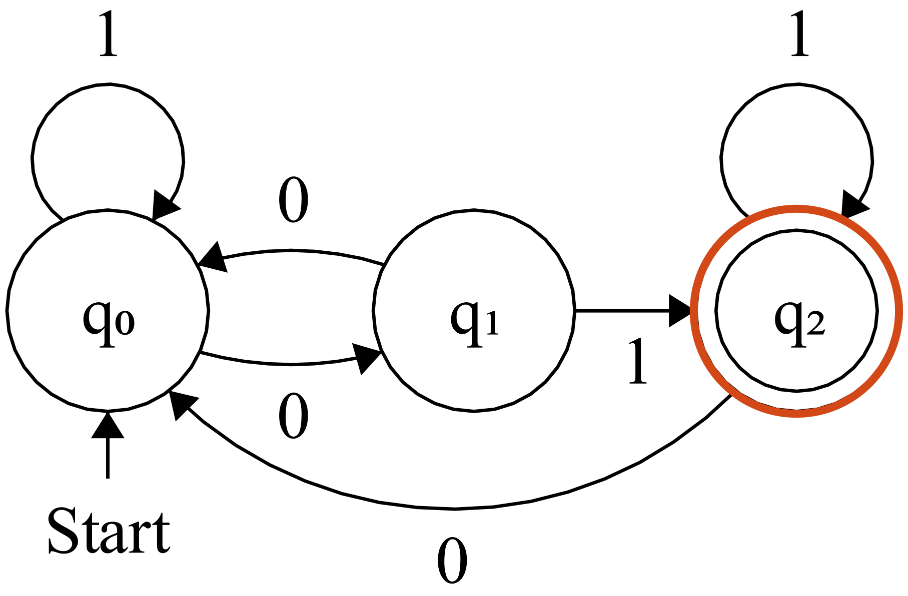
Example: Proofs About Automata

Induction part 3: state q_2

If $\hat{\delta}(q_0, x) = q_2$, then we can assume x contained an odd number of 0s and ends with a 1.

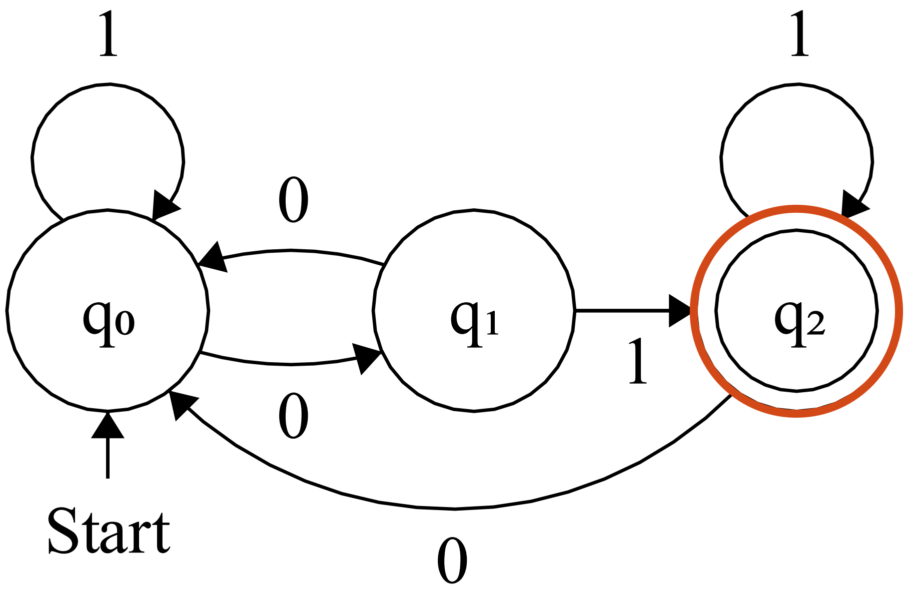
$\delta(q_2, 1) = q_2$. This doesn't change the # of 0s or the fact that the string ends with a 1. ✓

$\delta(q_2, 0) = q_0$. Reading an additional 0 means that the string contains an even number of 0s. ✓



Example: Proofs About Automata

Finishing up:
We've now proven that our claims about the states were correct, but we still need to prove the automaton recognizes the language.



The automaton ends in q_2 if and only if the string contained an odd number of 0s and ended with 1.

Since q_2 is the only accepting state, the automaton accepts strings if and only if they contain an odd number of 0s and end with a 1.

Nondeterministic Finite Automata

- We've been looking at deterministic finite automata (DFAs) so far.
  - δ returns exactly one state for every symbol in every state
- With nondeterministic finite automata (NFAs), the transition function δ returns a *set of states*.
  - $\delta: Q \times \Sigma \rightarrow 2^Q$
  - This can include no states at all!

2^Q: The *power set* of Q. (the set of all possible subsets of Q)

Note on "Nondeterministic" Terminology

- DFAs always follow the same path for a single input string.
  - DFAs accept a string if and only if this path leads to an accepting state.
- NFAs may follow one of many different paths for the same input string.
  - This is why they are called *nondeterministic*.
  - NFAs accept strings if and only if *it is possible* for them to reach an accepting state for a given input string.

