Theory of Computation
CSC 339 – Spring 2021

Chapter-1: part1
Regular Languages

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Outline

- **Midterm exam**
- **Recap**
- **Introduction**
- **Finite Automata (section 1.1 in the textbook)**

Midterm Exam

- **Date: Thursday 25/2/2021 (13/07/1442)**
- **Time: 5:00-6:30pm**
- **Midterm exam will make up 25% of the grade**
- **Topics included in the exam will be decided later.**

Recap

- **Set: a group of (unordered) objects represented as a unit**
- **Sequence: a list of objects in some order**
- **Function: an object to set up input-output relationship**
- **Graph: a set of nodes with lines connecting some of the nodes**
- **Alphabet: a non-empty <u>finite set</u>**
- **String: finite <u>sequence</u> of symbols from an alphabet**

Introduction

- **What is a computer?**

Introduction

- **What is a computer?**
  - **Real computers are complicated.. Hard to set up a <u>manageable</u> mathematical theory of them directly.**

Introduction

- **What is a computer?**
  - **Real computers are complicated.. Hard to set up a <u>manageable</u> mathematical theory of them directly.**
  - **So, we use an idealized (abstract) computer → computational model**

Introduction

- **What is a computer?**
  - **Real computers are complicated.. Hard to set up a <u>manageable</u> mathematical theory of them directly.**
  - **So, we use an idealized (abstract) computer → computational model**
- **We will start with the simplest model**
  - **Finite state machine (FSM) or Finite Automaton**

Finite Automata

- **What are Finite automata?**

Finite Automata

- What are Finite automata?
  - Good computational models for computers with extremely limited amount of memory.

Finite Automata

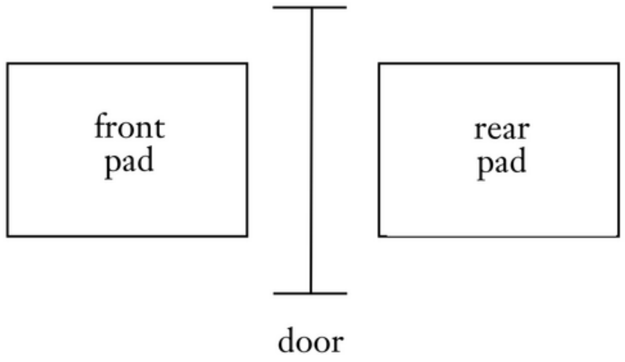
- **What are Finite automata?**
  - **Good computational models for computers with extremely limited amount of memory.**
  - **What can we do with such a small memory?**

Finite Automata

- **What are Finite automata?**
  - **Good computational models for computers with extremely limited amount of memory.**
  - **What can we do with such a small memory?**
  - **Computers with limited memory are everywhere**
    - **Embedded controllers..**
    - **IoT..**

