

Finite State Machines

February 11, 2009

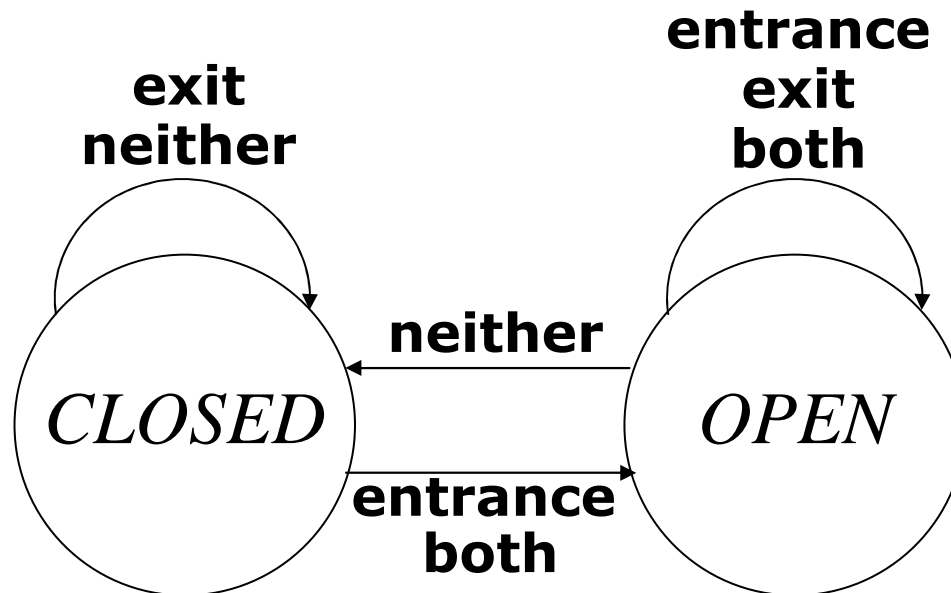
Finite State Automata

- A **finite state automaton** (or **FSA**) is:
 - a finite set of states...
 - ...one of which is a start state...
 - ...at least one of which is an end state...
 - ...with a finite number of ways to transition from one state to another.
- FSAs are used to model procedures, machines, etc.

Example: Automatic Doors

		<u>Input Signals</u>			
		entrance	exit	both	neither
<u>States</u>	<i>CLOSED</i>	<i>OPEN</i>	<i>CLOSED</i>	<i>OPEN</i>	<i>CLOSED</i>
	<i>OPEN</i>	<i>OPEN</i>	<i>OPEN</i>	<i>OPEN</i>	<i>CLOSED</i>

- Finally: we can specify this graphically:



Regular Languages

- Let's work towards a formal definition of a finite state automata (FSAs).
- Definition: a **regular language** is a language that can be modeled with a deterministic FSA.

Long-Term Goal

Determine how powerful a system we need to model natural language.

In particular: determine whether natural languages are regular languages.

Strings and Alphabets

Definition: An **alphabet** is any finite set. The elements of an alphabet are called **symbols**.

Definition: A **string over an alphabet** is a finite sequence of symbols from the alphabet.

Definition: The **length** of a string x , written $|x|$, is the number of symbols in the string.

Definition: The **empty string** (ϵ) is the string of length zero.

Definition: String y is a **substring** of string x if y appears consecutively within x .

Finite State Automata: A Formal Definition

Definition: A **finite state automaton** is a 5-tuple, $\langle Q, \Sigma, \delta, s, F \rangle$, in which:

- a. Q is a finite set of states
- b. Σ is the alphabet
- c. δ is the transition function
- d. $s \in Q$ is the start state
- e. $F \subseteq Q$ is the set of accept states (or “end states”)

Finite State Automata: A Formal Definition

For example, the cola machine:

$CM = \langle Q, \Sigma, \delta, s, F \rangle$, where:

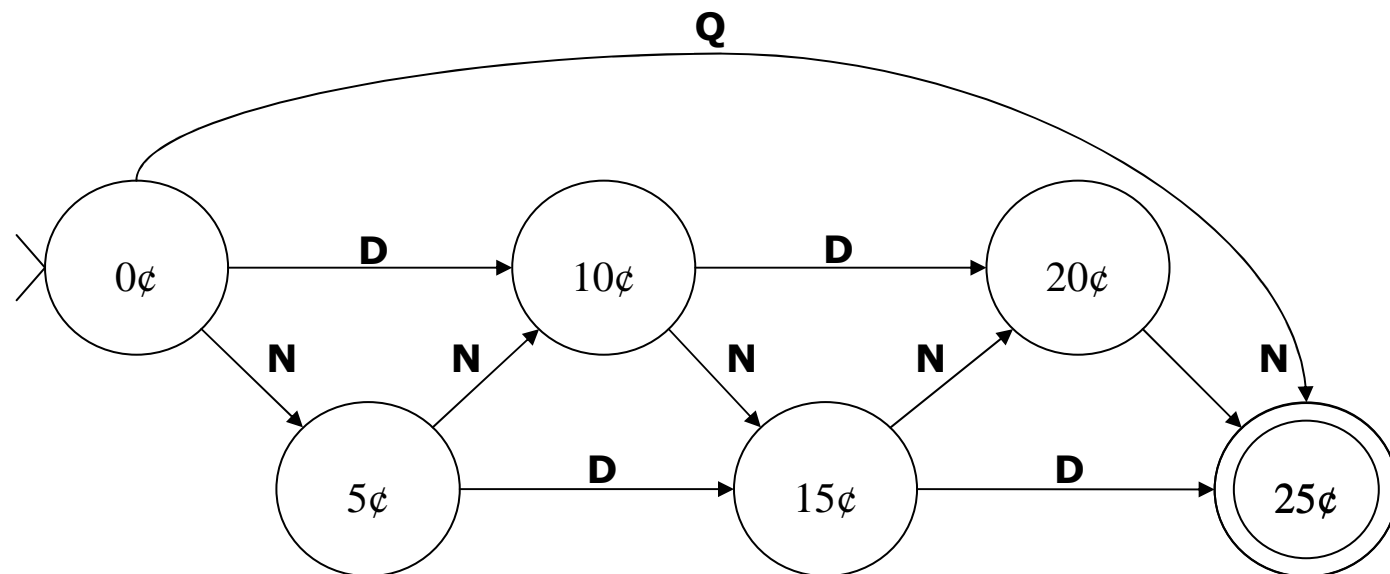
a. $Q = \{0\text{¢}, 5\text{¢}, 10\text{¢}, 15\text{¢}, 20\text{¢}, 25\text{¢}\}$

b. $\Sigma = \{\mathbf{Q}, \mathbf{D}, \mathbf{N}\}$

c. $\delta = \dots$

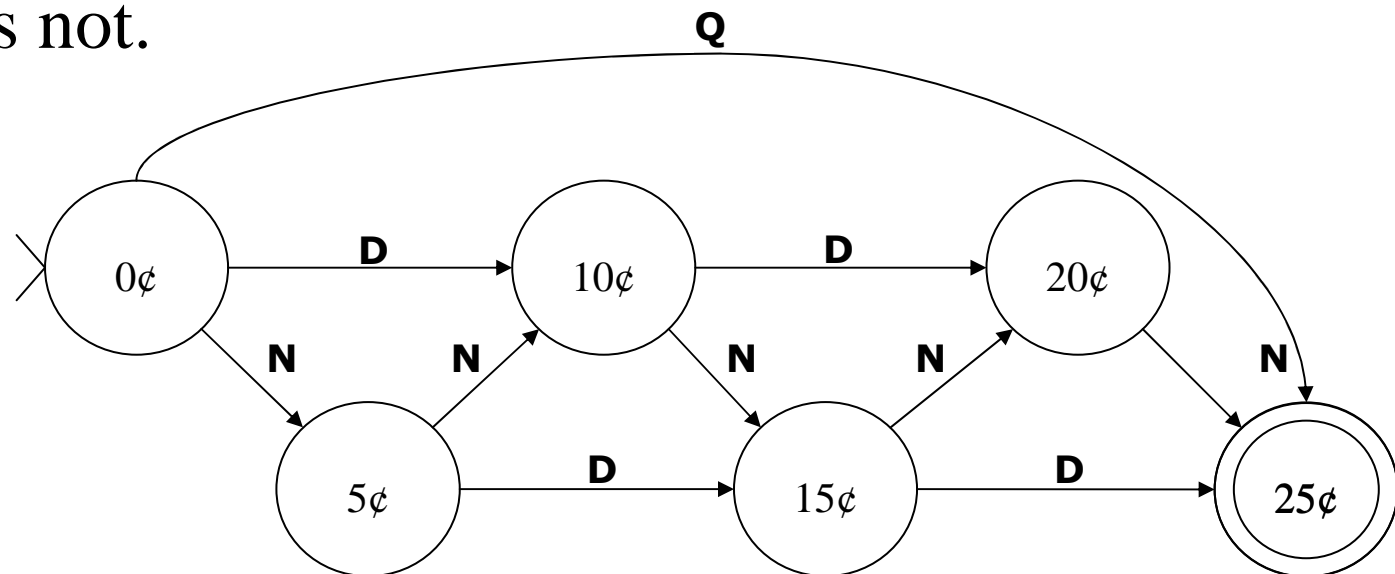
d. $s = 0\text{¢}$

e. $F = \{25\text{¢}\}$



Reminder: the connection to language

- One way to think of CM: it takes a series of coins and either accepts them because they add to 25¢, or rejects them because they don't.
- Another way to think of CM: it takes a string (i.e., a series of letters) and either accepts them as part of a language or rejects them as not.
- Goal: (try to) create a machine “E” that takes a string (i.e. a series of words) and either accepts them as part of English or rejects them as not.