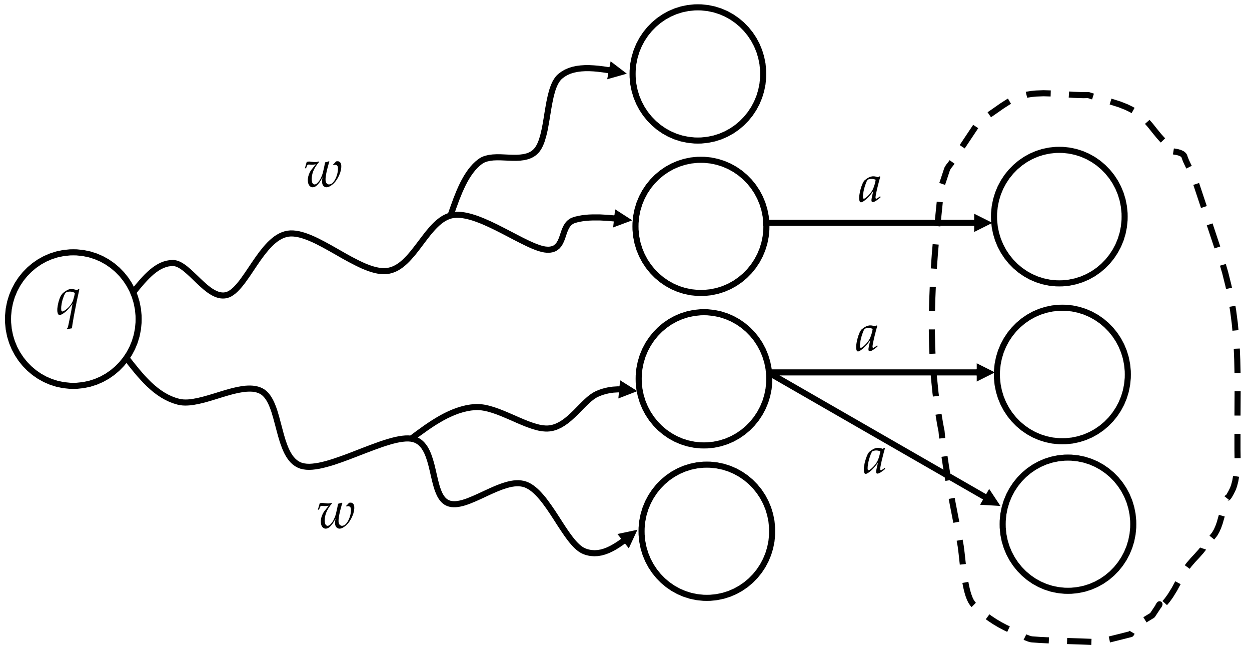
Extended Transition Function for NFAs

- As with DFAs, $\hat{\delta}$ for NFAs processes a string rather than a single character.
- As with the definition of δ for NFAs, $\hat{\delta}$ for NFAs returns the set of states an NFA is in after processing a string.
- If an NFA has a start state q, then the NFA accepts string x if and only if $\hat{\delta}(q, x) \cap F \neq \emptyset$
  - "The set of states after processing x contains at least one accepting state"

Recursive Definition of $\hat{\delta}$ for NFAs

As with DFAs, $\delta(q,a) = \hat{\delta}(q,a)$ for NFAs.

- $\hat{\delta}(q,\varepsilon) = q$
- $\hat{\delta}(q,wa) = \{\, p \mid \text{for some state } r \text{ in } \hat{\delta}(q,w),\ p \text{ is in } \delta(r,a) \}$



States in $\hat{\delta}(q,wa)$.

Note: This is a set.

Definition of δ for Sets of States

- Since δ can return a set of states for NFAs, it can be helpful to define a version of δ that takes a set of states rather than a single state.

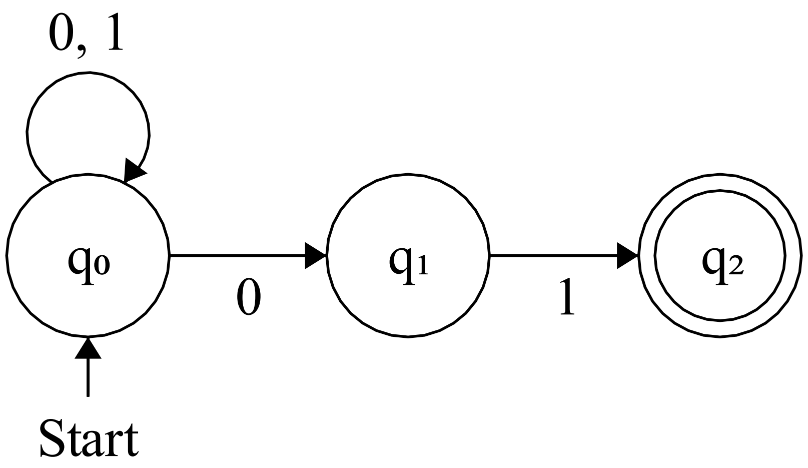
If P is a set of states,

$$\delta(P, a) = \bigcup_{q \in P} \delta(q, a)$$

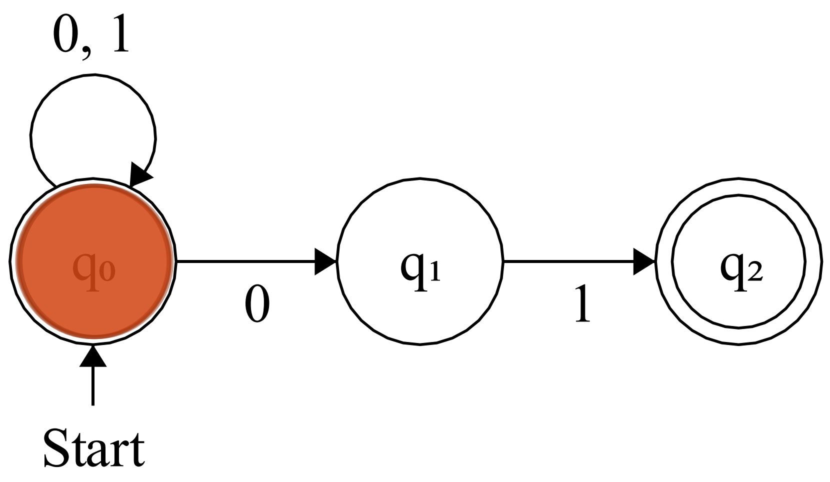
Nondeterministic Finite Automata

Example: Match all strings ending with 01

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$* q_2$	$\emptyset$	$\emptyset$



