Welcome to CS105 and Happy and fruitful New Year

שנה טובה
(Happy New Year)

Staff

Instructor: Amos Israeli room 2-106.
Office Hours: Fri 10-12 or by arrangement
Tel#: 534-886
e-mail: aisraeli “at” cs.ucsd.edu
TA: Scott Yilek room B-240a
Office Hours: Mon 4-5:30
e-mail: syilek “at” cs.ucsd.edu

• Web Page:
  http://www.cse.ucsd.edu/classes/fa08/cse105

Meeting Times

Lectures: Tue, Thu 3:30p - 4:50p WLH 2111
Note: Thu Oct 9 Lecture is canceled, there may be a make-up lecture by quarter’s end.
Sections: Mon, 1:00p - 1:50p WLH 2209 – Scott
         Tue, 3:00p - 3:50p WLH 2209 – Amos
Note: Students should come to both sections.
Mid-term: Thu Oct 23 3:30p WLH 2111
Final: Tue Dec 9 11:30a - 2:20p TBA

Homework

There will be 5-6 assignments.
Assignments will be given on Monday’s section and must be returned by the Thu lecture one week later.
First assignment will be given next Monday and must be returned on Wed Oct 8 or in Scott’s mailbox on Thu Oct 9th.
**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)

**Bibliography**

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.*

**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:

1. Finite Automaton (in short FA).
   recognizes Regular Languages.

2. Stack Automaton.
   recognizes Context Free Languages.

3. Turing Machines (in short TM).
   recognizes Computable Languages.

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 the German submarine transmission coding 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.

Washing Machine

• Accepts quarters
• Put in three (or more) quarters, should start washing
• Put in less than three, does nothing

Thanks: Vadim Lyubasehvsky
Now on PostDoc in Tel-Aviv
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.
• The second washing machine accepts 50 cents coins as well.

• The most complex washing machine accepts $1 coins too.

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.
• The second washing machine accepts 50 cents coins as well.
• The most complex washing machine accepts $1 coins too.
Finite Automaton - An Example

States: \( Q = \{ q_s, q_0, q_1 \} \)

Initial State: \( q_s \)

Final State: \( q_0 \)

Transition Function:
\[
\delta(q_s, 0) = q_0 \quad \delta(q_s, 1) = q_1 \\
\delta(q_0, 0) = \delta(q_0, 1) = q_0 \quad \delta(q_1, 0) = \delta(q_1, 1) = q_1
\]

Alphabet: \( \{ 0, 1 \} \)

Accepted words: 0, 00, 01, 000, 001, ...

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. \( \delta: Q \times \Sigma \rightarrow Q \) 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.
2'. $M_4$ accepts all words ending with 0 and the empty word $\varepsilon$.
   This is the Complement Automaton of $M_2$. 

25

28
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.

Examples (cont)

4. $M_5$ accepts all words of the form $0^n1^n$ 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:
  - $L_1 = \{0, 1, 10, 1110001\}$
  - $L_2 = \{0^n1^n \mid n, m \text{ are positive integers}\}$
  - $L_3 = \{\text{bit strings whose binary value is a multiple of } 4\}$

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.
Some Questions

Q1: How do you prove that a language $L_A$ is regular?
A1: By presenting an FA, $M$, satisfying $L_a = L(M)$.
Q2: How do you prove that a language $L_a$ is not regular?
A2: Hard! to be answered on Week3 of the course.
Q3: Why is it important?
A3: Recognition of a regular language requires a controller with bounded Memory.

The Regular Operations

Let $A$ and $B$ be 2 regular languages above the same alphabet, $\Sigma$. We define the 3 Regular Operations:

- **Union**: $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$.
- **Concatenation**: $A \circ B = \{ xy \mid x \in A \text{ and } y \in B \}$.
- **Star**: $A^* = \{ x_1, x_2, \ldots, x_k \mid k \geq 0 \text{ and } x_k \in A \}$.

The Regular Operations - Examples

\[
A = \{ \text{good, bad} \} \quad B = \{ \text{girl, boy} \}
\]

- $A \cup B = \{ \text{good, bad, girl, boy} \}$
- $A \circ B = \{ \text{goodgirl, goodboy, badgirl, badboy} \}$
- $A^* = \{ \varepsilon, \text{good, bad, goodgood, goodbad, goodgoodgoodbad, badbadgoodbad, ...} \}$
**Theorem**

The class of Regular languages, above the same alphabet, is **closed** under the **union** operation.
Meaning: The union of two regular languages is **Regular**.

**Example**

Consider $L_1 = \{w \mid w \text{ starts with 1}\}$, and $L_2 = \{w \mid w \text{ ends with 0}\}$.
The union set is the set of all bit strings that either start with 1, or end with 0.
Each of these sets can be recognized by an FA with 3 states.
Can you construct an FA that recognizes the union set?

**Proof idea**

If $A_1$ and $A_2$ are regular, each has its own recognizing automaton $N_1$ and $N_2$, respectively.
In order to prove that the language $A_1 \cup A_2$ is regular we have to construct an FA that accepts exactly the words in $A_1 \cup A_2$.
**Note:** cannot apply $N_1$ followed by $N_2$.

**Proof idea**

- We construct an FA that **simulates** the computations of, $N_1$ and $N_2$ simultaneously.
- Each state of the simulating FA represents a pair of states of $N_1$ and of $N_2$ respectively.
- Can you define the transition function and the final state(s) of the simulating FA?
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.