FSAs: Accepting Strings

- A language is a (possibly infinite) set of (finite-length) strings. Thus, we can connect the strings of the language to the FSA that models the language:

Definition: A FSA **accepts** a string x if there is a path from the start state to an accept state that moves along transitions labeled with the symbols in the string, in order.

It **rejects** the string if there is no such path.

FSA and Languages

- So: an FSA has an alphabet (\approx lexicon) and a set of strings over that alphabet that it accepts (\approx language).
To be explicit:

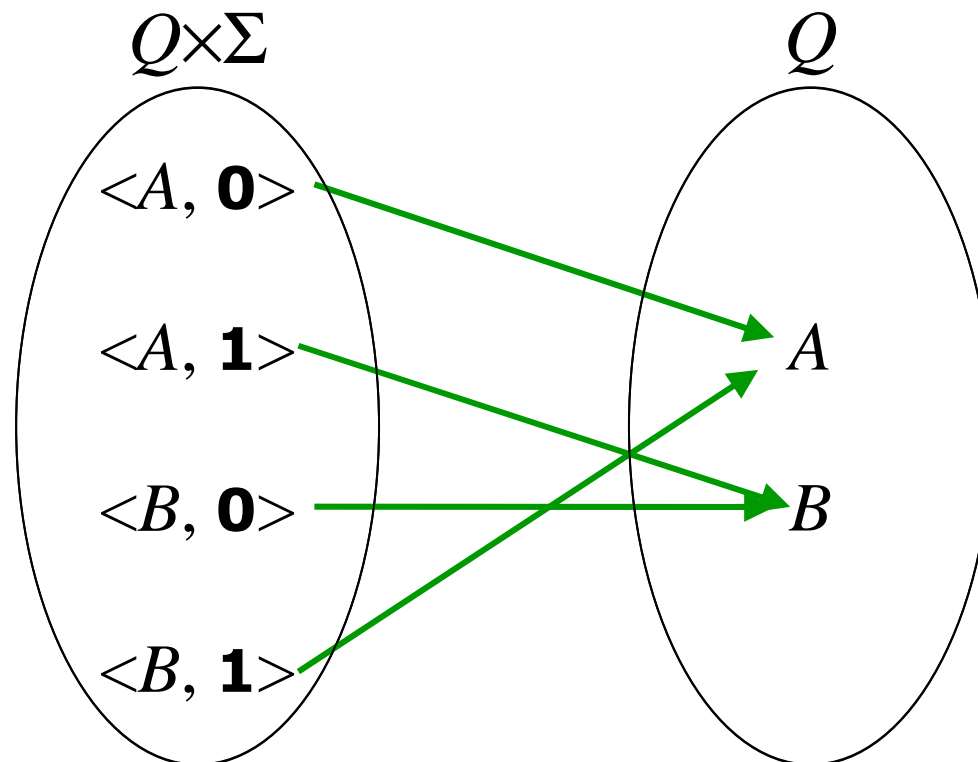
Definition: The language **modeled** (or **defined**) by a finite state automaton M , written $L(M)$, is the set of strings M accepts.

- Thus, a FSA acts as a syntax for a language: it provides an algorithm that produces all (and only) the strings in the language.

The Transition Function

Definition: A **transition function** δ is a function
 $Q \times \Sigma \rightarrow Q$

e.g., suppose $Q = \{A, B\}$, and $\Sigma = \{\mathbf{0}, \mathbf{1}\} \dots$



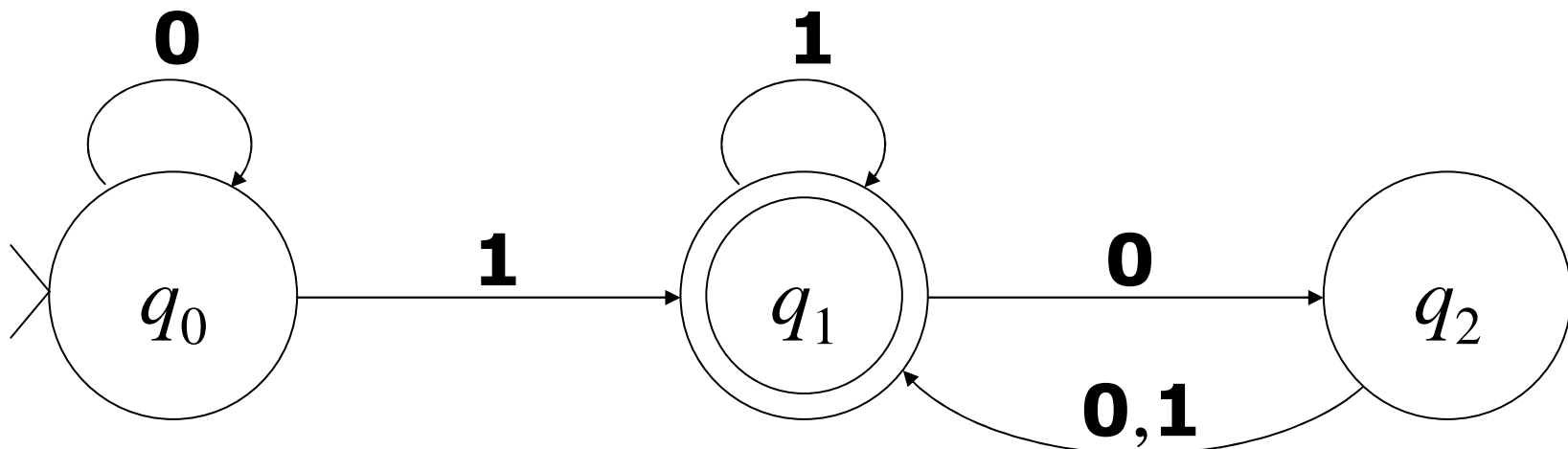
The Transition Function

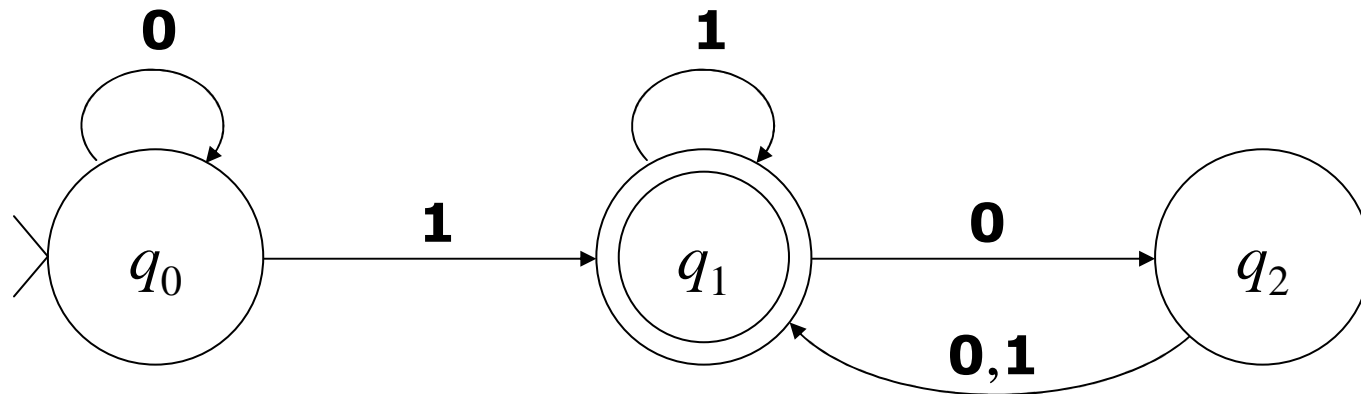
Definition: A **transition function** δ is a function
 $Q \times \Sigma \rightarrow Q$

- In particular, it specifies the transitions of a FSA, by saying, roughly:
 - If you're in state q and following a transition labelled \mathbf{x} ,
 - i.e., given the ordered pair $\langle q, \mathbf{x} \rangle$
 - then the next state you're going to is state q' .
 - i.e., return q' as the result.
- Because we're looking at DFAs, δ must be an actual function: it must give one and only one “next” state for each state/symbol pair.