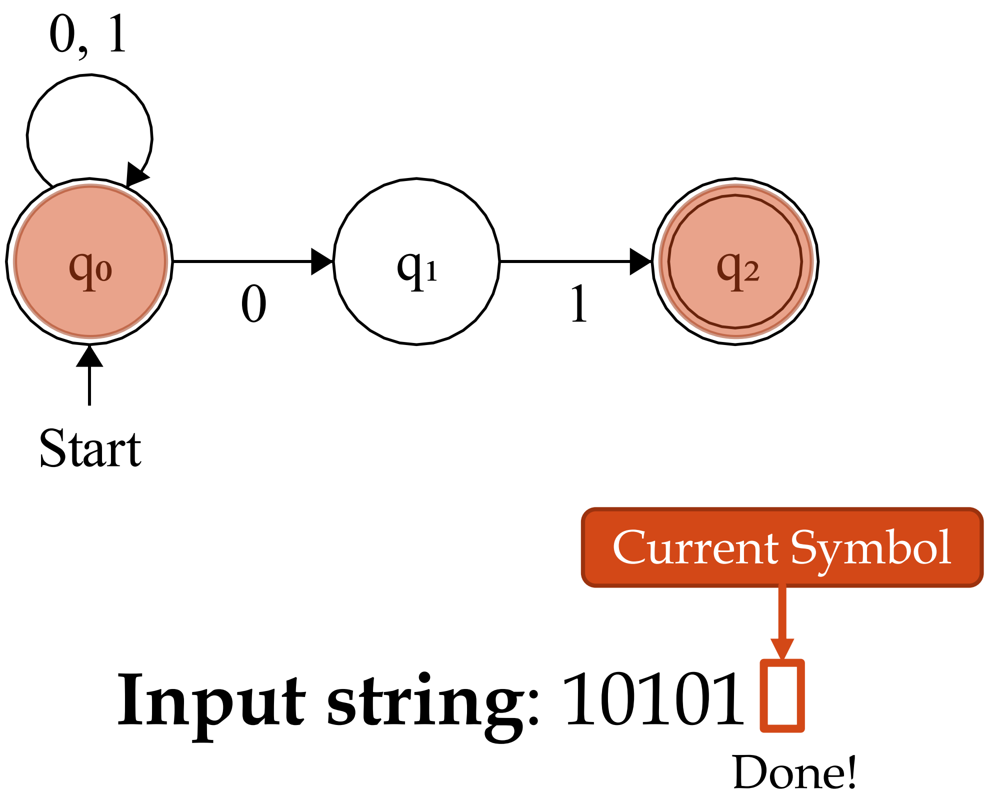
Example NFA Execution



Current Symbol

Input string: 10101

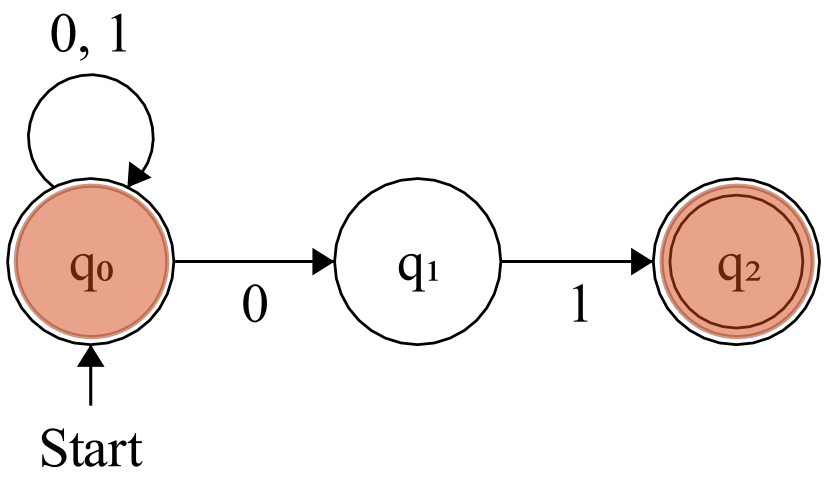
Example NFA Execution



Example NFA Execution

- $F = \{q_2\}$
- $\hat{\delta}(q_0, 10101) = \{q_0, q_2\}$
- $\{q_0, q_2\} \cap F = \{q_2\}$

As expected, this NFA accepts 10101, because it ends in a set of states containing an accepting state.



Input string: 10101

NFAs vs DFAs

- NFAs are often more convenient than DFAs
  - Write an NFA that accepts $L = \{x \mid x$ is a string of 0s or 1s that contains 0101000 as a substring$\}$
  - Now write a DFA that accepts L
- Good news: Any language that can be recognized by an NFA can also be recognized by a DFA.

Equivalence of NFAs and DFAs

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
- We will show how to construct an equivalent DFA, D, using a technique called *subset construction*.
- Main idea: D will keep track of the subset of states that N might be in.
  - In other words, each of D's states corresponds to a subset of N's states.

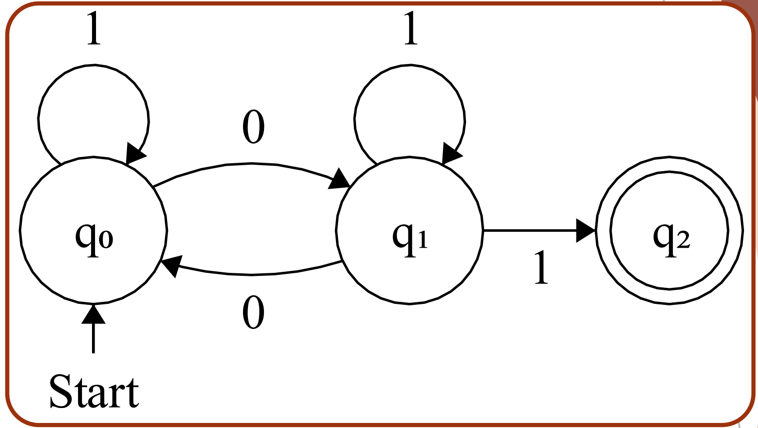
Subset Construction

- The constructed DFA $D = (Q', \Sigma, \delta', q_0', F')$, where:
  - $Q' = 2^Q$ = Assuming the states in Q are $\{q_0, q_1, \dots, q_n\}$, the states in Q' are all possible subsets of $\{q_0, q_1, \dots, q_n\}$.
  - $q_0' = \{q_0\}$
  - $F' = \{q \in Q' \mid q = \{\dots, q_j, \dots\}$ and $q_j \in F\}$
    - "D's final states consist of all subsets containing one or more of N's final states."
  - For all q and p in Q', $\delta'(q, a) = p$ iff $\delta(q, a) = p$.