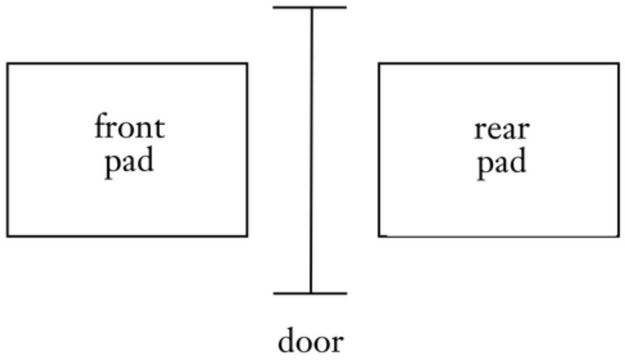
Example

- **Automatic door**



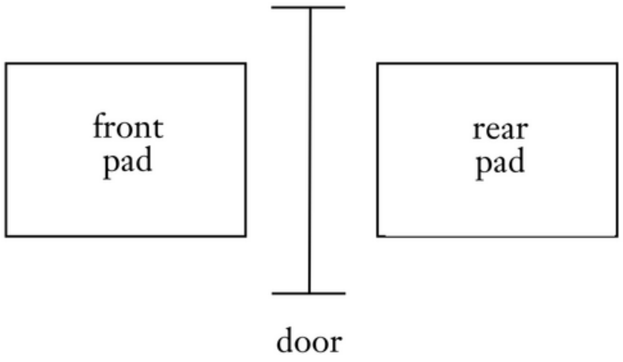
Example

- **Automatic door**
- **Two states:**
  - **OPEN**
  - **CLOSED**



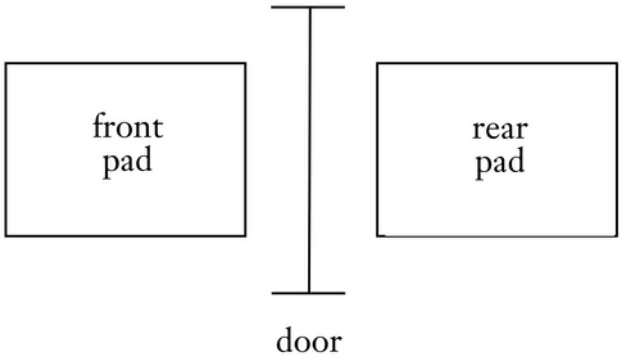
Example

- Automatic door
- Two states:
  - OPEN
  - CLOSED
- Input:
  - Front
  - Rear
  - Both
  - Neither

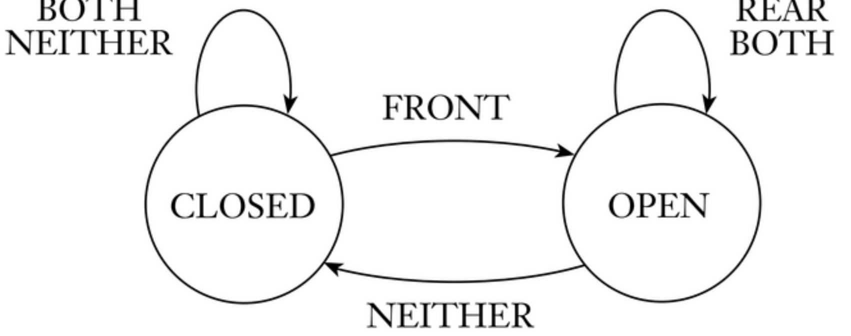


Example

- Automatic door
- Two states:
  - OPEN
  - CLOSED
- Input:
  - Front
  - Rear
  - Both
  - Neither



State Diagram



Example..

- **State transition table**

Input signal

State

	Neither	Front	Rear	Both
Closed				
Open				

Finite Automata

- **Types of finite automata**
  - **Deterministic finite automata (DFA)**
    - Given a word $\underline{w}$, the automaton will always end up in state $\underline{q}$

Finite Automata

- **Types of finite automata**
  - **Deterministic finite automata (DFA)**
    - Given a word *w*, the automaton will always end up in state *q*
  - **Non-deterministic finite automata (NFA)**
    - We cannot predict from *w* alone which state the automaton will end up in.
    - i.e., being in multiple states at once
    - We will look at ways to convert NFA to DFA

Finite Automata

- **One of the goals of designing finite automata is to recognize languages.**
  - **An alphabet specifies the symbols that a language may use.**
  - **A language provides the specifications and requirements for strings that should be considered as instances of this language.**
  - **A string is an instance representation of a given language such that it follows the rule of that language.**

Finite Automata

- **What makes a finite automaton?**

Finite Automata

- **What makes a finite automaton?**
  - **5-tuple (M)**
    - **Finite set of states (Q)**
    - **Alphabet (Σ)**
    - **Transition function (δ : Q x Σ → Q)**
    - **Start state ($q_0 \in Q$)**
    - **Accept states ($F \subseteq Q$)**

Finite Automata

- **What makes a finite automaton?**
  - **5-tuple (M)**
    - **Finite set of states (Q)**
    - **Alphabet (Σ)**
    - **Transition function (δ : Q x $\Sigma \rightarrow$ Q)**
    - **Start state ($q_0 \in Q$)**
    - **Accept states ($F \subseteq Q$)**

$$\boxed{M = (Q, \Sigma, \delta, q_0, F)}$$

Finite Automata

- **What makes a finite automaton?**

Can we have more than 1 accept state?

  - **5-tuple (M)**
    - **Finite set of states (Q)**
    - **Alphabet (Σ)**
    - **Transition function (δ : Q x Σ → Q)**
    - **Start state ($q_0 \in Q$)**
    - **Accept states ($F \subseteq Q$)**

$$\boxed{M = (Q, \Sigma, \delta, q_0, F)}$$

Finite Automata

- What makes a finite automaton?
  - 5-tuple (M)
    - Finite set of states (Q)
    - Alphabet (Σ)
    - Transition function (δ : Q x Σ → Q)
    - Start state ($q_0 \in Q$)
    - Accept states ($F \subseteq Q$)

Can we have more than 1 accept state?

