Lecture 1

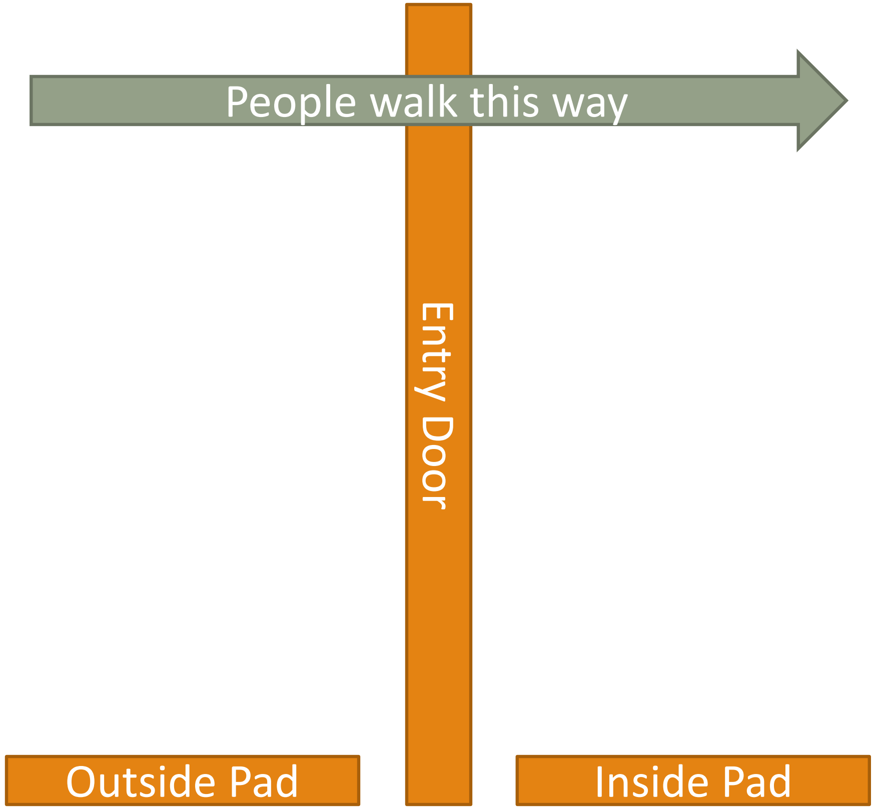
COT4210 DISCRETE STRUCTURES

DR. MATTHEW B. GERBER

5/19/2016

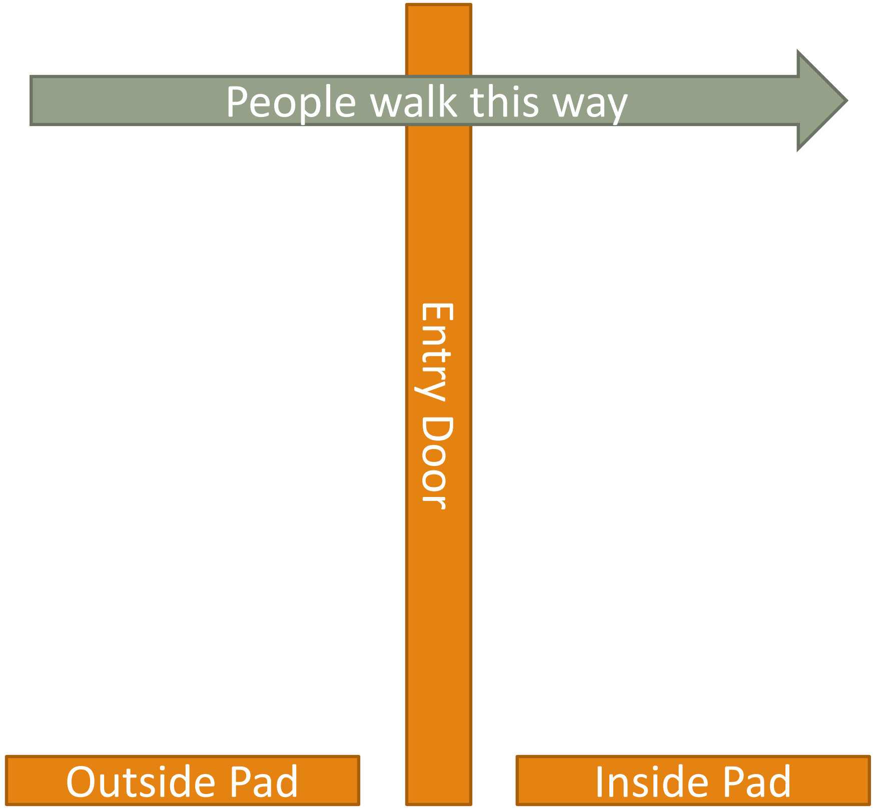
PORTIONS FROM SIPSER, *INTRODUCTION TO THE THEORY OF COMPUTATION*, 3RD ED., 2013

Consider an Automatic Entry Door



- When do we open and close the door?