Subset Construction: Example

- Start with the following NFA:
  - $N = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$
  - $\delta =$

δ	0	1
q_0	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_0\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\emptyset$

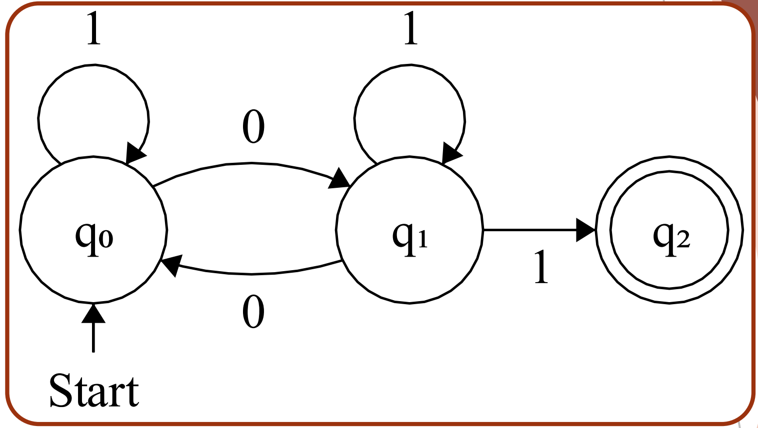


- Construct DFA $D = (Q', \{0,1\}, \delta', \{q_0\}, F')$, where
  - $Q' = \{\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$
  - $F' = \{\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$
  - δ' is defined on the following slide.

Subset Construction: Example

- $\delta' =$

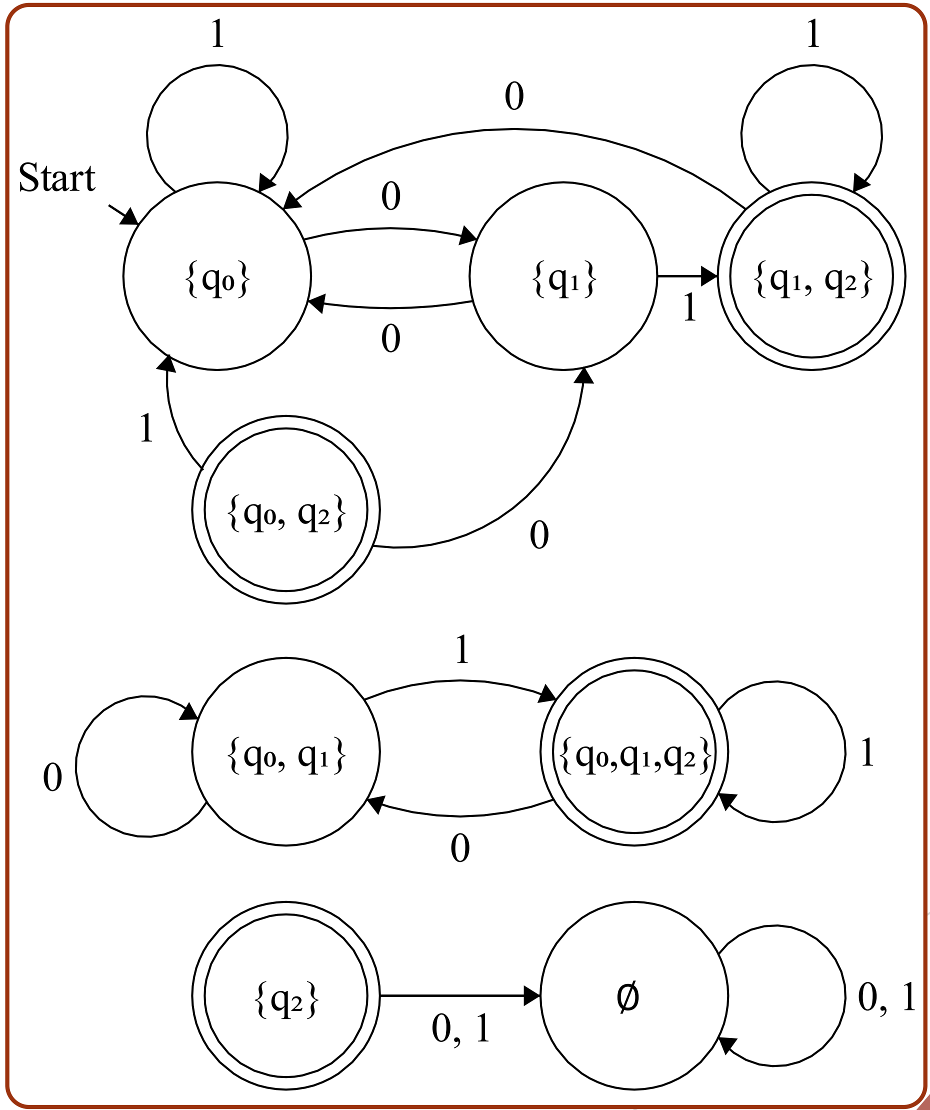
	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_0\}$	$\{q_1\}$	$\{q_0\}$
$\{q_1\}$	$\{q_0\}$	$\{q_1, q_2\}$
$\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_1\}$	$\{q_0\}$
$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$