Is there a limit on the number of states?

$$\boxed{M = (Q, \Sigma, \delta, q_0, F)}$$

Finite Automata

- **What makes a finite automaton?**
  - **5-tuple (M)**
    - **Finite set of states (Q)**
    - **Alphabet (Σ)**
    - **Transition function (δ : Q x $\Sigma \rightarrow$ Q)**
    - **Start state ($q_0 \in$ Q)**
    - **Accept states (F $\subseteq$ Q)**

Can we always describe an automaton using state diagrams?

$$\boxed{M = (Q, \Sigma, \delta, q_0, F)}$$

Finite Automata

- What makes a finite automaton?
  - 5-tuple (M)
    - Finite set of states (Q)
    - Alphabet (Σ)
    - Transition function ($\delta : Q \times \Sigma \rightarrow Q$)
    - Start state ($q_0 \in Q$)
    - Accept states ($F \subseteq Q$)

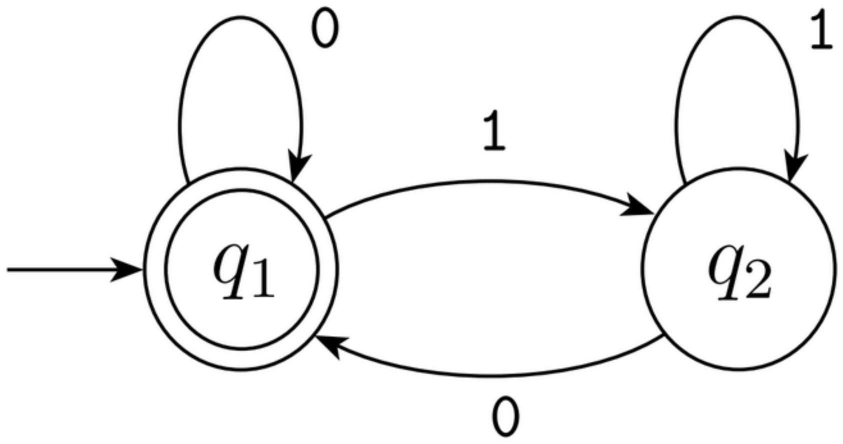
Can we always describe an automaton using state diagrams?

If number of states is too large, we resort to formal description

$$M = (Q, \Sigma, \delta, q_0, F)$$

Finite Automata - Example

- **What strings does this automaton accept?**
- **Q =**
- **Σ =**
- **δ =**
- **q_0 =**
- **F =**



Finite Automata

- How can we tell if a language is recognized by an automaton?
- Or how can we tell if a string would be accepted by a certain automaton?

Finite Automata

- **$w = a_1a_2a_3\ldots a_n$ is a string over the alphabet Σ. Automaton *M* <u>*accepts*</u> w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in *Q* such that:**

- **Three main conditions**

  1) $r_0 = q_0$

Finite Automata

- $w = a_1a_2a_3\ldots a_n$ is a string over the alphabet Σ. Automaton M ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in Q such that:
- **Three main conditions**

  1) $r_0 = q_0$

The machine (automaton) starts in the start state

Finite Automata

- $w = a_1a_2a_3\ldots a_n$ is a string over the alphabet Σ. Automaton *M* ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in *Q* such that:

- **Three main conditions**

  1) $r_0 = q_0$

  2) $r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$

Finite Automata

- **$w = a_1a_2a_3 \ldots a_n$ is a string over the alphabet Σ. Automaton *M* *<u>accepts</u>* w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in *Q* such that:**

- **Three main conditions**

  1) **$r_0 = q_0$**
  2) **$r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$**

The machine goes from state to state according to the transition function

Finite Automata

- $w = a_1a_2a_3 \ldots a_n$ is a string over the alphabet Σ. Automaton *M* ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in *Q* such that:
- Three main conditions
  1) $r_0 = q_0$
  2) $r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$
  3) $r_n \in F$

Finite Automata

- **$w = a_1a_2a_3\ldots a_n$ is a string over the alphabet Σ. Automaton *M* *accepts* w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in *Q* such that:**
- **Three main conditions**
  1) $r_0 = q_0$
  2) $r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$
  3) $r_n \in F$

The machine accepts its input if it ends up in an accept state

Finite Automata

- **$w = a_1a_2a_3 \ldots a_n$ is a string over the alphabet Σ. Automaton *M* <u>*accepts*</u> w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in *Q* such that:**

- **Three main conditions**
  1) **$r_0 = q_0$**
  2) **$r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$**
  3) **$r_n \in F$**

