

Syntax: Combining Words into Sentences

LING 106

February 4, 2009

1. REMINDER: WHAT IS A LANGUAGE?

A **LANGUAGE** is a set of finite-length strings with meaning, which we can call “sentences”. A **GRAMMAR** is an algorithm that specifies:

- a. a **LEXICON**: a set containing every minimal unit of the language, in the form of 3-tuples consisting of word, its syntactic information, and its semantic information.
- b. a **SYNTAX**: a set of rules that describes how to combine elements of different syntactic categories to form larger units (and, ultimately, sentences)
- c. a **SEMANTICS**: a set of rules for determining the meanings of sentences based on the semantic information of the units and their syntactic combinations

...such that it produces a string S iff S is a sentence of the language

Goal: we want to describe a way to string together words to produce sentences.

(Bearing in mind that lists of possible strings won't work.)

2. FINITE STATE AUTOMATA

A **FINITE STATE AUTOMATON (FSA)** is a finite set of states—one of them a start state, at least one of them an end state—with a finite number of ways to transition from one state to another. (We'll be more formal in the next lecture.) FSAs can be used to model a number of procedures, machines, etc.

2.1. Example: Automatic doors

Automatic doors have two states: *CLOSED* and *OPEN*; and people can be in four positions relative to the door: there can be someone on the **entrance** pad in front of the door, on the **exit** pad on the far side of the door, on **both** pads, or on **neither** pad. To specify the behavior of such a door:

- If the door is *CLOSED*, then...
 - If someone is standing on the **entrance** pad: the door opens.
 - If someone is standing on **both** pads: the door opens.
 - If someone is standing on the **exit** pad: the door stays closed.
 - If someone is standing on **neither** pad: the door stays closed.
- If the door is *OPEN*, then...
 - If someone is standing on the **entrance** pad: the door stays open.
 - If someone is standing on **both** pads: the door stays open.
 - If someone is standing on the **exit** pad: the door stays open.
 - If someone is standing on **neither** pad: the door closes.

Another way to look at “opens” and “closes”: the door transitions from the *CLOSED* state to the *OPEN* state, and vice versa.

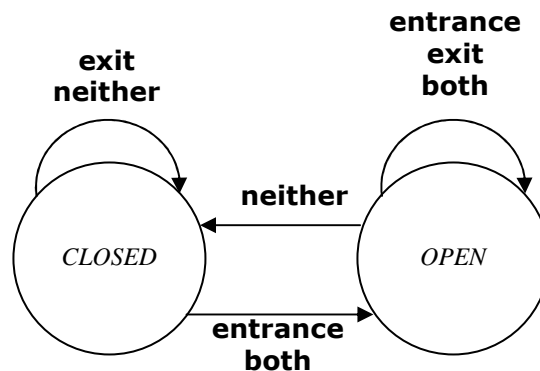
We can specify the behavior in table form, by specifying, for each state, what action to take when given each input signal:

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

or more fully, by specifying, for each state, what state to go to when given each input signal:

		<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>

And finally, we can specify the behavior in graphical form:



(Note: this doesn't actually have a start or an end state, so as a FSA it's a little incomplete.)

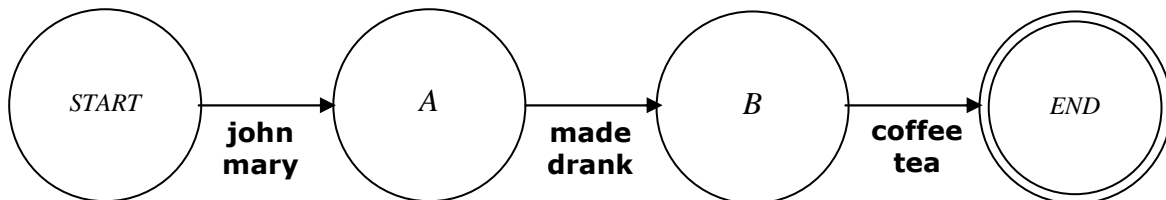
3. GRAMMAR AS A FINITE STATE AUTOMATON

Suppose we label the states arbitrarily, designate one as the starting point and one as the ending point, and have our “input signals” be words. Then we can use a FSA as a grammar. For example:

- If language *F1* is the following set of strings:

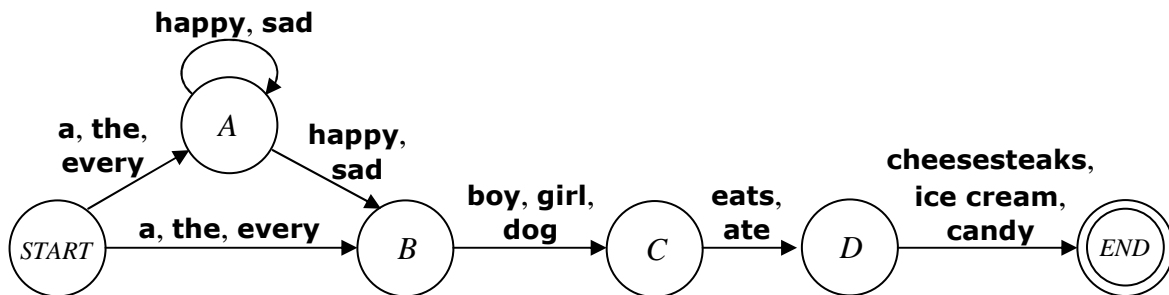
{john made coffee, mary made coffee, john made tea, mary made tea, john drank coffee, mary drank coffee, john drank tea, mary drank tea}

Then a finite state automaton that models it is:



(By convention, accept states are marked with a double circle.) That covers the language. Of course, it’s not a very good language.

We can also make more complex automata. Call the language it describes *F2*.



Question: Which of the following strings are sentences of *F2*? That is, which ones are generated by the above FSA?

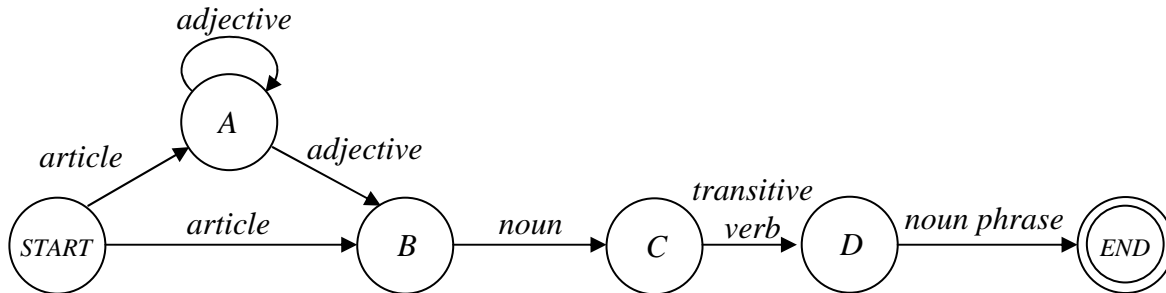
- (1)
- a happy boy eats cheesesteaks**
 - every girl eats a dog**
 - the happy happy happy boy ate ice cream**
 - the happy sad happy sad dog ate ice cream**
 - a happy dog ate happy candy**
 - a girl ate candy**
 - every angry boy ate ice cream**

Question: What is the cardinality of *F2* (i.e., how many sentences does the language have; how many different strings does the above FSA generate)?

Of course...

- Our understanding of language doesn't seem to involve mere strings of words;
- We did all that work earlier in the course to determine distributional classes, i.e. syntactic categories. Wouldn't it be nice to use them?

The above FSA reconsidered as *F3*...



...along with some lexical information.

article **the, a, every**
adjective **happy, sad**
noun **boy, girl, dog**

transitive verb **eats, ate**
noun phrase **ice cream, candy,
cheesesteaks**

4. FINITE STATE AUTOMATA AND REGULAR LANGUAGES

Finite state automata can model some, *but not all*, languages. A **REGULAR LANGUAGE** is a language that can be modeled with a FSA.

Long-Term Goal

Determine how powerful a system we need to model natural language.
In particular: determine whether natural languages are regular.

In a minute, we'll formally define finite state automata. First...

5. STRINGS AND ALPHABETS

Definition: An **ALPHABET** is any finite set. The elements of the alphabet are called **SYMBOLS**.