Subset Construction: Example

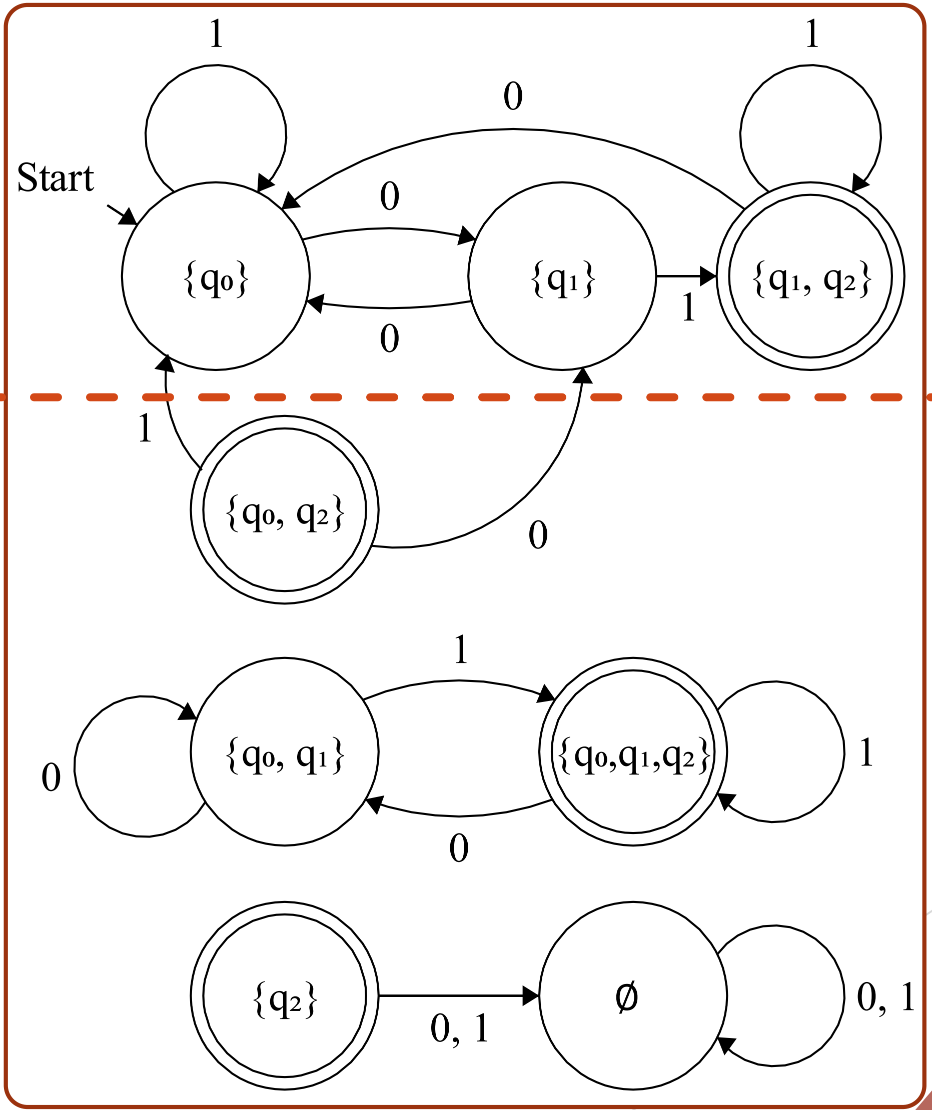
Diagram for D:

δ'	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_1\}$	$\{q_0\}$
$\{q_1\}$	$\{q_0\}$	$\{q_1, q_2\}$
$* \{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$
$* \{q_0, q_2\}$	$\{q_1\}$	$\{q_0\}$
$* \{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$
$* \{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$



Subset Construction: Example

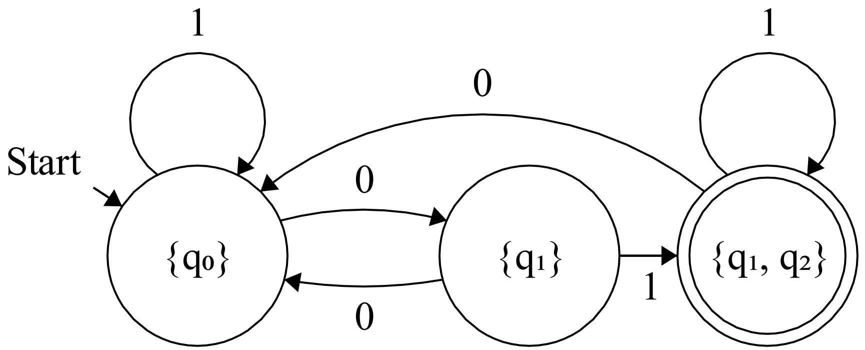
Diagram for *D*:



We don't care about these states; they are *unreachable* from the start state.

Subset Construction: Example

Diagram for *D* without unreachable states:



Theorem 2.11

Theorem 2.11 (from the textbook):
If N is the original NFA and D is the constructed DFA (as defined earlier), then $L(N) = L(D)$.

Proof: We need to show that $\hat{\delta}'(q_0', x) = S$ if and only if $\hat{\delta}(q_0, x) = S$.

We will prove this by induction on $|x|$.

Proof of Theorem 2.11

Reminder:
δ' is for the DFA D
δ is for the NFA N

Base case:

$|x| = 0$. (Put another way, $x = \varepsilon$).

Verifying the base case:

$$\hat{\delta}'(q_0', \varepsilon) = q_0' = \{q_0\}$$

and

$$\hat{\delta}(q_0, \varepsilon) = \{q_0\}$$

by the definition of the extended transition function.

So, $L(N) = L(D)$ holds for the base case.

Proof of Theorem 2.11

Reminder:
δ' is for the DFA D
δ is for the NFA N

Inductive step:

By the inductive hypothesis we assume that:

$$\hat{\delta}'(q_0', x) = S \text{ iff } \hat{\delta}(q_0, x) = S$$

We now apply the definition of the extended transition function to advance by a single symbol:

Reminder:
For string w and symbol a:
$\hat{\delta}(q, wa) = \delta\big(\hat{\delta}(q, w), a\big)$

$\hat{\delta}'(S, a) = T$ iff $\hat{\delta}(S, a) = T$, by the definition of δ'.

Therefore $\hat{\delta}'(q_0', x) = T$ iff $\hat{\delta}(q_0, x) = T$.

Finishing the Proof of Theorem 2.11

Reminder:
δ' is for the DFA D
δ is for the NFA N

Finally, since $\hat{\delta}'(q_0', x)$ is in F' if and only if $\hat{\delta}(q_0, x)$ contains a state in F, $L(D) = L(N)$.

- Note: This construction results in a *state explosion*.

