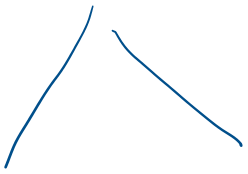
Introduction

10 January 2021 23:22

This is the Theory of Computation Course.



CS2410 CS2420 for IITH.

This is one of the most fundamental courses that you will see as part of the CSE curriculum.

We will try to find out

* What is computation?
* What are the questions for which computers can provide an answer?

What are the fundamental capabilities and limitations of computers?

Though computers have evolved significantly over the years, the theory of computation has remained applicable throughout.

Modern digital computers were developed

Modern digital computers were developed around and after the second world war. The theoretical basis was provided by Alan Turing in 1936.

"On Computable Numbers, with an Application to the Entscheidungs problem".

Algorithms, Complexity Theory } How fast / efficiently can we solve a problem?

Computability Theory } Is something "computable" at all?

Automata Theory } Some computation models with some very limited resources.

* Very little math as you know it. But mostly discrete maths. You [illegible] with theorems

But mostly ... should get comfortable with theorems and proofs.

Today is lecture 1. We will move to pre-recorded lectures and "Q&A meetings". All recordings and notes will be provided. We may not meet for all slots each week.

Evaluation – Weekly Quizzes
2 or 3 exams (CS 2401 & CS 2420 together)

Exact distribution to be announced next class.

Text: Michael Sipser: Introduction to Theory of Computation

Communication through Google Classroom. All announcements to be posted there.

Exercise before next lecture: Read Chapter 0

Exercise before next lecture. Read chapter 0 of the book. Sets, Functions, Relations, Graphs, Logic, Theorem, Proofs etc. Try out some problems.

Every Computational Problem is represented as a string input to a computer.

String will be over alphabet, say Σ.

Examples of Alphabet: $\Sigma_1 = \{0,1\}$ Binary

$\Sigma_2 = \{0,1,2,3,\ldots 9\}$ Decimal

$\Sigma_3 = \{a,b,c,\ldots y,z\}$. English.

$\Sigma_4 = \{a,b,0,1,2\}$.

A string over an alphabet Σ, is a finite sequence of symbols written one after another.

Example: 01101 is a string in binary

aabbaa is a string in English

string is a string in English

839 is a string in decimal

We say w is a string over Σ.

$|w|$ denotes the length of w.

$w = 110110 \Rightarrow |w| = 6$

$x = 825 \Rightarrow |x| = 3$

$y = \text{hello} \Rightarrow |y| = 5$

$\varepsilon \rightarrow$ empty string $\Rightarrow |\varepsilon| = 0$

Q: Is "hello there!" a string over $\{a, b, c, d, \ldots x, y, z\}$? $\cup \{\_, !\}$

w^R = reverse of w.

Concatenation: x is a string of length m

y is a string of length n.

$x = \text{"top"}$ $y = \text{"hat"}$.

$xy = \text{"tophat"}$

xy = "tophat"

$x^R y$ = "pothat"

$x^k = \underbrace{x x \ldots x}_{k \text{ times}}$

Kleene Star: $x^* =$ Set of all $x^k = \{x^k \mid k \geq 0\}$.

Σ^* = Set of all strings over Σ.

Substring: v is a substring of w if there exists strings x and y such that $w = xvy$.

A language over Σ is a set of strings over Σ. That is, a language over Σ is a subset $A \subseteq \Sigma^*$.

Example: Set of all binary strings with odd number of 1's is a language over $\{0,1\}$.

Languages can be specified in several ways.

1. Brute Force listing. $\{a, ab, abb, abbb, \ldots\}$
2. Language operations. ab^*.
3. Other set theoretic descriptions.