- $\Sigma_1 = \{0, 1\}$
- $\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$
- $\Sigma_3 = \{article, noun, transitive-verb, adjective, noun-phrase\}$

Definition: A **STRING OVER AN ALPHABET** is a finite sequence of symbols from the alphabet. (Traditionally, if the symbols are single characters, as in Σ_1 and Σ_2 above, the sequence is written by placing the symbols one after the other without spaces.)

- Some strings over Σ_1 : **0110110, 1101001010101010, 1**
- Some strings over Σ_2 : **penelope, syzygy, fnord, htpzorsthg, bbbbbb**
- Some strings over Σ_3 : *article noun, adjective transitive-verb adjective*

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

- $|0110110| = \underline{\quad}?$
- $|\text{penelope}| = \underline{\quad}?$
- $|\text{article noun}| = \underline{\quad}?$

Definition: The **EMPTY STRING**, written ϵ , is the string of length zero.

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

- Some substrings of **penelope**: **pen, lop, nelo, e**

Definition: The **REVERSE** of a string x (x^R) is the string obtained by writing the symbols of x in the opposite order.

- $\text{penelope}^R = \underline{\hspace{2cm}}?$
- $0110110^R = \underline{\hspace{2cm}}?$
- $[\text{article noun}]^R = \underline{\hspace{2cm}}?$

Definition: The **CONCATENATION** of strings x and y is the string obtained by appending y to the end of x .

- The concatenation of **penelope** and **fnord** is **penelopefnord**.

6. A SAMPLE FSA: THE COLA MACHINE

Imagine a cola machine with the following characteristics:

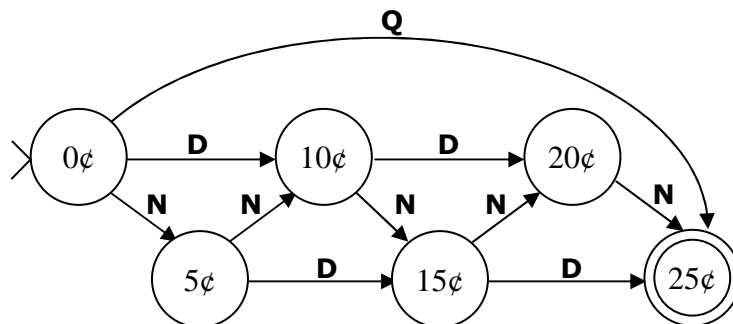
- Drinks cost 25 cents.
- The machine requires exact change—if you’ve put two dimes and try to put in a third, the machine will reject it.
- The machine takes only quarters, dimes, and nickels.
- The machine takes any combination of these coins, in any order, as long as the total is exactly 25 cents.

Exercise: List the sequences of coins the machine will accept.
(e.g., it will accept “dime, dime, nickel”, but not “dime, dime, dime”)

We can describe this machine with a finite state automaton; let’s call it *CM*. It will have the following characteristics:

- At any given point, the machine might contain 0¢, 5¢, 10¢, 15¢, 20¢, or 25¢. We’ll use these to label the states of the FSA.
- The “input signals” that tell the machine to go from state to state are the coins it accepts: quarters (**Q**), dimes (**D**), and nickels (**N**).
- Because the machine starts with no money in it, 0¢ is the start state.
- We’re done putting coins into the machine when it has 25 cents in it, so we’ll make that the end state.

We have:



7. FINITE STATE AUTOMATA: A FORMAL DEFINITION

Definition: A FINITE STATE AUTOMATON is a 5-tuple $\langle Q, \Sigma, \delta, s, F \rangle$, in which:

- Q is a finite set of states;
- Σ is the alphabet;
- δ is the transition function;
- $s \in Q$ is the start state;
- $F \subseteq Q$ is the set of accept states (also called “end states”).

We still need to explore the “transition function”—roughly, it’s the arrows, which are called **TRANSITIONS**, and their labels—but the other four are familiar enough. Take the cola machine:

- (2) $CM = \langle Q, \Sigma, \delta, s, F \rangle$, where:
- $Q = \{0\text{¢}, 5\text{¢}, 10\text{¢}, 15\text{¢}, 20\text{¢}, 25\text{¢}\}$
 - $\Sigma = \{\mathbf{Q}, \mathbf{D}, \mathbf{N}\}$
 - $\delta = \dots$
 - $s = 0\text{¢}$
 - $F = \{25\text{¢}\}$

Recall that a language is a (possibly infinite) set of (finite-length) strings. We can say:

Definition: A FSA **ACCEPTS** a string 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** a string if there is no such path.)