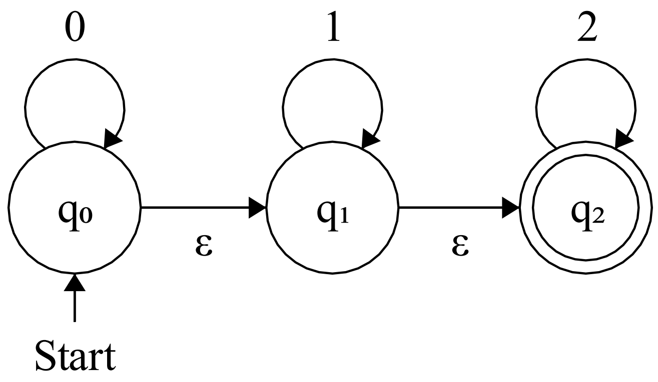
NFAs with ε-Transitions

NFAs with ε-transitions have all the same rules as regular NFAs, but with additional flexibility.

The transition function for NFAs with ε-transitions:

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$$

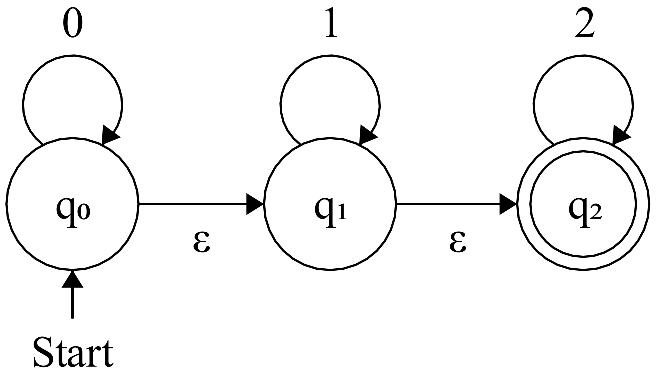
Example:



ε-Closure

Let $ECLOSE(q) \equiv$
$\{q\} \cup \{p \mid p \text{ is reachable from } q \text{ via } \varepsilon - \text{transitions}\}$

- $ECLOSE(q_0) = \{q_0, q_1, q_2\}$
- $ECLOSE(q_1) = \{q_1, q_2\}$



For a set of states P,

$$ECLOSE(P) \equiv \cup_{q \in P} ECLOSE(q)$$

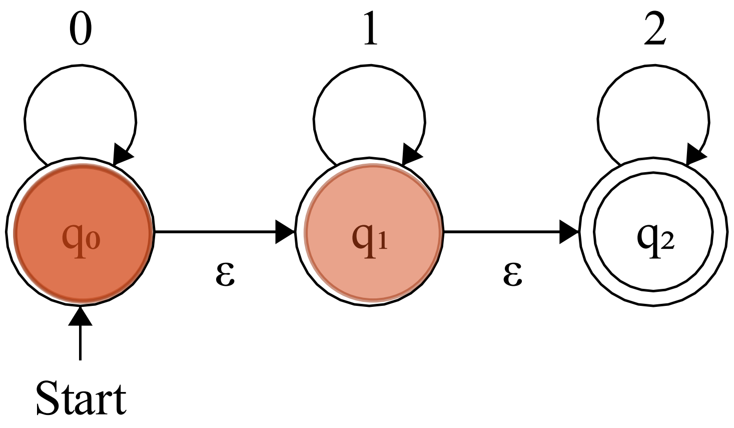
Definition of $\hat{\delta}$ for ε-NFAs

Recursive definition of $\hat{\delta}$:

- $\hat{\delta}(q, \varepsilon) = ECLOSE(q)$
- For a string w and symbol a: $\hat{\delta}(q, wa) = ECLOSE(P)$, where $P = \{p \mid \text{for some } r \text{ in } \hat{\delta}(q, w),\ p \text{ is in } \delta(r, a)\}$
- In other words, $\hat{\delta}(q, a) = ECLOSE\big(\delta(ECLOSE(q), a)\big)$ for a starting state q and a single symbol a.

Unlike before, $\hat{\delta}(q, a) \neq \delta(q, a)$ for ε-NFAs!

ε-NFA Example



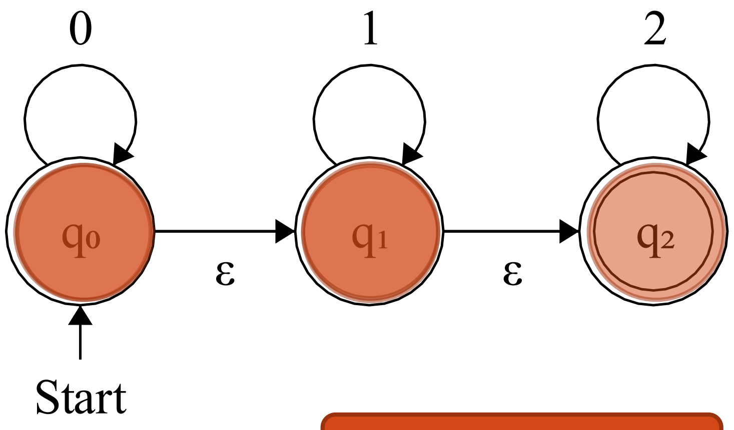
How to simulate an ε-NFA:

1. Follow transitions as you would for a normal NFA.
2. Take the ε-closure for any states you end up in.

Input string: 011

$$\hat{\delta}(\{q_0\}, \varepsilon) = \{q_0, q_1, q_2\}$$

ε-NFA Example



How to simulate an ε-NFA:

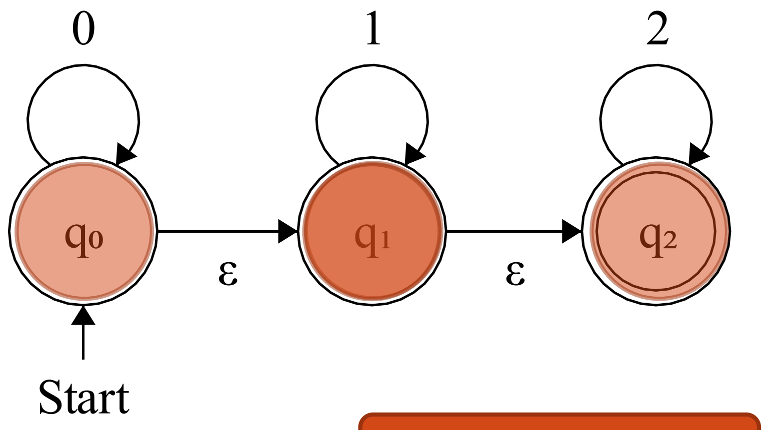
1. Follow transitions as you would for a normal NFA.
2. Take the ε-closure for any states you end up in.