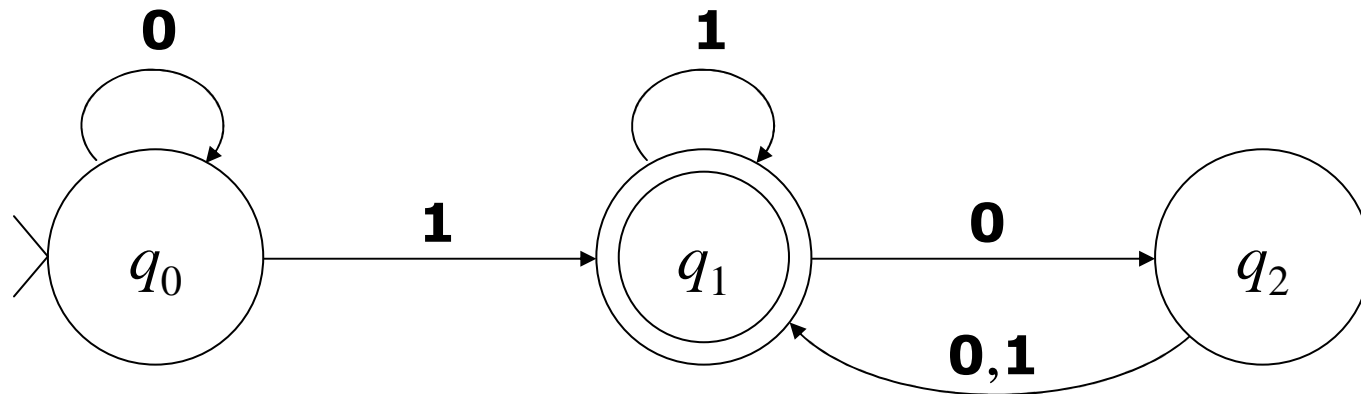
Reading and Constructing State Diagrams

- Given a diagram of a FSA, how can we determine the 5-tuple $\langle Q, \Sigma, \delta, s, F \rangle$?
- Given a 5-tuple $\langle Q, \Sigma, \delta, s, F \rangle$, how can we draw the diagram?
- For instance, the finite state automaton M_1 :





- $M_1 = \langle Q, \Sigma, \delta, s, F \rangle$, where...
- The states of $M_1 = Q = \{q_0, q_1, q_2\}$
- The alphabet of $M_1 = \Sigma = \{0, 1\}$
- The start state of $M_1 = s = q_0$
- The accept states of $M_1 = F = \{q_1\}$
- Note that M_1 is deterministic. For each state, there is one and only one **0** transition leading out, and one and only one **1** transition leading out.



- So what is δ ?

$$\underline{Q \times \Sigma \rightarrow Q}$$

$$\langle q_0, \mathbf{0} \rangle \rightarrow q_0$$

$$\langle q_0, \mathbf{1} \rangle \rightarrow q_1$$

$$\langle q_1, \mathbf{0} \rangle \rightarrow q_2$$

$$\langle q_1, \mathbf{1} \rangle \rightarrow q_1$$

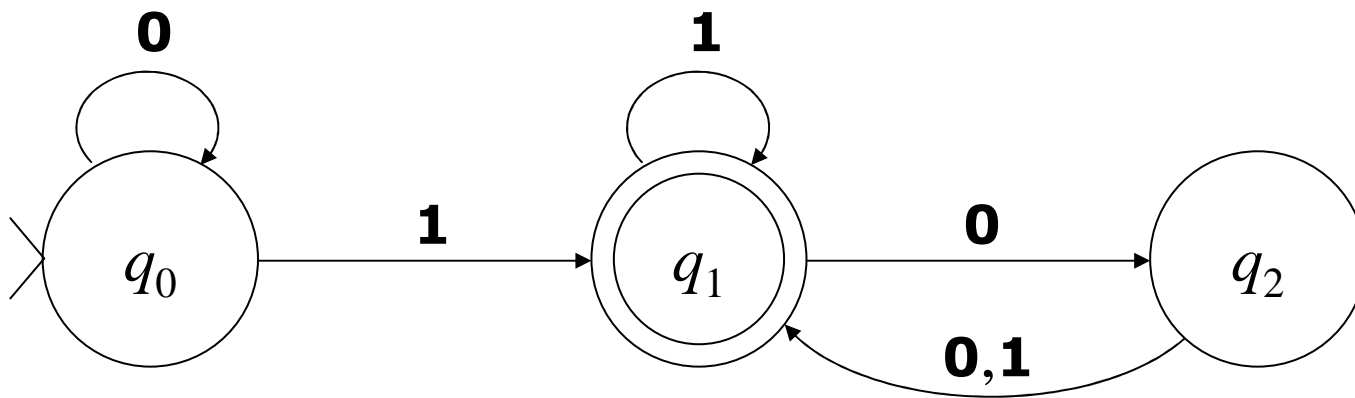
$$\langle q_2, \mathbf{0} \rangle \rightarrow q_1$$

$$\langle q_2, \mathbf{1} \rangle \rightarrow q_1$$

- Or, less formally but more concisely:

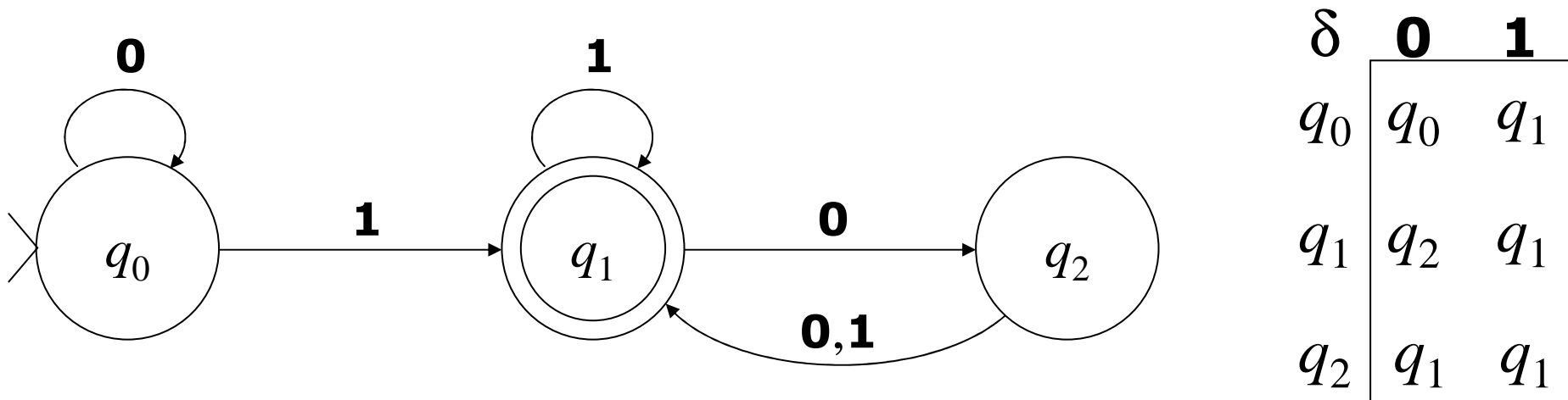
δ	$\mathbf{0}$	$\mathbf{1}$
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_1	q_1

- e.g., $\delta(\langle q_0, \mathbf{0} \rangle) = q_0$, etc.



δ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_1	q_1

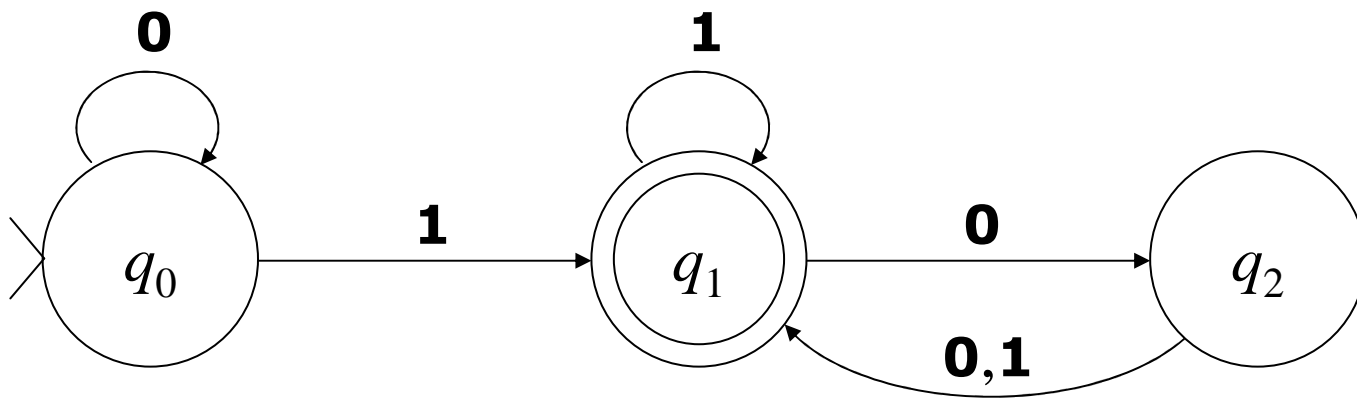
- Any string can be *processed*, to see whether the FSA accepts or rejects it, as follows:
 - Start in the start state, at the beginning of the string
 - Read the next symbol in the string
 - Follow that symbol's transition from the current state to the next state, and make that the new current state
 - If there are symbols left, go to step 2
 - When there are no symbols left:
 - If the current state is an accept state, accept the string.
 - Otherwise, reject the string.



- For example: does M_1 accept the string **1101**?