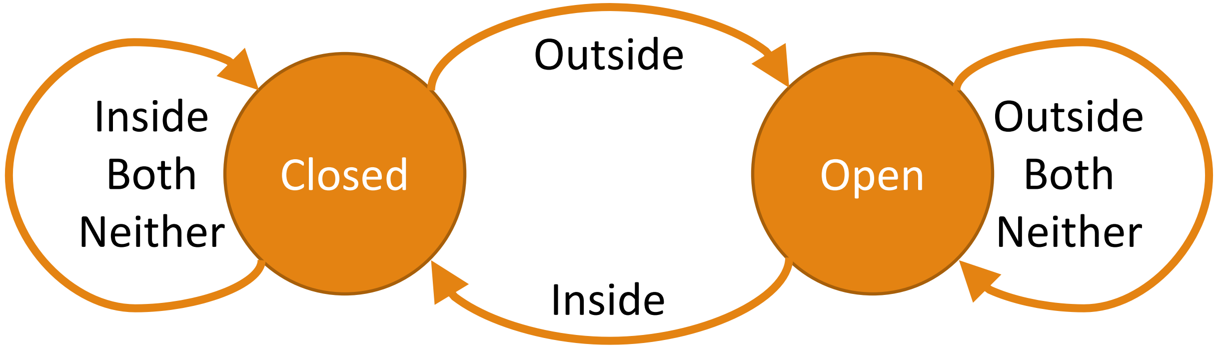
Consider an Automatic Entry Door



- When do we open and close the door?
- Four possible inputs: Outside, Inside, Both and Neither
- We *open* the door at Outside (only)
- We *close* the door at Neither
- Otherwise the door stays right where it is

States and Transitions

	Neither	Outside	Inside	Both
Closed	Closed	Open	Closed	Closed
Open	Closed	Open	Open	Open



We can think of the door in terms of:

- Its *states*: Open and Closed
- Its *transitions*: When it opens and closes

Two concise ways to depict these

- A *state transition table*
- A *state diagram*
- Either may be better for a given machine

We call logical constructs that we think of in these terms *automata*, or *machines*

- Let’s make this less fuzzy
- First, let’s remember strings

Review: Strings

- An *alphabet* is a non-empty, finite set of *symbols*
- A *string* over an alphabet is a finite sequence of symbols from that alphabet
- Strings have *length*, like any sequence; the empty string ε is the string with length 0
- A *language* is a set of strings over a given alphabet
- ***Do not confound the empty language with the empty string***

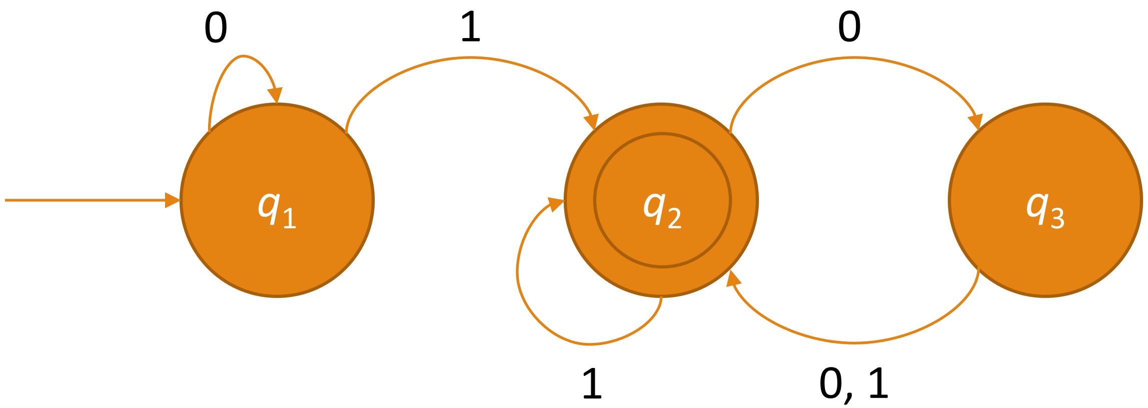
Given strings S, T, U and V, we write:

- S_i to denote the i^{th} symbol in S
- ST to denote the *concatenation* of S and T
- S^R to denote the *reverse* of S

