Welcome to CSE105  
and  
Happy and fruitful New Year

שנה טובה
(Happy New Year)

Staff

Instructor: Amos Israeli room 2-108.
Office Hours: Fri 10-12 or by arrangement
Tel#: 534-886
e-mail: aisraeli “at” cs.ucsd.edu

TA: William Matthews
Office Hours: William Matthews
e-mail: wgmatthews “at” cs.ucsd.edu

Web Page:
http://www.cse.ucsd.edu/classes/wi09/cse105

Meeting Times

Lectures: Tue, Thu 2:00p - 3:20p Cognitive Science Building 001
Sections: Mon, 1:00p - 1:50p Pepper Canyon Hall 120 – Will
          Tue, 2:00p - 2:50p WLH 2205 – Amos
Note: Students should come to both sections.
Mid-term: Thu. Feb. 12 14:00-15:20 CSB 01
Final: Thu Mar 19 15:00a - 18:00p TBA

Homework

There will be 6-7 assignments.
Assignments will be given on Monday’s section and must be returned by the Thu lecture one week later.
First assignment is already given.
Academic Integrity

• You are encouraged to discuss the assignments problems among yourselves.
• Each student must hand in her/his own solution.
• You must Adhere to all rules of Academic Integrity of CS Dept. UCSD and others. (Meaning: no cheating or copying from any source)

Grading

• Assignments: 20%
• Midterm: 30% - 0% (students are allowed to drop it), open book, open notes.
• Final: 50% - 80%, open book, open notes.

Grading

• A: (85-100)
• B: (70-84)
• C: (55-69)
• D: (40-54)
• F: (0-39)

Students who get a c- can ask me to bump their grade to D if they wish.

Bibliography

Introduction to Computability Theory

Lecture 1: Finite Automata and Regular Languages
Prof. Amos Israeli

Introduction

Computer Science stems from two starting points:

Mathematics: What can be computed? And what cannot be computed?
Electrical Engineering: How can we build computers?
Not in this course.

Computability Theory deals with the profound mathematical basis for Computer Science, yet it has some interesting practical ramifications that I will try to point out sometimes.
The question we will try to answer in this course is:
“What can be computed? What Cannot be computed and where is the line between the two?”

Computational Models

A Computational Model is a mathematical object (Defined on paper) that enables us to reason about computation and to study the properties and limitations of computing.
We will deal with Three principal computational models in increasing order of Computational Power.
Computational Models

We will deal with three principal models of computations:


Alan Turing - A Short Detour

Dr. Alan Turing is one of the founders of Computer Science (he was an English Mathematician).

Important facts:

1. “Invented” Turing machines.
2. “Invented” the Turing Test.
3. Broke into the German submarine transmission encoding machine “Enigma”.
4. Was arraigned for being gay and committed suicide soon after.

Finite Automata - A Short Example

• The control of a washing machine is a very simple example of a finite automaton.
• The most simple washing machine accepts quarters and operation does not start until at least 3 quarters were inserted.

Control of a Simple Washing Machine

• Accepts quarters.
• Operation starts after at least 3 quarters were inserted.
• Accepted words: 25,25,25; 25,25,25,25; ...
Finite Automata - A Short Example

- The second washing machine accepts 50 cents coins as well.
- Accepted words: 25,25,25; 25,50; ...

Finite Automaton – An Example

**States:** $Q = \{q_s, q_0, q_1\}$

**Initial State:** $q_s$

**Final State:** $q_0$

**Transition Function:**
- $\delta(q_s, 0) = q_0$
- $\delta(q_s, 1) = q_1$
- $\delta(q_0, 0) = \delta(q_0, 1) = q_0$
- $\delta(q_1, 0) = \delta(q_1, 1) = q_1$

**Alphabet:** $\{0,1\}$. **Note:** Each state has all transitions.

**Accepted words:** 0,00,01,000,001,...
Finite Automaton – Formal Definition

A **finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\) where:
1. \(Q\) is a finite set called the **states**.
2. \(\Sigma\) is a finite set called the **alphabet**.
3. \(\delta: Q \times \Sigma \to Q\) is the **transition function**.
4. \(q_0 \in Q\) is the **start state**, and
5. \(F \subseteq Q\) is the set of **accept states**.

Observations

1. Each state has a **single transition** for each symbol in the alphabet.
2. Every FA has a computation for every finite string over the alphabet.

Examples

1. \(M_2\) accepts all words (strings) ending with 1.
   The language **recognized** by \(M_2\), called \(L(M_2)\) satisfies: \(L(M_2) = \{w \mid w \text{ ends with } 1\}\).

How to do it

1. Find some simple examples (short accepted and rejected words).
2. Think what should each state “remember” (represent).
3. Draw the states with a proper name.
4. Draw transitions that preserve the states’ “memory”.
5. Validate or correct.
6. Write a correctness argument.
The automaton

Correctness Argument:
The FA’s states encode the last input bit and $q_1$ is the only accepting state. The transition function preserves the states encoding.

Examples

1. $M_2$ accepts all words (strings) ending with 1.
   \[ L(M_2) = \{ w \mid w \text{ ends with 1} \} \]
2. $M_3$ accepts all words ending with 0.
3. $M_4$ accepts all strings over alphabet $\{a, b\}$ that start and end with the same symbol.
4. $M_5$ accepts all words of the form $0^m1^n$ where $m, n$ are integers and $m, n > 0$.
5. $M_6$ accepts all words in $\{0,1,00,01,10\}$.
**Languages**

- Definition: A **language** is a set of strings over some alphabet.
- Examples:
  - $L_1 = \{0,1,10,111001\}$
  - $L_2 = \{0^n1^m \mid n, m \text{ are positive integers}\}$
  - $L_3 = \{\text{bit strings whose binary value is a multiple of 4}\}$

**Some Questions**

Q1: How do you prove that a language $La$ is regular?
A1: By presenting an FA, $M$, satisfying $La = L(M)$.

Q2: Why is it important?
A2: Recognition of a regular language requires a controller with bounded Memory.

Q3: How do you prove that a language $La$ is not regular?
A3: Hard! to be answered on Week3 of the course.

**Languages**

- A language of an FA, $M$, $L(M)$, is the set of words (strings) that $M$ accepts.
- If $La = L(M)$ we say that $M$ recognizes $La$.
- If $La$ is recognized by some finite automaton $A$, $La$ is a **Regular Language**.

**Wrap up**

In this talk we:
1. Motivated the course.
2. Defined **Finite Automata** (Latin Pl. form of automaton).
3. Learned how to deal with construction of automata and how to come up with a correctness argument.