<i>Current State</i>	<i>Next Symbol</i>	<i>New State</i>
$q_0 (= s)$	1	q_1
q_1	1	q_1
q_1	0	q_2
q_2	1	q_1
q_1	(end)	

- Check: is q_1 an accept state?
- Answer: **yes**. Therefore: accept **1101**.



δ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_1	q_1

• **Question:** which of the following strings are accepted by M_1 ?

- | | | | | | |
|----|-------------------|------------|----|---------------------|------------|
| a. | 1 | yes | g. | 0101000000 | yes |
| b. | 01 | yes | h. | 100 | yes |
| c. | 101000 | no | i. | 0 | no |
| d. | 11 | yes | j. | 0100 | yes |
| e. | 0101010101 | yes | k. | 110000 | yes |
| f. | 10 | no | l. | 111010100000 | no |

Drawing State Diagrams

- **Question:** draw a state diagram of $M_2 = \langle Q, \Sigma, \delta, s, F \rangle$, where:

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

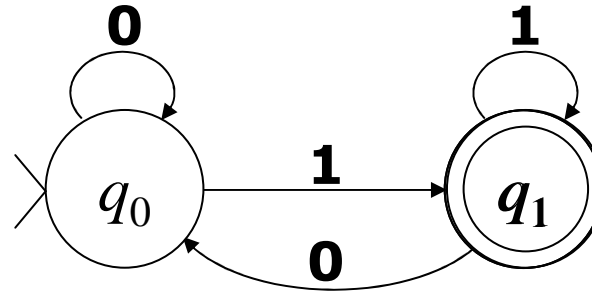
- $\Sigma = \{\mathbf{0}, \mathbf{1}\}$

- δ

	0	1
q_0	q_0	q_1
q_1	q_2	q_1

- $s = q_0$

- $F = \{q_1\}$

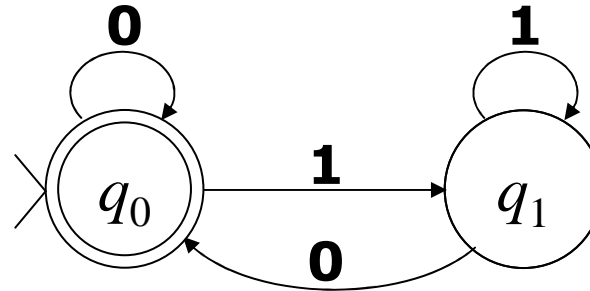


- Which of the following strings does M_2 accept?

- a. **0** **no**
- b. **1** **yes**
- c. **00** **no**
- d. **11111** **yes**
- e. **1000000** **no**
- f. **1010011** **yes**
- g. ϵ **no**

Drawing State Diagrams

- **Question:** give the formal description for the following FSA, M_3 :



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

- $\Sigma = \{0, 1\}$

- δ

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

- $s = q_0$

- $F = \{q_0\}$

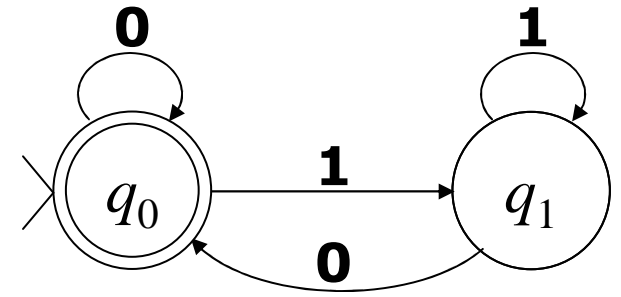
- Which of the following strings does M_3 accept?

- a. **0** **yes**
- b. **1** **no**
- c. **00** **yes**
- d. **11111** **no**
- e. **1000000** **yes**
- f. **1010011** **no**
- g. ϵ **yes**

Understanding State Diagrams

- So what is $L(M_3)$?

- **0**, **00**, **1000000**, $\epsilon \in L(M_3)$
- **1**, **11111**, **1010011** $\notin L(M_3)$



- Generally:

- If the last symbol was **0**, the transition led into q_0 , which is an accept state.
- If the last symbol was **1**, the transition led into q_1 , which is not an accept state.
- In other words: to end in an accept state, the string must end in **0**.

- $L(M_3) = \{x \mid x \text{ ends in } \mathbf{0}\}$

- ...well, almost, because the empty string is also accepted.

- $L(M_3) = \{x \mid x \text{ ends in } \mathbf{0} \text{ or is the empty string}\}$

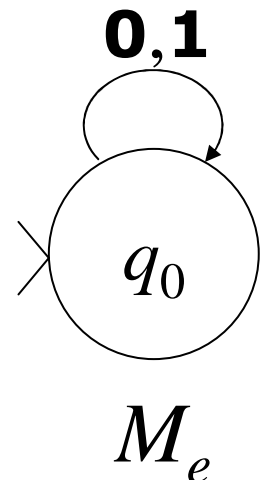
- or, $L(M_3) = \{x \mid x \text{ does not end in } \mathbf{1}\}$, etc.

Regular Languages

Definition: A **regular language** is a language that can be modeled with a DFA.

Definition: The language **modeled** (or **defined**) by a finite state automaton M , written $L(M)$, is the set of strings M accepts.

- Note that *every* DFA defines a language (i.e., a set of strings). Even M_e :
- What is $L(M_e)$?
 - M_e doesn't accept any strings, so...
 - It's the empty language, \emptyset .



Regular Languages

- So how can we tell that a language L is a regular language?
- Construct a DFA that models it!
- For example: show that the following languages are regular (with $\Sigma = \{\mathbf{0}, \mathbf{1}\}$) by providing DFAs that model them:

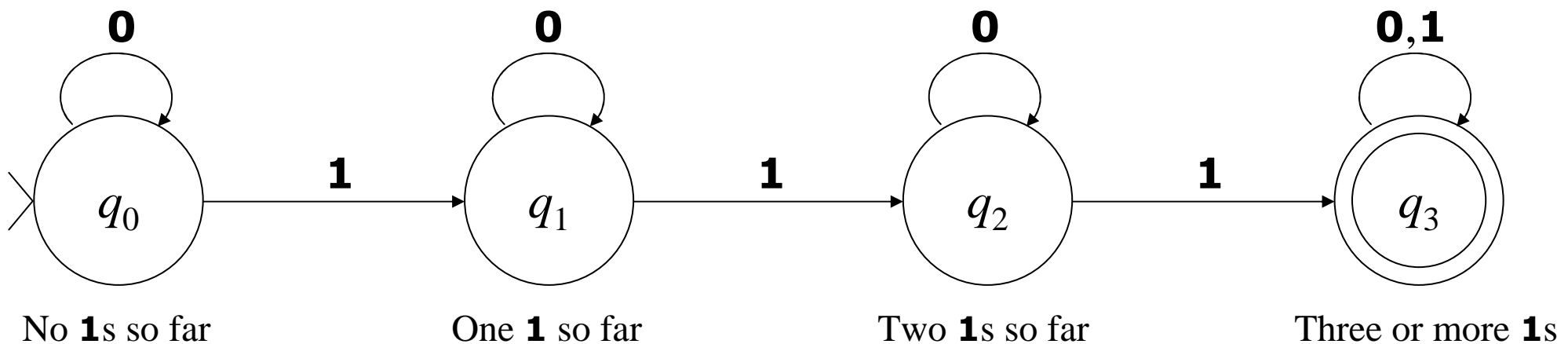
$\{s \mid s \text{ contains at least three } \mathbf{1}s\}$

$\{s \mid s \text{ begins with } \mathbf{1} \text{ and ends with } \mathbf{0}\}$

Regular Languages

$\{s \mid s \text{ contains at least three } \mathbf{1}s\}$

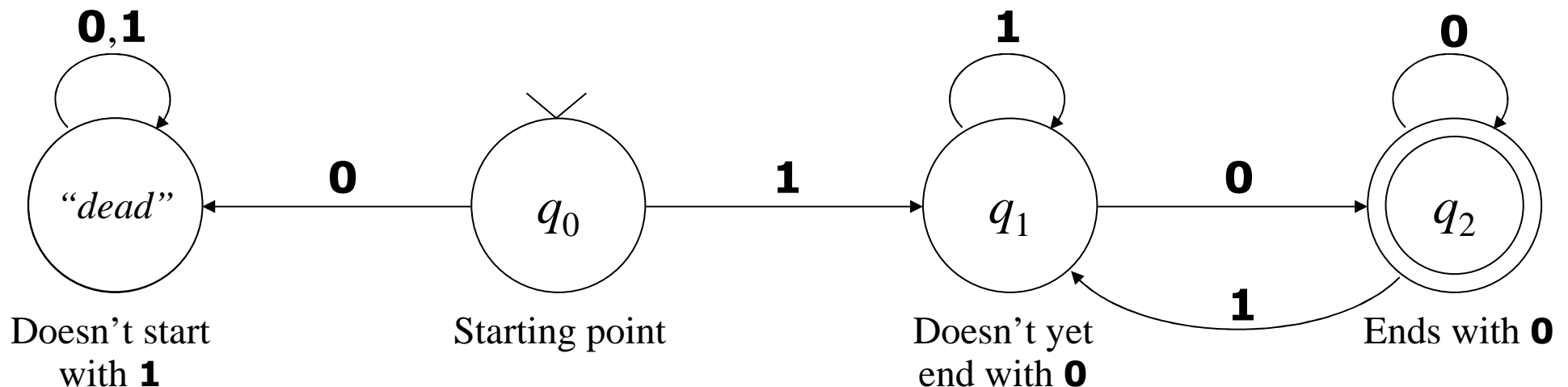
- Suppose we're in the middle of a string. What “states” matter?
 - Some possible situations we could be in: “the last symbol we saw was a **1**”; “the first symbol in the string was a **0**”; “the last two symbols were the same”
 - But the ones that matter here are....