Current Symbol

Input string: 011

$$\hat{\delta}(\{q_0\}, 0) = \{q_0, q_1, q_2\}$$

ε-NFA Example



Current Symbol

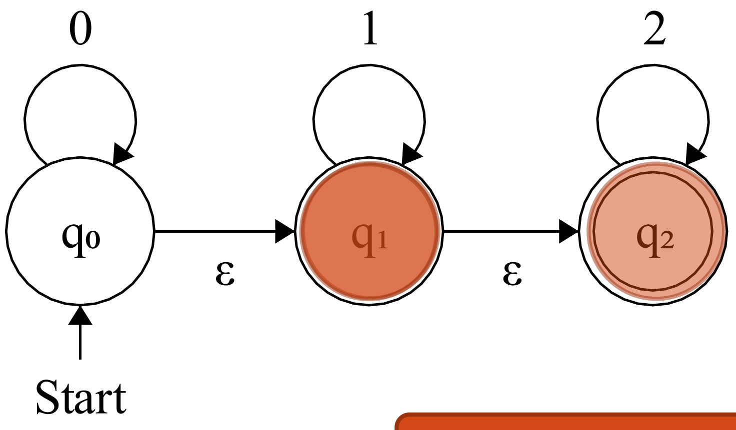
Input string: 011

$$\hat{\delta}(\{q_0\}, 01) = \{q_1, q_2\}$$

How to simulate an ε-NFA:

1. Follow transitions as you would for a normal NFA.
2. Take the ε-closure for any states you end up in.

ε-NFA Example



How to simulate an ε-NFA:

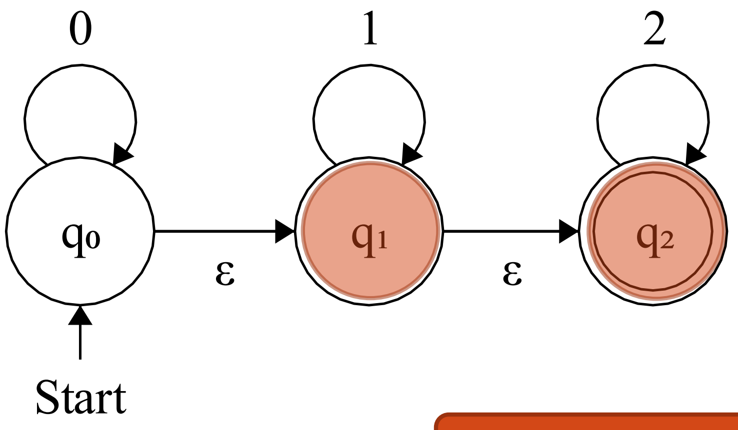
1. Follow transitions as you would for a normal NFA.
2. Take the ε-closure for any states you end up in.

Current Symbol

Input string: 011

$$\hat{\delta}(\{q_0\}, 011) = \{q_1, q_2\}$$

ε-NFA Example



Current Symbol

Input string: 011 Done!

$$\hat{\delta}(\{q_0\}, 011) = \{q_1, q_2\}$$

How to simulate an ε-NFA:

1. Follow transitions as you would for a normal NFA.
2. Take the ε-closure for any states you end up in.

Extending δ to Sets of States

This is similar to normal NFAs. If R is a set of states:

$$\delta(R, a) = \cup_{q \in R}\, \delta(q, a)$$
$$\hat{\delta}(R, w) = \cup_{q \in R}\, \hat{\delta}(q, w)$$

The Language of an ε-NFA

NFAs with ε-transitions define languages similarly to standard NFAs:

If M is an NFA with ε-transitions, then:

$$L(M) \equiv \{w \mid \hat{\delta}(q_0, w) \text{ contains a state in } F\}$$

More good news: any language that can be recognized by an ε-NFA can also be recognized by an NFA without ε-transitions.

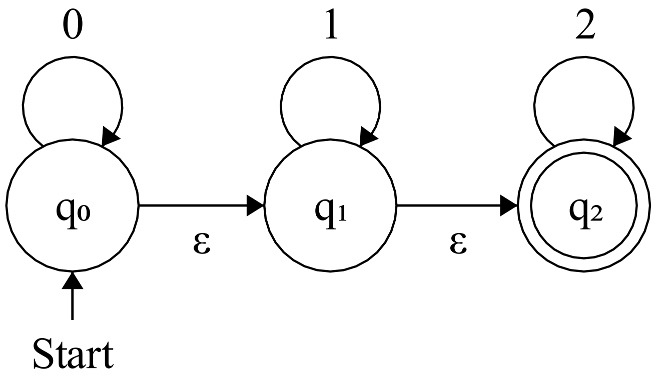
Eliminating ε-Transitions

- The textbook shows how to transform an NFA with ε-transitions to a DFA.
- We will instead show how to transform an NFA with ε transitions into an NFA without ε-transitions.
  - You could then transform such an NFA into a DFA using subset construction.