- The machine CM accepts the string **DDN**, because there is a path as follows:
 - Start at the start state, 0¢ .
 - The first symbol in the string is **D**. Follow the transition labeled **D** from the current state to the next state. We’re now at the state labeled 10¢ .
 - The next symbol in the string is **D**. Follow the transition labeled **D**. We’re now at the state labeled 20¢ .
 - The next symbol in the string is **N**. Follow the transition labeled **N**. We’re now at the state labeled 25¢ .
 - We’ve used all the symbols in the string, so check to see if we’re at an accept state. We are, so the string is accepted.
- The machine CM does not accept the string **DD**: starting at 0¢ and following the relevant transitions causes us to end at the state 20¢ , which is not an accept state.
- The machine CM does not accept the string **DDDN**: starting at 0¢ and following the relevant transitions takes us first to 10¢ , from there to 20¢ , at which point the next symbol is **D** but there is no transition with that label to follow.

So now, given a FSA, we have an alphabet and a certain set of strings over that alphabet. To complete the analogy to language, let us say:

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

Question: what language does CM model?

7.1. *Deterministic FSAs, Non-Deterministic FSAs, and Transition Functions*

We can divide FSAs into two types: *deterministic* and *non-deterministic*.

- In a deterministic FSA (DFA), all moves are uniquely determined. That is, for any state q , each symbol of the alphabet will have exactly one transition from q to another state q' (which may be the same as state q).

The Automatic Door from the last lecture, if it had had start and end states, would be a DFA: each state has one and only one transition for each symbol.

- In a non-deterministic FSA (NFA), not all moves are uniquely determined. That is, for some state q and some symbol x , either there is no transition labeled x from q , or there is more than one transition labeled x from q .

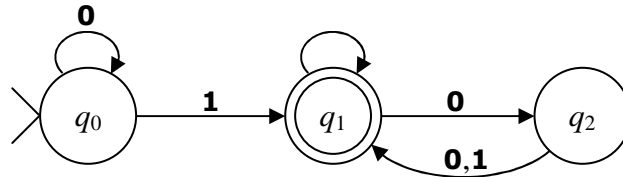
The machines $F1$ - $F3$ from the last lecture, which modeled small sets of English-like sentences, were NFAs: for instance, $F3$ had two transitions from *start* labeled with the symbol “*article*”, and no transitions labeled with the symbol “*noun*”.

Both DFAs and NFAs are 5-tuples $\langle Q, \Sigma, \delta, s, F \rangle$, as defined above, and they specify Q , Σ , s , and F in the same way. They differ in how they define δ . We'll start with DFAs and come back to NFAs later.

Definition: A **TRANSITION FUNCTION** δ is a function $Q \times \Sigma \rightarrow Q$. That is, it is a mapping from state/symbol pairs to states.

7.2. Reading and constructing state diagrams

Consider the following finite state automaton, M_1 .



By definition, $M_1 = \langle Q, \Sigma, \delta, s, F \rangle$, so it must have a set of states, an alphabet, a transition function, a start state, and a set of accept states.

- The states of M_1 are $\{q_0, q_1, q_2\}$.
- The alphabet of M_1 is the set of symbols labeling its transitions: $\{0, 1\}$.
- The start state $s = q_0$. (Unless otherwise specified, the start state of a FSA is by convention labeled q_0 .)
- The only accept state—i.e., the only one marked with a double circle—is q_1 . Thus, the set of accept states is $F = \{q_1\}$.