Finite Automata

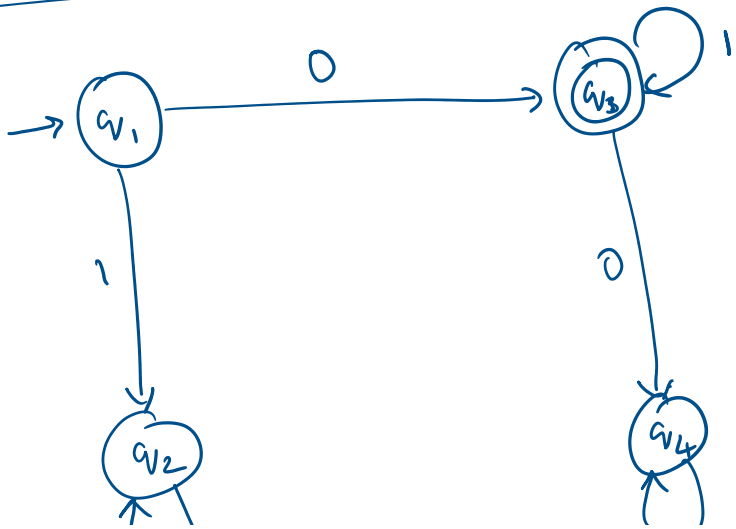
Computers with limited amount of memory.

State based devices.

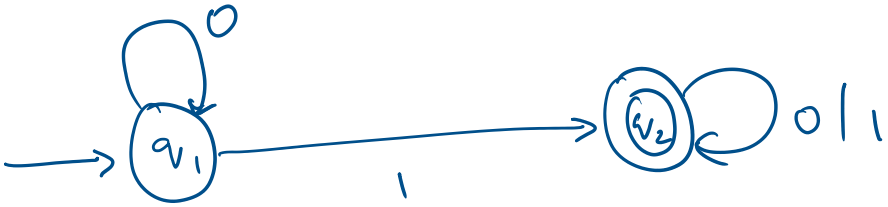
Examples: Timers, door open/close controller, Thermostat.

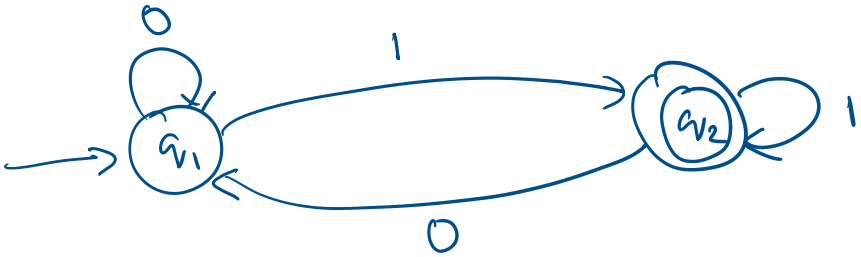
Why? These are abstract models that help us gain understanding.

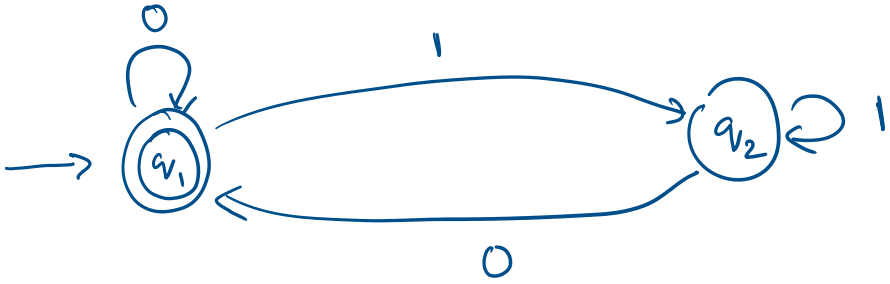
Deterministic Finite Automata



$A = \{0, 01, 011, 0111, 01111, \ldots\}$







DFA.

Def 1.5: A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where.

- Q is a finite set of states

where:

1. Q is a finite set of states
2. Σ is a finite alphabet.
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accepting states.

Read example 1.11, 1.13, 1.17, 1.21.

Read definition 1.16.