Regular Languages

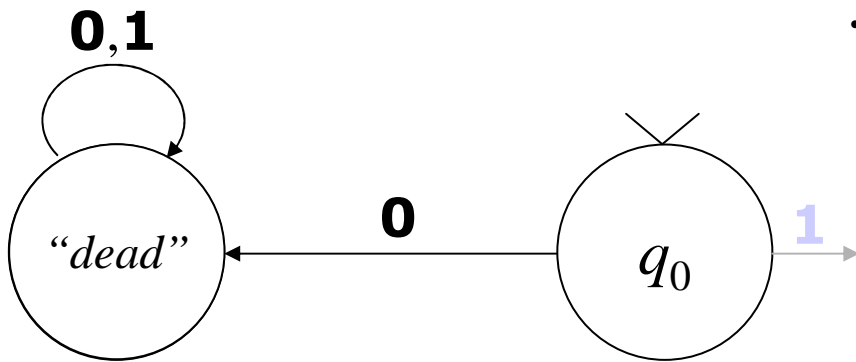
$\{s \mid s \text{ begins with } \mathbf{1} \text{ and ends with } \mathbf{0}\}$

- Once again, start with the states the string might be in....
 - ...though this time, we'll do it in two parts.
- Now, what happens if you're at q_0 and the symbol you see is a **0**?



“Dead” States

- Why a “dead” state?
- Because in this language, it’s possible to reach a point in the string where it’s “too late”: it’s already not part of the language, and adding more symbols won’t help.
 - That is to say, it should be impossible to reach an accept state at this point.
 - Compare this to the “ends with **0**” part: even if the string so far doesn’t end with **0**, it’s still could.

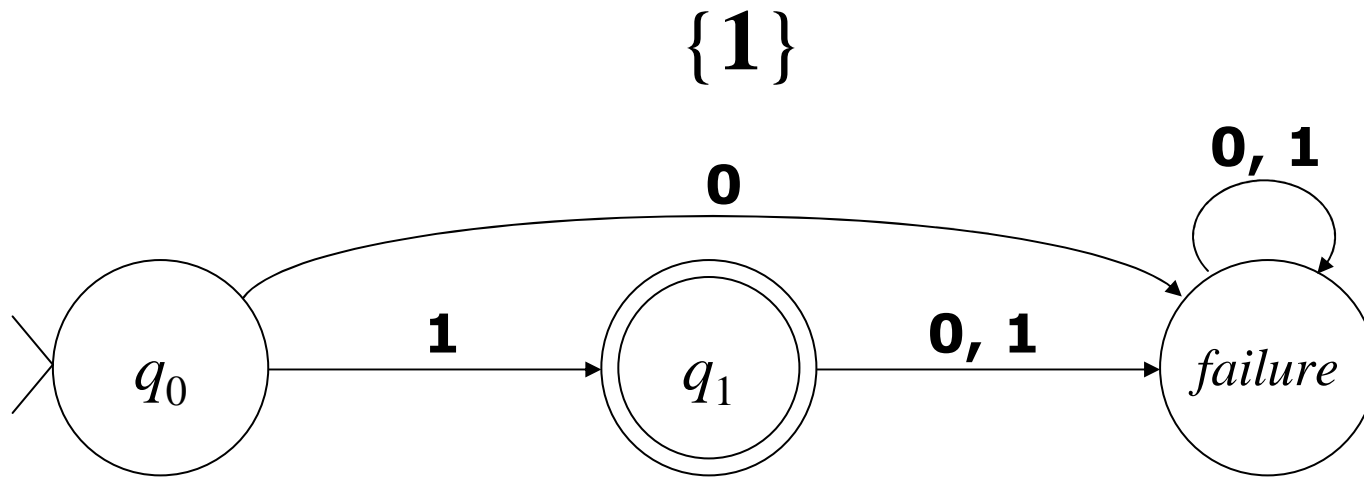


- Or similarly for the “at least three **1**s” language: adding more symbols could make the string part of the language.

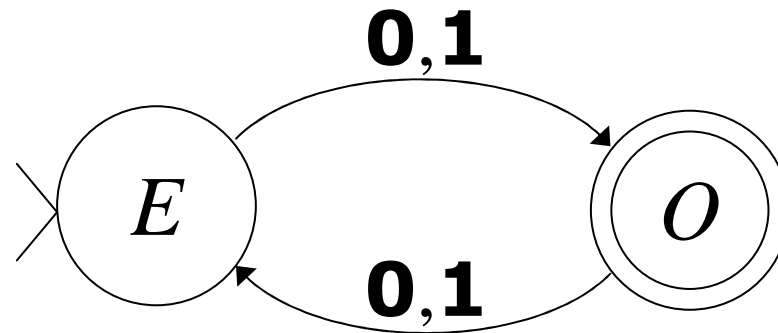
$\{s \mid s \text{ begins with } \mathbf{1} \text{ and ends with } \mathbf{0}\}$

Please pair up with a partner.
(No, really.)

Regular Languages



$\{s \mid \text{the length of } s \text{ is odd}\}$



M_A

Regular Operations

Let A and B be languages with the same alphabet Σ .

- e.g., $P = \{\mathbf{10}, \mathbf{11}\}$ and $Q = \{\mathbf{10}, \mathbf{100}, \mathbf{1000}\}$

Definition: The **union** of A and B is the language
 $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

- $P \cup Q = \{\mathbf{10}, \mathbf{11}, \mathbf{100}, \mathbf{1000}\}$

Definition: The **intersection** of A and B is the
language $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

- $P \cap Q = \{\mathbf{10}\}$

Regular Operations

$$P = \{\mathbf{10}, \mathbf{11}\} \text{ and } Q = \{\mathbf{10}, \mathbf{100}, \mathbf{1000}\}$$

Definition: The **concatenation** of A and B is the language $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

- $P \circ Q = \{\mathbf{1010}, \mathbf{10100}, \mathbf{101000}, \mathbf{1110}, \mathbf{11100}, \mathbf{111000}\}$

Definition: The **star** of A is the language

$$A^* = \{x_1x_2\dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$$

- $P^* = \{\epsilon, \mathbf{10}, \mathbf{11}, \mathbf{1010}, \mathbf{1011}, \mathbf{1110}, \mathbf{1111}, \mathbf{101010}, \mathbf{101011}, \mathbf{101110}, \mathbf{101111}, \mathbf{111010}, \mathbf{111011}, \mathbf{111110}, \mathbf{111111}, \dots\}$

Properties of Regular Operations

Definition: A set S is **closed under** an operation x if, when the inputs to x are members of S , the output is a member of S .

- For instance:
 - The set of positive integers is closed under the operation “addition”: if x and y are positive integers, then $x+y$ is a positive integer.
 - The set of positive integers is **not** closed under the operation “subtraction”.
 - The set of all integers **is** closed under subtraction.

Properties of Regular Operations

- Thus: Four theorems
 - The class of regular languages is closed under the union operation.
 - (i.e., the union of two regular languages is also a regular language)
 - The class of regular languages is closed under the intersection operation.
 - (i.e., the intersection of two regular languages is etc.)
 - The class of regular languages is closed under the concatenation operation.
 - The class of regular languages is closed under the star operation.
- Of course, we still need to prove these.

Proof of Closure Under Union

- **Given:** A, B are regular languages.
- **To prove:** $A \cup B$ is a regular language.
- Which is to say:
 - **Given:** There is a FSA M_A that models A , and a FSA M_B that models B .
 - **To prove:** There is a FSA M_{AB} that models $A \cup B$.
- Thus: if we can provide a way to construct M_{AB} from any arbitrarily chosen M_A and M_B , we've proven that the set of regular languages is closed under union.

Proof of Closure Under Union

- How can we construct such a machine?
 - The idea: the machine should check both machines “in parallel”.
 - As it goes through the string, it should keep track of where M_A would be and where M_B would be.
 - That is, it needs to “remember” a pair of states.
 - So: the states of M_{AB} will be...
 - ...the Cartesian product of the states of M_A and M_B .
- And now, the method...

Proof of Closure Under Union

- By definition:

A is modeled by $M_A = \langle Q_A, \Sigma_A, \delta_A, s_A, F_A \rangle$

B is modeled by $M_B = \langle Q_B, \Sigma_B, \delta_B, s_B, F_B \rangle$

- Then let $M_{AB} = \langle Q_{AB}, \Sigma_{AB}, \delta_{AB}, s_{AB}, F_{AB} \rangle$, where...

- $Q_{AB} = Q_A \times Q_B = \{ \langle q_i, q_j \rangle \mid q_i \in Q_A \text{ and } q_j \in Q_B \}$

- $\Sigma_{AB} = \Sigma_A = \Sigma_B$

- δ_{AB} is on the next slide.