Note that this machine is deterministic: there is one and only one transition labeled **0** from each of the three states, and one and only one transition labeled **1** from each of the three states. So the transition function δ is the following function from state/symbol pairs to states:

$Q \times \Sigma$	\rightarrow	Q
$\langle q_0, 0 \rangle$	\rightarrow	q_0
$\langle q_0, 1 \rangle$	\rightarrow	q_1
$\langle q_1, 0 \rangle$	\rightarrow	q_2
$\langle q_1, 1 \rangle$	\rightarrow	q_1
$\langle q_2, 0 \rangle$	\rightarrow	q_1
$\langle q_2, 1 \rangle$	\rightarrow	q_1

i.e. $\delta(\langle q_0, 0 \rangle) = q_0$, etc. Less formally but more concisely, it can be represented:

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* in a straightforward manner, thereby either accepting or rejecting the string, as follows:

1. Start in the start state, and at the beginning of the string.
2. Read the next symbol in the string.
3. Follow that symbol's transition from the current state to the next state, and make that the new current state.
4. If there are symbols left, go to step 2.
5. Otherwise: if the current state is an accept state, accept the string, and if not, 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**.

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

- a. **1**
- b. **01**
- c. **101000**
- d. **11**
- e. **0101010101**
- f. **10**
- g. **0101000000**
- h. **100**
- i. **0**
- j. **0100**
- k. **110000**
- l. **111010100000**

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

1. $Q = \{q_0, q_1\}$
2. $\Sigma = \{0, 1\}$
3. δ is defined as

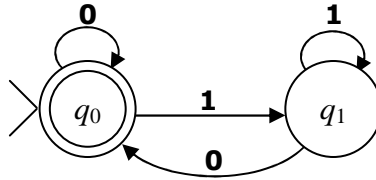
	0	1
q_0	q_0	q_1
q_1	q_0	q_1

4. $s = q_0$
5. $F = \{q_1\}$

Which of the following strings are accepted by M_2 ?

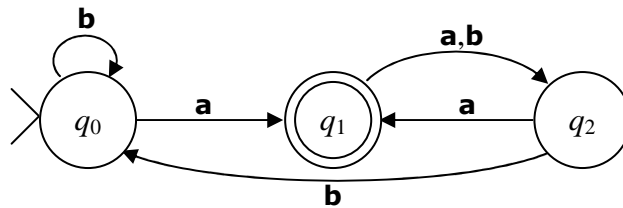
- a. **0**
- b. **1**
- c. **00**
- d. **11111**
- e. **1000000**
- f. **1010011**
- g. ϵ

Question: given the following state diagram for the FSA M_3 , give the formal description.



Which of the strings (a)-(g) above does M_3 accept?

Exercise: given the following state diagram for the FSA M_4 , give the formal description.



Which of the following strings does M_4 accept?

- a. **aabb**
- b. **abaab**
- c. **bbbba**
- d. **baaba**
- e. **aabaa**
- f. ϵ
- g. **ba**
- h. **a**
- i. **aaaaa**
- j. **abaa**

8. REGULAR LANGUAGES

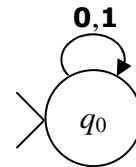
Recall from earlier:

Definition: A **REGULAR LANGUAGE** is a language that can be modeled with a FSA.

...actually, we've now got two different kinds of finite state automata: deterministic and non-deterministic. Let's use "regular language" to refer to a language that can be modeled by a *deterministic* FSA, and "ND-regular language" to refer to a language that requires a *non-deterministic* FSA.

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

Note that every FSA defines a language, i.e. a set of strings, even M_e : $M_e =$



What is $L(M_e)$? In fact, it's the empty language, \emptyset .

To show that a language is regular, construct a DFA for it.

Question: show that the following languages are regular for $\Sigma = \{0, 1\}$ by giving a DFA.

- $\{s \mid s \text{ contains at least three } 1s\}$
- $\{s \mid s \text{ begins with } 1 \text{ and ends with } 0\}$

Exercise: show that the following languages are regular for $\Sigma = \{0, 1\}$.

- $\{s \mid s \text{ contains at exactly three } 1s \text{ and any number of } 0s\}$
- $\{s \mid \text{the length of } s \text{ is odd}\}$
- $\{1\}$
- $\{s \mid 111 \text{ is a substring of } s\}$
- $\{s \mid \text{the length of } s \text{ is a multiple of } 3\}$

8.1. Regular Operations

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

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

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

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

Definition: The STAR of A is the language $A^* = \{x_1x_2\dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Example: suppose $P = \{\mathbf{10}, \mathbf{11}\}$ and $Q = \{\mathbf{10}, \mathbf{100}, \mathbf{1000}\}$, with $\Sigma = \{\mathbf{0}, \mathbf{1}\}$.