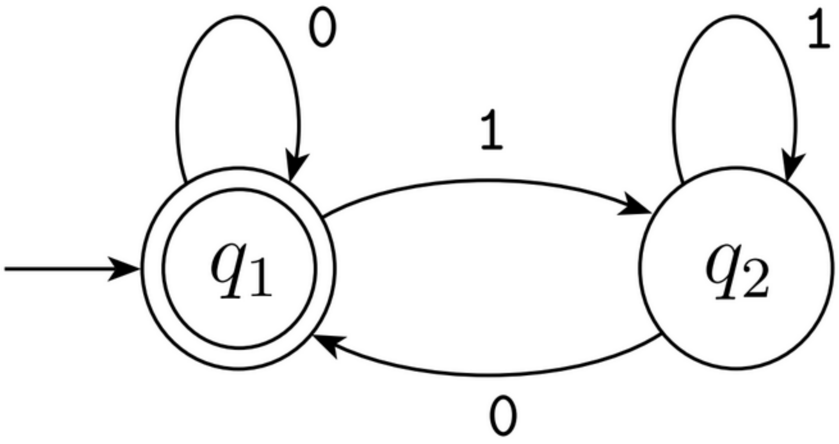
M recognizes language **A** if **A** = {**w**| **M** accepts **w**}

Finite Automata – Regular Languages

A language is called a **regular language** if some finite automaton **recognizes** it.

Finite Automata - Example

- **What is the language recognized by this automaton?**
  - **L(M) = {w|w is ε or ends in a 0}**



Finite Automata: Regular Operations

- **What tools can we use to manipulate finite automata?**

Finite Automata: Regular Operations

- **What tools can we use to manipulate finite automata?**
- **We will look at tools and techniques to help us recognize regular languages.**

Finite Automata: Regular Operations

- **What tools can we use to manipulate finite automata?**
- **We will look at tools and techniques to help us recognize regular languages.**
- **Regular operations**
  - **Union**
  - **Concatenation**
  - **Star**

Finite Automata: Regular Operations

- **Union**
- **Concatenation**
- **Star**

Finite Automata: Union Operation

- **$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$**
- **Binary operation (involving two sets)**

Finite Automata: Concatenation Operation

- $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- Binary operation
- Example:

  A = {0,1,2,3,4,5,6,7,8,9}

  B = {A,B,C,D,E,…,Z}

  x = CSC

  y = 339

  xy = CSC339

Finite Automata: Star Operation

- $A^* = \{x_1x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$
- Unary operation
- Attach any number of strings in A together to get a string in the new language.
- The empty string (ε) is always a member of A^*
- Also called kleene star

Finite Automata: Operations - Example

- **A = {fast, slow}**
- **B = {car, truck}**
  - **A ∪ B = {fast, slow, car, truck}**
  - **A ○ B = {fastcar, fasttruck, slowcar, slowtruck}**
  - **A* = { ε , fast, slow, fastfast, fastslow, slowslow, fastfastslow, fastfastfast, slowfastslow, …}**

Finite Automata: Regular Operations

A collection of objects is considered "***closed*** under some operation" if applying that operation to members of the collection also returns an object still in the collection

Finite Automata: Being closed under union operation

- **If A_1 and A_2 are regular languages, so is $A_1 \cup A_2$**
- **A finite automaton (M_1) recognizes A_1, M_2 recognizes A_2.**
- **To prove that $A_1 \cup A_2$ is regular, we use a finite automaton (M) that recognizes $A_1 \cup A_2$.**

Theorem 1.25

The class of regular languages is closed under the union operation

Finite Automata: Being closed under union operation

Proof by construction

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and

M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Finite Automata: Being closed under union operation

Proof by construction

Construct M to recognize A1 ∪ A2, where $M = (Q, \Sigma, \delta, q_0, F)$

1. $Q = \{(r_1, r_2) | \; r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
   – this is equivalent to $Q_1 \times Q_2$ (Cartesian product)
2. Σ, the alphabet, is the same as in M_1 and M_2
3. $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$ for each $(r_1, r_2) \in Q$ and each $a \in \Sigma$
4. q_0 is the pair (q_1, q_2)
5. $F = \{(r_1, r_2) | \; r_1 \in F_1$ **or** $r_2 \in F_2\}$

Finite Automata: Being closed under union operation

Proof by construction

Construct M to recognize A1 ∪ A2, where M = (Q, Σ, δ, q_0, F)

1. Q = {(r_1, r_2)| r_1 ∈ Q_1 and r_2 ∈ Q_2}

We can observe that every element of an ordered pair gives us a sequence of states from either machine M_1 or M_2

2. Σ, th

3. δ((r_1

4. q_0 is the pair (q_1, q_2)

5. F = {(r_1, r_2)| r_1 ∈ F_1 **or** r_2 ∈ F_2}

Homework

Exercise:

1.1, 1.2, and 1.6 (a-f)

Reading:

1.2