- $s_{AB} = \langle s_A, s_B \rangle$

- $F_{AB} = \{ \langle q_i, q_j \rangle \mid q_i \in F_A \text{ or } q_j \in F_B \}$

- That is, the set of pairs representing states in which either M_A or M_B would accept the string.

Proof of Closure Under Union

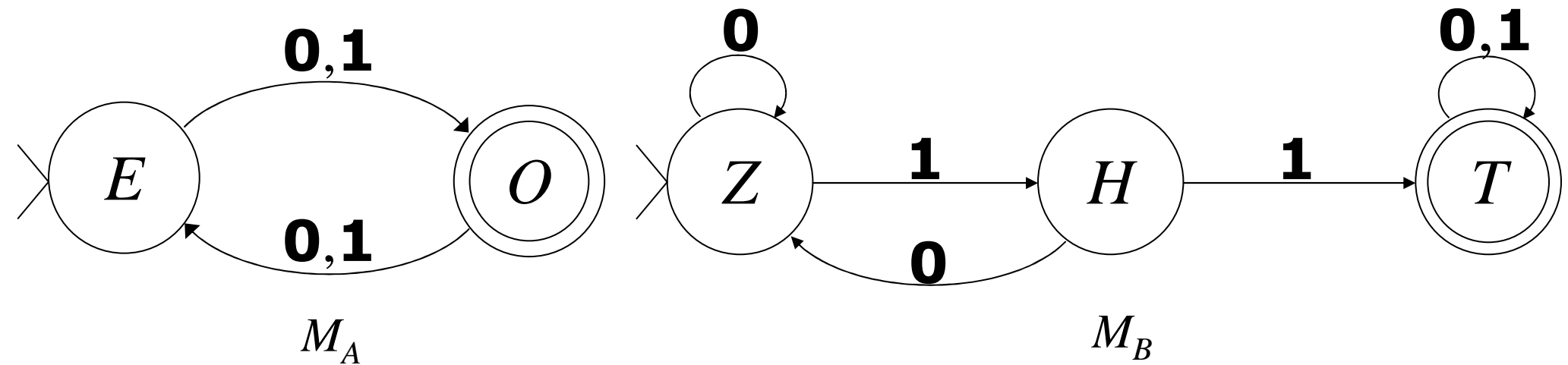
- So what about δ_{AB} ?
- For each $\langle q_a, q_b \rangle \in Q_{AB}$ and $x \in \Sigma$:
$$\delta_{AB}(\langle q_a, q_b \rangle, x) = \langle \delta_A(q_a, x), \delta_B(q_b, x) \rangle$$
- In other words: suppose we're at some state in Q_{AB} , which represents a state from M_A and a state from M_B ; and the symbol we read is x .
- We transition to the state that represents
 - the state x takes us to in M_A , and
 - the state x takes us to in M_B .

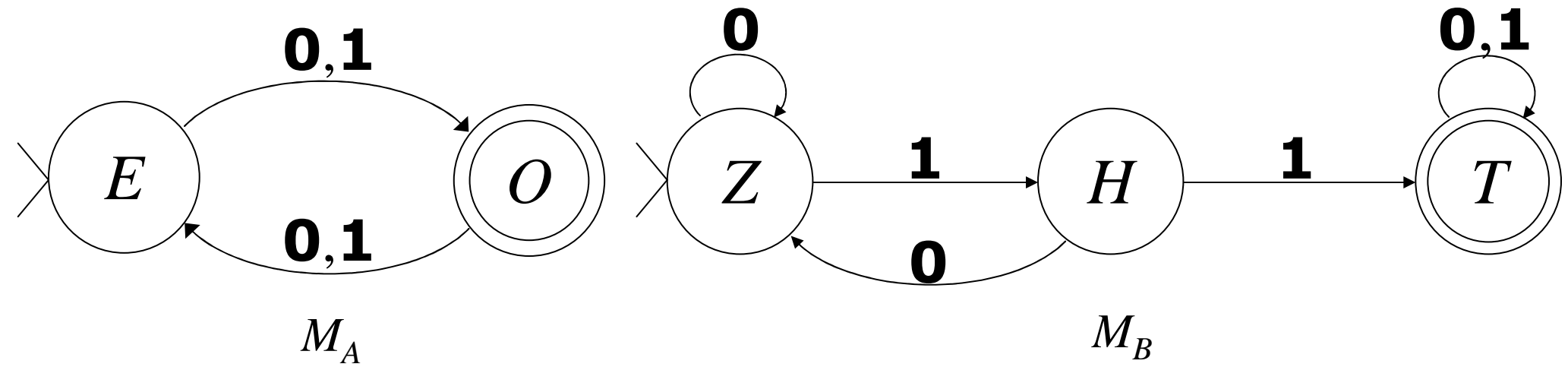
Proof of Closure Under Union

- Given these definitions: M_{AB} is a finite state automaton:
 - It has a finite number of states,
 - a finite alphabet
 - etc.
- And M_{AB} models the language $A \cup B$: it accepts any string that A or B would accept.
- Therefore: $A \cup B$ is a regular language.

For Example...

- $A = \{s \mid \text{the length of } s \text{ is odd}\}$
- $B = \{s \mid \mathbf{11} \text{ is a substring of } s\}$





- $Q_A = \{E, O\}$

- $\Sigma_A = \{\mathbf{0}, \mathbf{1}\}$

- $\delta_A = \begin{array}{c} \mathbf{0} \quad \mathbf{1} \\ E \quad \begin{array}{|c|c|} \hline O & O \\ \hline \end{array} \\ O \quad \begin{array}{|c|c|} \hline E & E \\ \hline \end{array} \end{array}$

- $s_A = E$

- $F_A = \{O\}$

- $Q_B = \{Z, H, T\}$

- $\Sigma_B = \{\mathbf{0}, \mathbf{1}\}$

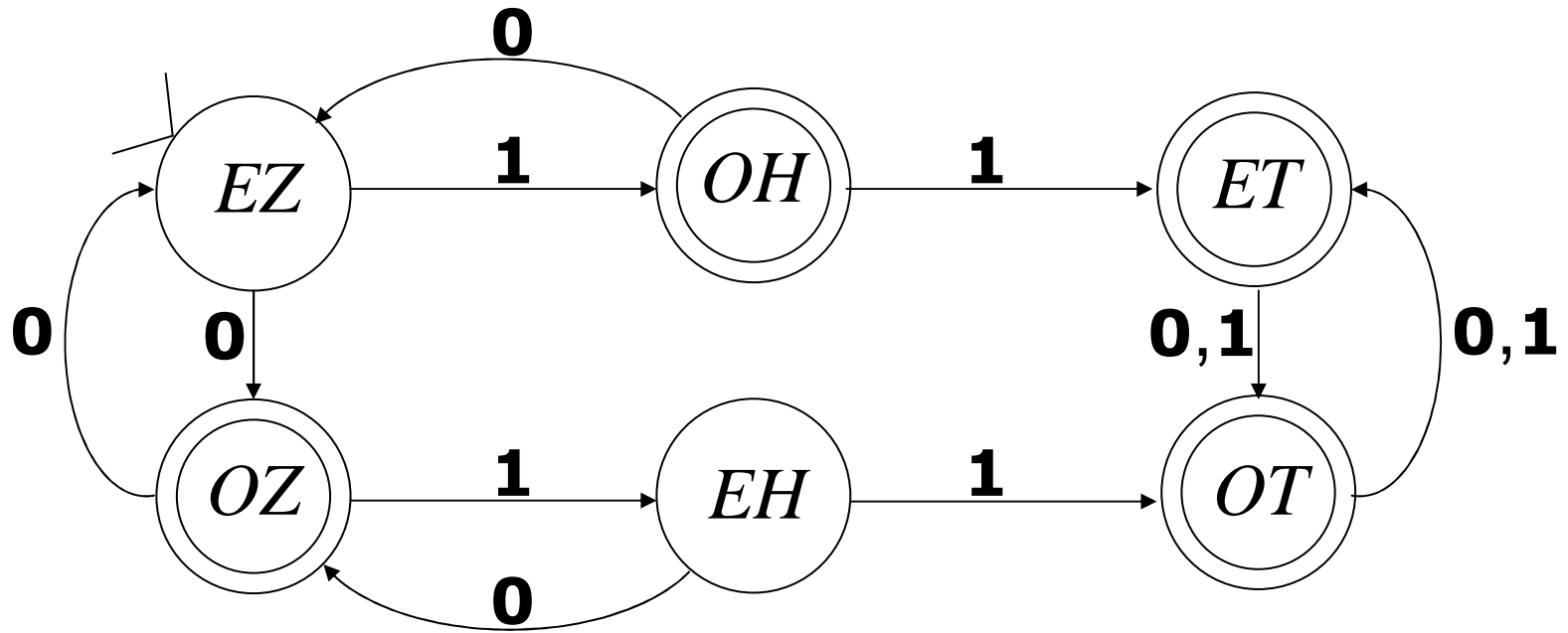
- $\delta_B = \begin{array}{c} \mathbf{0} \quad \mathbf{1} \\ Z \quad \begin{array}{|c|c|} \hline Z & H \\ \hline \end{array} \\ H \quad \begin{array}{|c|c|} \hline Z & T \\ \hline \end{array} \\ T \quad \begin{array}{|c|c|} \hline T & T \\ \hline \end{array} \end{array}$