- $P \cup Q = \{\mathbf{10}, \mathbf{11}, \mathbf{100}, \mathbf{1000}\}$
- $P \cap Q = \{\mathbf{10}\}$
- $P \circ Q = \{\mathbf{1010}, \mathbf{10100}, \mathbf{101000}, \mathbf{1110}, \mathbf{11100}, \mathbf{111000}\}$
- $P^* = \{\epsilon, \mathbf{10}, \mathbf{11}, \mathbf{1010}, \mathbf{1011}, \mathbf{1110}, \mathbf{1111}, \mathbf{101010}, \mathbf{101011}, \mathbf{101110}, \mathbf{101111}, \mathbf{111010}, \mathbf{111011}, \mathbf{111110}, \mathbf{111111}, \mathbf{10101010}, \dots\}$

8.2. Properties of regular operations

Four theorems:

- The class of regular languages is closed under the union operation.
- The class of regular languages is closed under the intersection operation.
- The class of regular languages is closed under the concatenation operation.
- The class of regular languages is closed under the star operation.

where “is closed under” means that if the inputs to the operation are regular languages, the output is a regular language. So: if A and B are regular languages, then $A \cup B$ is a regular language, $A \cap B$ is a regular language, $A \circ B$ is a regular language, and A^* is a regular language.

8.2.1. Proof of closure under union

Given: A, B are regular languages.

To show: $A \cup B$ is a regular language.

In other words, by the definition of “regular language”: given that there is a FSA M_A that models A and a FSA M_B that models B , we want to show that there is a FSA M' that models $A \cup B$. How can we construct such a machine?

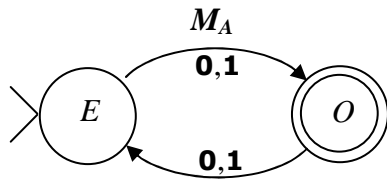
- The idea: the machine should check both machines “in parallel”. As it goes through the string, it should keep track of where M_A would be and where M_B would be—that is, it needs to “remember” a pair of states.

So: the states of M' will be the Cartesian product of the states of M_A and M_B .

The method to construct the FSA is as follows:

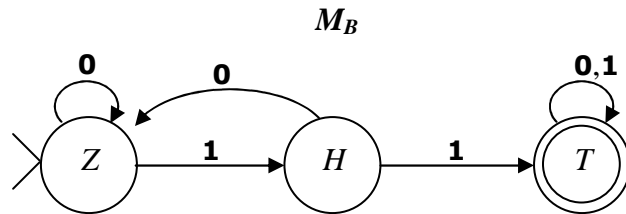
- **By definition:** A is modeled by some $M_A = \langle Q_A, \Sigma, \delta_A, s_A, F_A \rangle$
 B is modeled by some $M_B = \langle Q_B, \Sigma, \delta_B, s_B, F_B \rangle$
- Let $M' = \langle Q', \Sigma, \delta', s', F' \rangle$, where
 1. $Q' = Q_A \times Q_B = \{\langle q_i, q_j \rangle \mid q_i \in Q_A \text{ and } q_j \in Q_B\}$
 i.e., the states Q_x are pairs of states from machines A and B
 2. Σ' is the same alphabet as the alphabet of M_A and M_B .
 3. For each $\langle q_a, q_b \rangle \in Q'$ and $x \in \Sigma$: $\delta'(\langle q_a, q_b \rangle, x) = \langle \delta_A(q_a, x), \delta_B(q_b, x) \rangle$
 In other words: suppose we're at some state in Q' , 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 we would have gone to in M_A and the state we would have gone to in M_B .
 4. $s' = \langle s_A, s_B \rangle$
 5. $F' = \{\langle q_i, q_j \rangle \mid q_i \in F_A \text{ or } q_j \in F_B\}$
 i.e., the set of pairs in which either M_A or M_B would accept the string.
- M' is a finite state automaton: it has a finite number of states, a finite alphabet, etc. And M' models the language $A \cup B$: it accepts any string accepted by either A or B . Therefore, $A \cup B$ is a regular language.