…and we say:

- S is a *substring* of V if $\exists\ T, U \ni TSU = V$
  - …and a *proper substring* if $S \neq V$
- S is a *prefix* of V if $\exists\ T \ni ST = V$
  - …and a *proper prefix* if $S \neq V$
- S is a *suffix* of V if $\exists\ T \ni TS = V$
  - …and a *proper suffix* if $S \neq V$

Another Machine



This machine can read strings over the binary alphabet

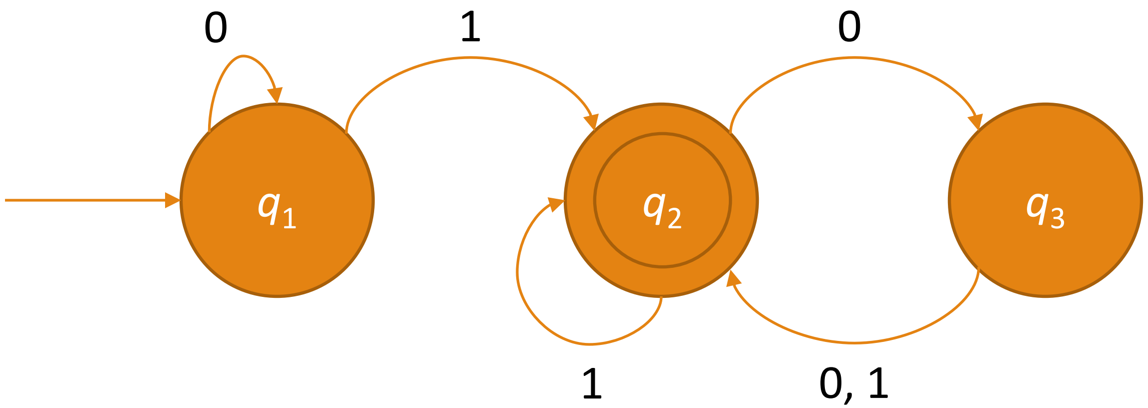
- The incoming arrow on the left means q_1 is the **starting** state
- The double circle around q_2 means it is an **accepting** state
- This kind of machine, given any string over its alphabet, either **accepts** or **rejects** it
- So what strings will this machine accept?

Definition:
Deterministic Finite Automata

A **deterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that consists of:

- Q — A finite set of *states*
- Σ — An *alphabet*
- $\delta: Q \times \Sigma \to Q$ — A *transition function*
- $q_0 \in Q$ — A *start state*
- $F \subseteq Q$ — A set of *accept* (or *final*) states

Example 1 (1.6)

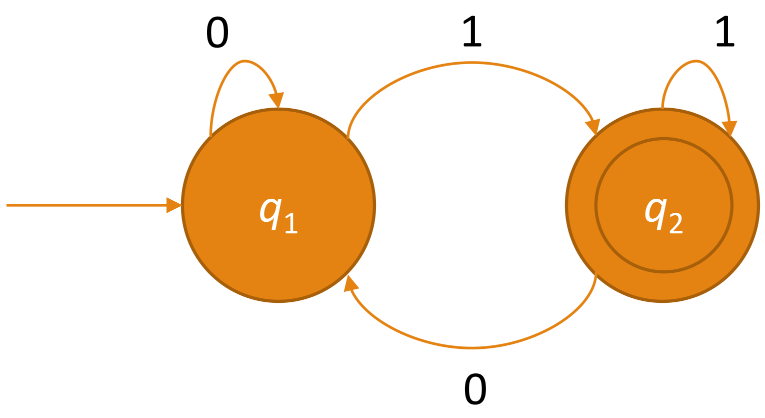


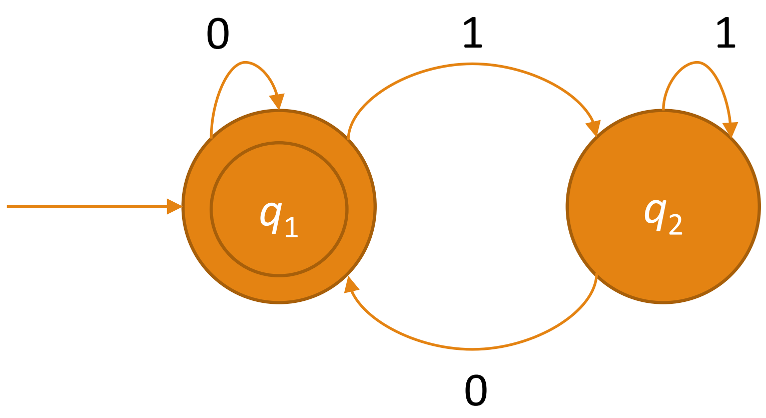
- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- δ:

	0	**1**
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

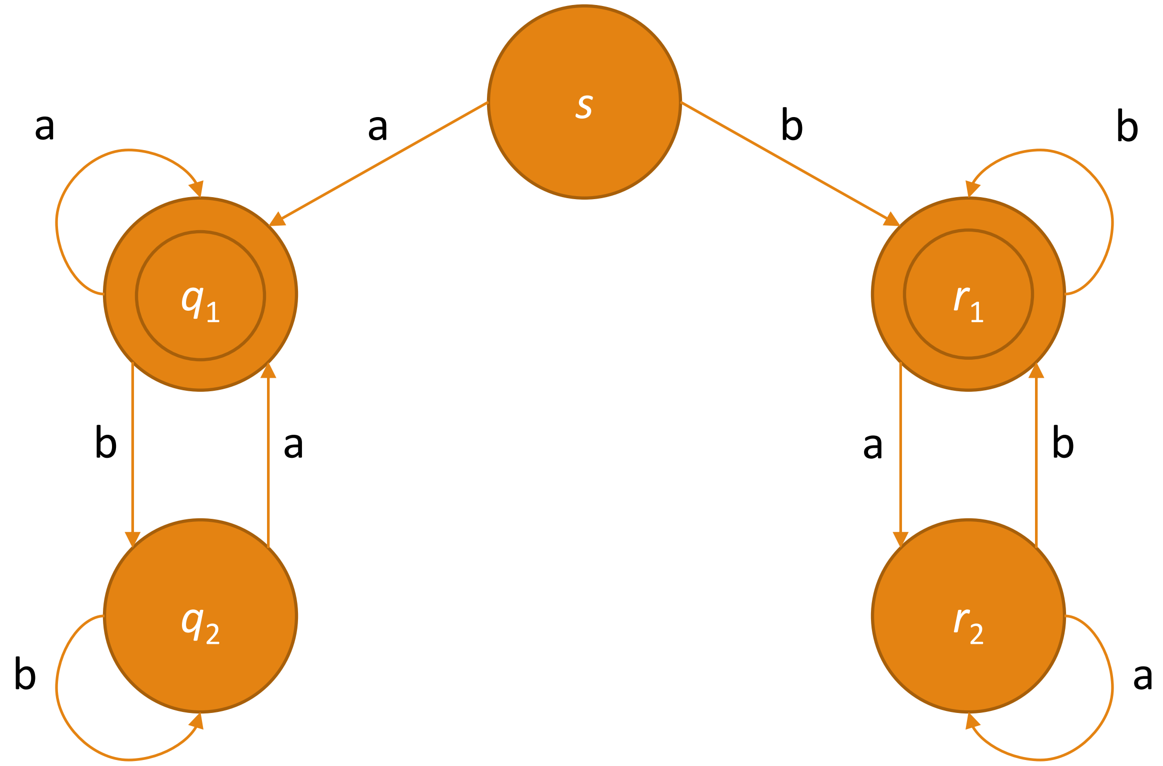
- q_1 *(start state)*
- $F = \{q_2\}$

Examples 2 and 3 (1.7, 1.9)





Example 4 (1.11)



Definition: Acceptance

Let $M = (Q, \Sigma, \delta, q_0, F)$ and $w = w_1w_2 \ldots w_n$ be a string of length n over Σ.

M **accepts** w if there exists a sequence of states in Q $r_0, r_1, \ldots, r_n$ so that:

1. $r_0 = q_0$
2. For i from 0 to $n - 1$, $\delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$

Definition: Acceptance (Computation)

Let $M = (Q, \Sigma, \delta, q_0, F)$ and $w = w_1w_2...w_n$ be a string of length n over Σ.

M **accepts** w if there exists a sequence of states in Q $r_0, r_1, ..., r_n$ so that:

1. $r_0 = q_0$
2. For i from 0 to $n - 1$, $\delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$

Definition: Recognition

A machine M **recognizes** language A if $A = \{w \mid M \text{ accepts } w\}$.

Definition: Regular Language

A language is **regular** if and only if it can be recognized by a DFA.

Designing Finite Automata (pp. 41-44)

(board work)

The Regular Operations

Let A and B be languages. We define **union**, **concatenation** and **star** (or **Kleene Closure**) as:

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

$$A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$$

Regular Languages: Union Closure

We want to prove that the class of regular languages is **closed** under the regular operations – that performing those operations on regular languages results in regular languages.

Let’s start with union – and for that, let’s go to the board…

Next Time: Nondeterminism