- $s_B = Z$

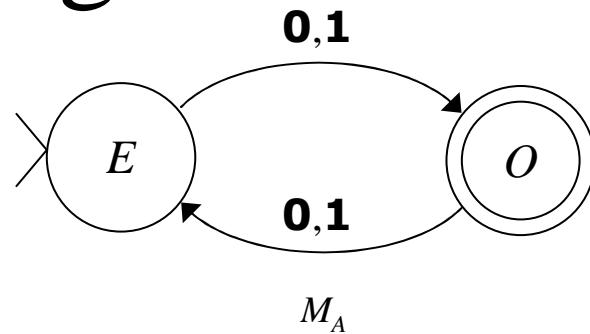
- $F_B = \{T\}$

- $Q_A = \{E, O\}$ $Q_B = \{Z, H, T\}$ • $Q_{AB} = \{\cancel{EZ}, \cancel{EH}, \langle ET, H \rangle, \langle OE, OH, OT \rangle, Z\}$
- $\Sigma_A = \{\mathbf{0}, \mathbf{1}\}$ $\Sigma_B = \{\mathbf{0}, \mathbf{1}\}$ • $\Sigma_{AB} = \{\mathbf{0}, \mathbf{1}\} \{H\}, \langle O, T \rangle\}$
- $s_A = E$ $s_B = Z$ • $s_{AB} = EZ$
- $F_A = \{O\}$ $F_B = \{T\}$ • $F_{AB} = \{OZ, OH, OT, ET\}$
- $\delta_A = \begin{matrix} & \mathbf{0} & \mathbf{1} \\ E & O & O \\ O & E & E \end{matrix}$ $\delta_B = \begin{matrix} & \mathbf{0} & \mathbf{1} \\ Z & Z & H \\ H & Z & T \\ T & T & T \end{matrix}$ • $\delta_{AB} = \begin{matrix} & \mathbf{0} & \mathbf{1} \\ EZ & OZ & OH \\ EH & OZ & OT \\ ET & OT & OT \\ OZ & EZ & EH \\ OH & EZ & ET \\ OT & ET & ET \end{matrix}$

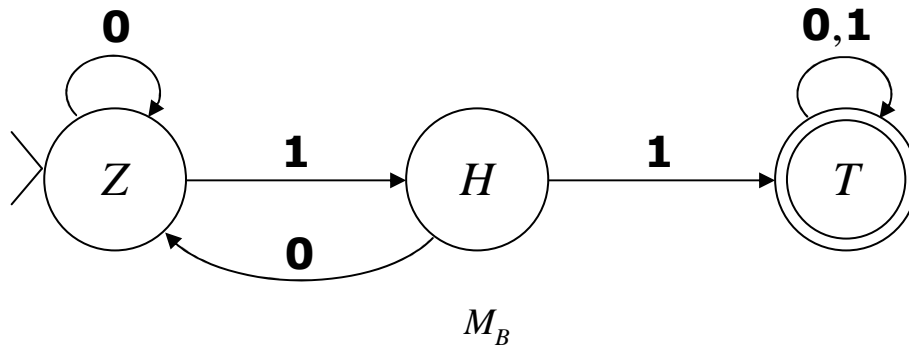
$$M_{AB} = M_A \cup M_B$$



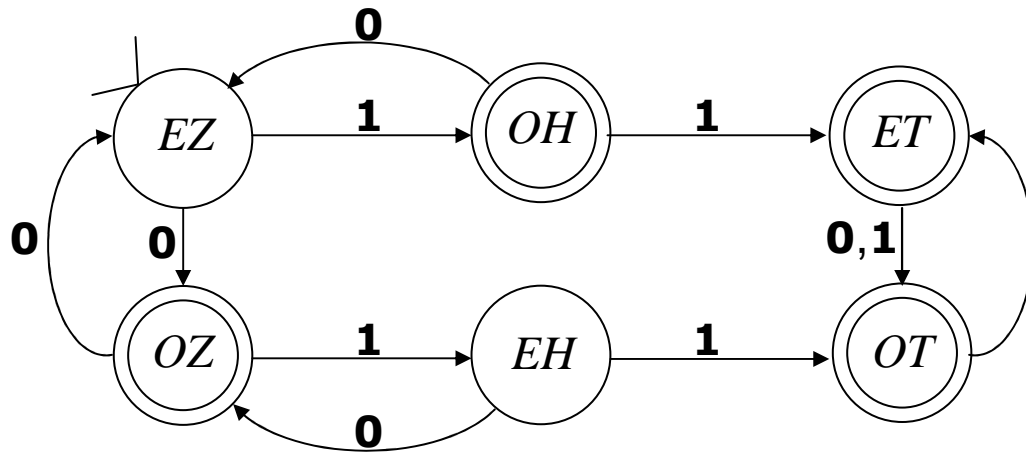
Seeing It In Action: String **0101**



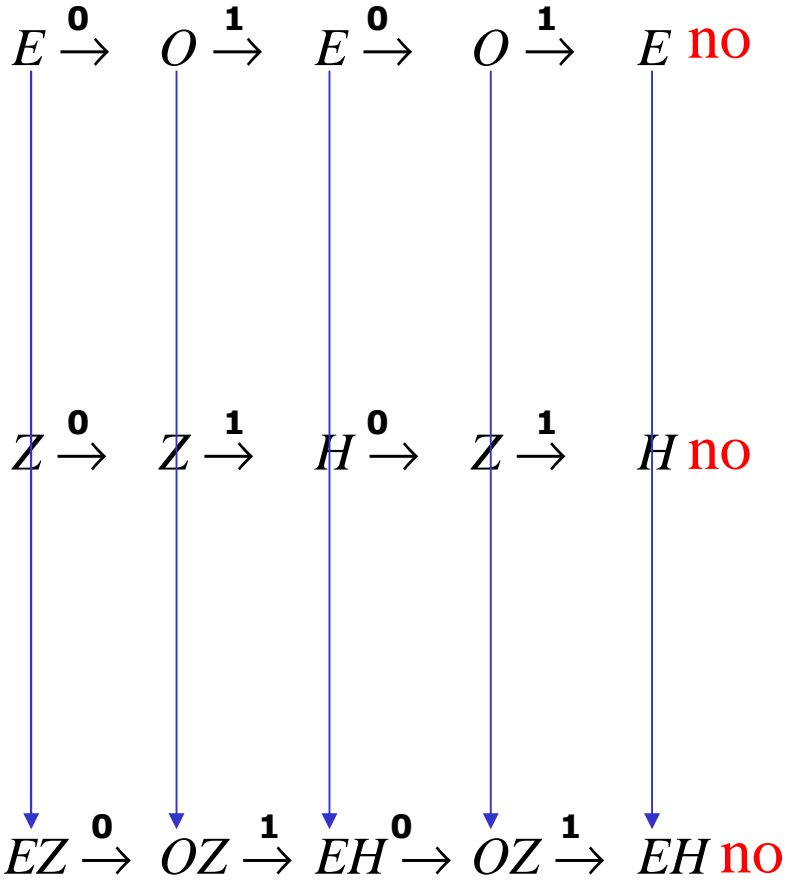
$M_A: E \xrightarrow{0} O \xrightarrow{1} E \xrightarrow{0} O \xrightarrow{1} E$ **no**



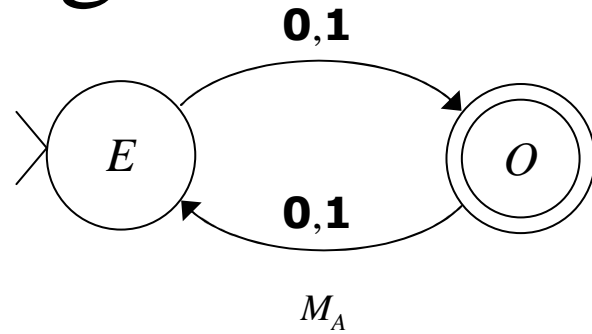
$M_B: Z \xrightarrow{0} Z \xrightarrow{1} H \xrightarrow{0} Z \xrightarrow{1} H$ **no**



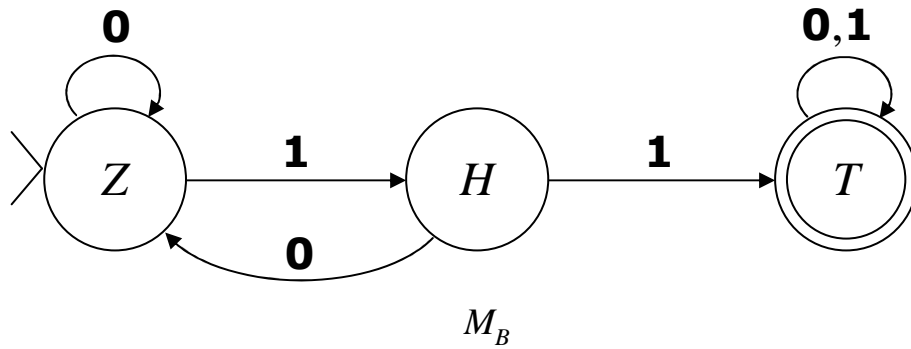
$M': EZ \xrightarrow{0} OZ \xrightarrow{1} EH \xrightarrow{0} OZ \xrightarrow{1} EH$ **no**