Example: $A = \{s \mid \text{the length of } s \text{ is odd}\}$, $B = \{s \mid \mathbf{11} \text{ is a substring of } s\}$. We have:



1. $Q_A = \{E, O\}$
2. $\Sigma = \{0, 1\}$
3. $\delta_A =$

	0	1
<i>E</i>	<i>O</i>	<i>O</i>
<i>O</i>	<i>E</i>	<i>E</i>
4. $s_A = E$
5. $F_A = \{O\}$



1. $Q_B = \{Z, H, T\}$
2. $\Sigma = \{0, 1\}$
3. $\delta_B =$

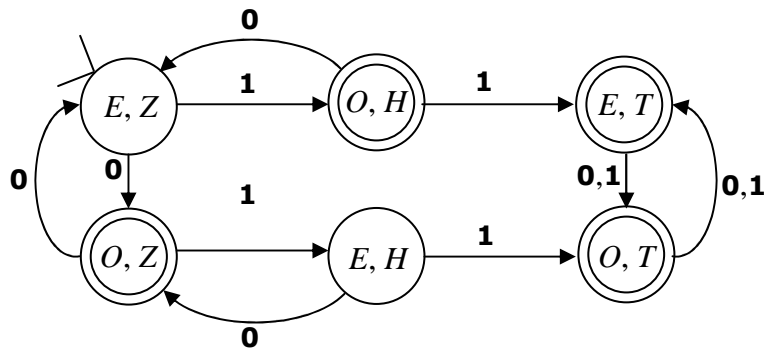
	0	1
<i>Z</i>	<i>Z</i>	<i>H</i>
<i>H</i>	<i>Z</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>T</i>
4. $s_B = Z$
5. $F_B = \{T\}$

Our new machine, M' :

- $Q' = \{ \langle E, Z \rangle, \langle E, H \rangle, \langle E, T \rangle, \langle O, Z \rangle, \langle O, H \rangle, \langle O, T \rangle \}$
- $\Sigma = \{ \mathbf{0}, \mathbf{1} \}$
- $\delta' =$

	0	1
$\langle E, Z \rangle$	$\langle O, Z \rangle$	$\langle O, H \rangle$
$\langle E, H \rangle$	$\langle O, Z \rangle$	$\langle O, T \rangle$
$\langle E, T \rangle$	$\langle O, T \rangle$	$\langle O, T \rangle$
$\langle O, Z \rangle$	$\langle E, Z \rangle$	$\langle E, H \rangle$
$\langle O, H \rangle$	$\langle E, Z \rangle$	$\langle E, T \rangle$
$\langle O, T \rangle$	$\langle E, T \rangle$	$\langle E, T \rangle$

- $s' = \langle E, Z \rangle$
- $F' = \{ \langle E, T \rangle, \langle O, Z \rangle, \langle O, H \rangle, \langle O, T \rangle \}$



$$M' = M_A \cup M_B$$

8.2.2. Proof of closure under intersection

Given: A, B are regular languages.

To show: $A \cap B$ is a regular language.

- **By definition:** A is modeled by some $M_A = \langle Q_A, \Sigma, \delta_A, s_A, F_A \rangle$
 B is modeled by some $M_B = \langle Q_B, \Sigma, \delta_B, s_B, F_B \rangle$
- Let $M' = \langle Q', \Sigma, \delta', s', F' \rangle$, where
 1. $Q' = Q_A \times Q_B = \{ \langle q_i, q_j \rangle \mid q_i \in Q_A \text{ and } q_j \in Q_B \}$
 2. Σ is the same alphabet as the alphabet of M_A and M_B .
 3. For each $\langle q_a, q_b \rangle \in Q'$ and $x \in \Sigma$: $\delta'(\langle q_a, q_b \rangle, x) = \langle \delta_A(q_a, x), \delta_B(q_b, x) \rangle$
 4. $s' = \langle s_A, s_B \rangle$
 5. $F' = \{ \langle q_i, q_j \rangle \mid q_i \in F_A \text{ and } q_j \in F_B \}$

Example: $A = \{s \mid \text{the length of } s \text{ is odd}\}$, $B = \{s \mid \mathbf{11} \text{ is a substring of } s\}$.