Eliminating ε-Transitions

- Let E be an NFA with ε-transitions: $(Q, \Sigma, \delta, q_0, F)$
- Define N to be an NFA without ε-transitions. $N = (Q, \Sigma, \delta', q_0, F')$, where:
  - $\delta'(q, a) \equiv \hat{\delta}(q, a)$
  - $F' \equiv \begin{cases} F \cup \{q_0\} & \text{, if } ECLOSE(q_0) \text{ contained a state in } F \\ F & \text{, otherwise} \end{cases}$

Eliminating ε-Transitions: Example



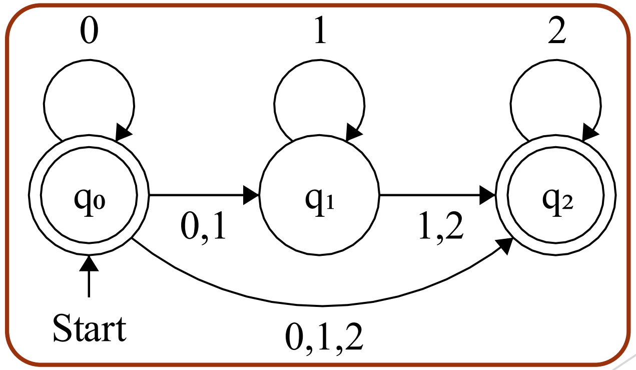
$N = (Q, \Sigma, \delta', q_0, F')$, where:

- $\delta'(q, a) \equiv \delta(ECLOSE(q), a)$
- $F' \equiv \begin{cases} F \cup \{q_0\}, & \text{if } ECLOSE(q_0) \text{ contained a state in F} \\ F & \text{, otherwise} \end{cases}$

$\delta' =$

	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	$\emptyset$	$\{q_1, q_2\}$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$	$\{q_2\}$

$F' = \{q_0, q_2\}$



Theorem 2.22

Theorem 2.22 (from the textbook):
Language L is accepted by an ε-NFA E if and only if it is accepted by some DFA D.

- **If**: Proving this is easy (see Theorem 2.22 in the book)
- **Only if**: The textbook directly constructs a DFA D from ε-NFA E, but we will instead construct an ordinary NFA N, and use Theorem 2.11 to conclude that Theorem 2.22 holds. We start by defining N as it was on the preceding slides.

First Claim in Proof of Theorem 2.22

Reminder:
δ' is for the NFA N
δ is for the ε-NFA E

Rather than starting with a claim about $L(E)$ or $L(N)$, we instead claim that $\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, x)$ for some string x.

$N = (Q, \Sigma, \delta', q_0, F')$, where:

- $\delta'(q, a) \equiv \delta(ECLOSE(q), a)$
- $F' \equiv \begin{cases} F \cup \{q_0\}, & \text{if } ECLOSE(q_0) \text{ contained a state in F} \\ F, & \text{otherwise} \end{cases}$

We will prove this claim using induction on $|x|$.

- **Note**: This may *not* hold for $|x| = 0$. For example, in the previous example $\hat{\delta}(q_0, \varepsilon) = \{q_0, q_1, q_2\}$, but $\hat{\delta}'(q_0, \varepsilon) = \{q_0\}$.
- We instead use $|x| = 1$ as our base case (next slide).

Proof of Theorem 2.22: Base Case

Reminder:
δ' is for the NFA N
δ is for the ε-NFA E

$N = (Q, \Sigma, \delta', q_0, F')$, where:

- $\delta'(q, a) \equiv \delta(ECLOSE(q), a)$
- $F' \equiv \begin{cases} F \cup \{q_0\}, & \text{if } ECLOSE(q_0) \text{ contained a state in F} \\ F, & \text{otherwise} \end{cases}$

Base case: $|x| = 1$

For any symbol a, $\hat{\delta}'(q_0, a) = \hat{\delta}(q_0, a)$, by the definition of δ'.

Claim:
$\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, x)$

Proof of Theorem 2.22: Inductive Step

Reminder:
δ' is for the NFA N
δ is for the ε-NFA E

Let $x = wa$, where w is a string and a is a symbol.
We must show that $\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, x)$.

$N = (Q, \Sigma, \delta', q_0, F')$, where:
- $\delta'(q, a) \equiv \delta(ECLOSE(q), a)$
- $F' \equiv \begin{cases} F \cup \{q_0\}, & \text{if } ECLOSE(q_0) \text{ contained a state in F} \\ F, & \text{otherwise} \end{cases}$

Claim:
$\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, x)$

By the inductive hypothesis, $\hat{\delta}'(q_0, w) = \hat{\delta}(q_0, w)$.

$\hat{\delta}'(q_0, wa)$	
$= \delta'(\hat{\delta}'(q_0, w), a)$	, by the inductive definition of $\hat{\delta}'$
$= \bigcup_{q \in \hat{\delta}'(q_0, w)} \delta'(q, a)$	, by the definition of δ' for sets of states
$= \bigcup_{q \in \hat{\delta}(q_0, w)} \hat{\delta}(q, a)$	, by the inductive hypothesis and definition of δ'
$= \hat{\delta}(\hat{\delta}(q_0, w), a)$	, by the definition of $\hat{\delta}$ for ε-NFAs and sets of states
$= \hat{\delta}(q_0, wa)$	, by the definition of $\hat{\delta}$

Proof of Theorem 2.22: Finishing Up

Reminder:
δ' is for the NFA N
δ is for the ε-NFA E

To show that $L(N) = L(E)$ we must show that $\hat{\delta}'(q_0, x)$ contains a state in F iff $\hat{\delta}(q_0, x)$ contains a state in F.

Additionally, we need to deal with $|x| = 0$.

Case where $\boldsymbol{x = \varepsilon}$:

- $\hat{\delta}'(q_0, \varepsilon) = \{q_0\}$
- $\hat{\delta}(q_0, \varepsilon) = ECLOSE(q_0)$
- q_0 is in F' if and only if $ECLOSE(q_0)$ contains a state in F, by the definition of F'.

Proof of Theorem 2.22: Finishing Up

Reminder:
δ' is for the NFA N
δ is for the ε-NFA E

Case where $x \neq \varepsilon$:

- By the inductive proof earlier, $\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, x)$.
- If $F' = F$ or $q_0 \notin \hat{\delta}'(q_0, x)$, we are done.
- Otherwise, $q_0 \in F'$, $q_0 \notin F$, and $q_0 \in \hat{\delta}'(q_0, x)$.
  - In this case, some state in $ECLOSE(q_0)$ is in F.
  - By construction of $\hat{\delta}$, that state is in $\hat{\delta}(q_0, x)$.