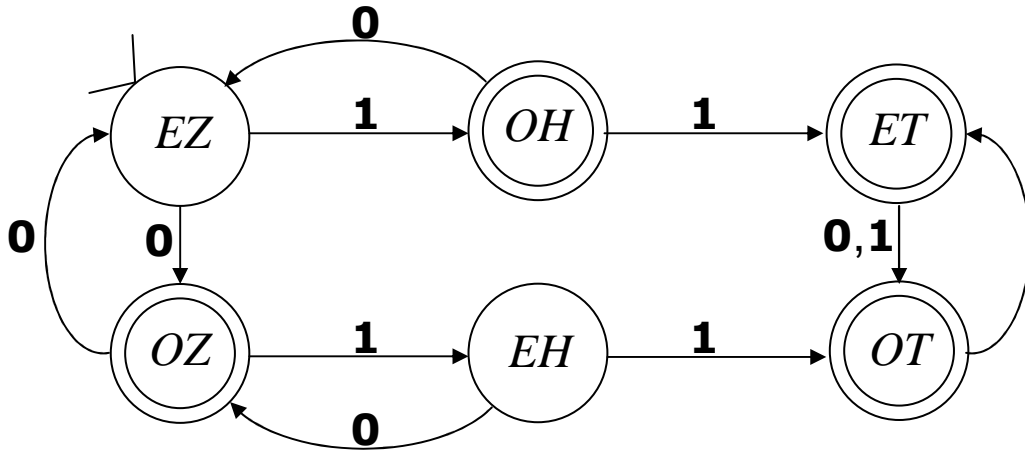
Seeing It In Action: String **0110**



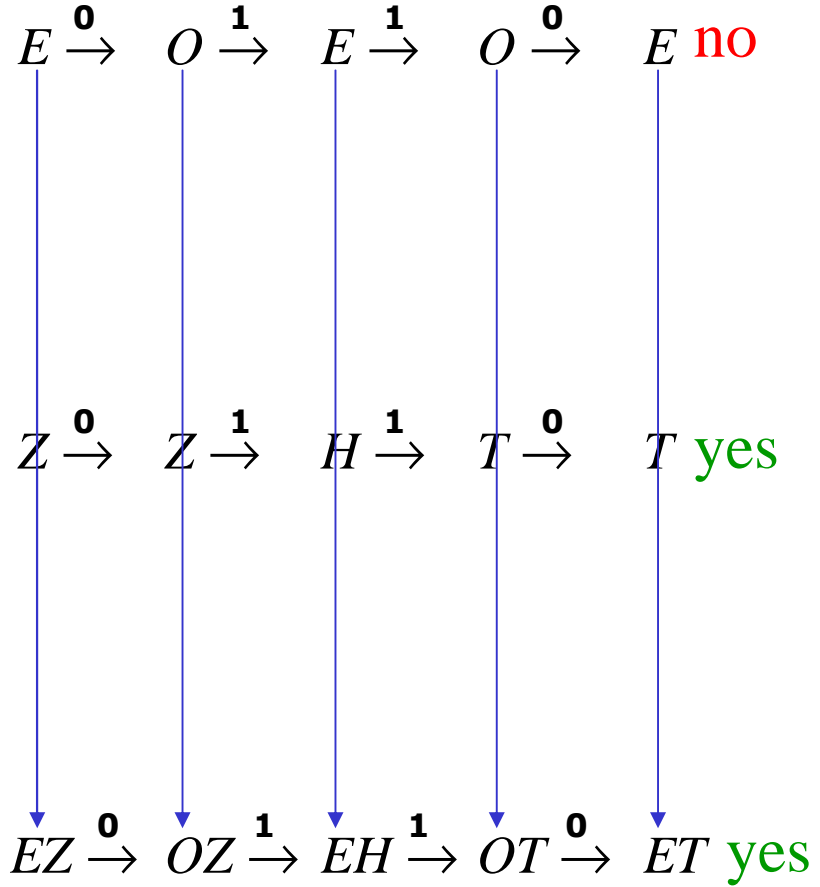
$M_A: E \xrightarrow{0} O \xrightarrow{1} E \xrightarrow{1} O \xrightarrow{0} E$ **no**



$M_B: Z \xrightarrow{0} Z \xrightarrow{1} H \xrightarrow{1} T \xrightarrow{0} T$ **yes**



$M': EZ \xrightarrow{0} OZ \xrightarrow{1} EH \xrightarrow{1} OT \xrightarrow{0} ET$ **yes**

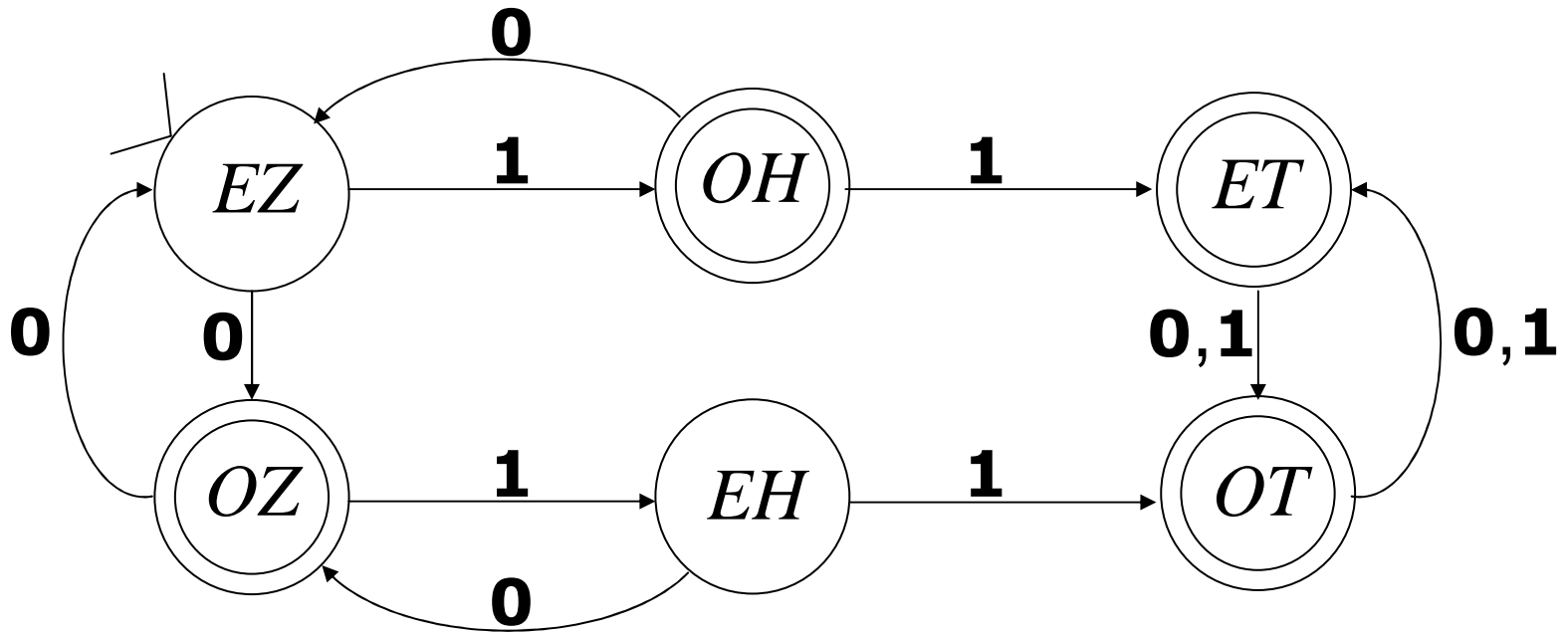


Proof of Closure Under Intersection

- If that was union, what would intersection look like?
 - The idea: much the same! Track through both machines at once.
 - Except that this time...
 - ...accept the string only if you end at a state corresponding to end states of *both* machines, instead of *either* machine.
- That is: let $M_{AB} = \langle Q_{AB}, \Sigma_{AB}, \delta_{AB}, s_{AB}, F_{AB} \rangle$, where...
 - $Q_{AB} = Q_A \times Q_B = \{ \langle q_i, q_j \rangle \mid q_i \in Q_A \text{ and } q_j \in Q_B \}$
 - $\Sigma_{AB} = \Sigma_A = \Sigma_B$
 - For each $\langle q_a, q_b \rangle \in Q_{AB}$ and $x \in \Sigma$:
$$\delta_{AB}(\langle q_a, q_b \rangle, x) = \langle \delta_A(q_a, x), \delta_B(q_b, x) \rangle$$
 - $s_{AB} = \langle s_A, s_B \rangle$
 - $F_{AB} = \{ \langle q_i, q_j \rangle \mid q_i \in F_A \text{ **and** } q_j \in F_B \}$

So instead of...

$$M_{AB} = M_A \cup M_B$$



$$M_{AB} = M_A \cap M_B$$

