

CS3063 Theory of Computing

Semester 4 (20 Intake), Feb – Jun 2023

Lecture 1

- 1. Course Details**
- 2. Introduction to Theory of Computing**

Sanath Jayasena

Announcements

- This Lecture 1 is online on Zoom
- The rest in **flipped-classroom model**
 - Video recorded lecture and slides on Moodle; students study these privately in their own time
 - Interactive session physically every week from Week 3 (13th March), on Mondays, by dividing the students into 2 Groups
 - Start with a 5-6 minute online Quiz on the previous week's lecture
 - Group 1: Monday 10.15am – 11.00am in L2 Lab
 - Group 2: Monday 11.15am – 12.00 noon in L2 Lab

Announcements

- Lecture 1: online on Zoom
 - Today's password for Attendance marking on Moodle: **ianidw**

Today's Outline

Lecture 1

- **Course Details**
 - Objectives
 - Syllabus and Calendar
 - Evaluation
- **Introduction to Theory of Computing**
 - Overview
 - Mathematical Background

PART 1

Today's Outline

Lecture 1

- **Course Details**
 - Objectives
 - Syllabus and Calendar
 - Evaluation
- Introduction to Theory of Computing
 - Overview
 - Mathematical Background

Course Details

- On Moodle:
<https://online.uom.lk/course/view.php?id=20231>
- Videos and slides made available on Moodle
- Interactive sessions, starting with a Quiz, on Mondays
- [Details](#)

PART 2

Today's Outline

Lecture 1

- Course Details
 - Objectives
 - Syllabus and Calendar
 - Evaluation
- Introduction to Theory of Computing
 - Overview
 - Mathematical Background

Subjects of Interest...

- What are the fundamental **capabilities** and **limitations** of a computer?
- Above can be discussed in the contexts of
 - Complexity theory
 - Computability theory
 - Automata theory

What is *Computation*?

- One possible answer
 - Consists of executing an algorithm
 - Inputs + step-by-step procedure → result
- One might say: a *step* in computation is an operation a computer can perform
- What kind of *computers* will we consider?

What are *Computers*?

- The computers we consider are not real ones
 - Will a theory based on an actual piece of current hardware be useful?
 - Real computers are too complex for theoretical study
- We consider (simpler) *abstract machines* or *models of computation*
 - Defined mathematically (several of them)
 - Can be as powerful as real computers

Decision Problems

- We mainly consider *decision problems*
 - Computational problems for which the answer is either “yes” or “no”
- E.g., given an integer $n > 0$, *is n prime?*
 - n is encoded as a string of digits
 - This string will be the input to the problem

Languages

- Input to “is n prime?” problem is a string
- This is a **language recognition** problem
 - [What is a **language** ? → a set of strings]
 - For an arbitrary string of digits, determine whether it is one of the strings in the language of all strings that represent primes
- A decision problem can be stated as a problem of recognizing a language

Models of Computation & Languages

- Different types of abstract models can recognize languages of different complexity → *hierarchy of language types*
- Types of abstract machines
 - Finite automata
 - Pushdown automata
 - Linear-bounded automata
 - Turing machines

The *Chomsky* Hierarchy

Type	Abstract Machine	Languages (Grammars)
3	Finite automaton	Regular
2	Pushdown automaton	Context-free
1	Linear-bounded automaton	Context-sensitive
0	Turing machine	Recursively enumerable (unrestricted, phrase-structure)

Finite Automata

- A **finite automaton** (FA) is a finite-state machine
 - At each moment, in one of finite number of states
 - Moves among states in a predictable way responding to inputs
 - Recognizes a **regular language**

Pushdown Automata

- The limitation of an FA: **little or no memory**
 - It can only keep track of its current state
 - So, can recognize only simple languages
- **Context-free languages** can be recognized by **pushdown automata**
 - An FA with an auxiliary stack memory
 - **Context-free grammars**: important because they can describe syntax of programming languages

Turing Machines

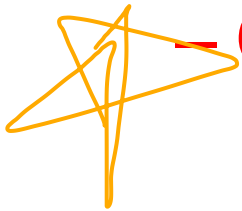
- A pushdown automaton cannot be a general model of computation
- Turing machine: general computing device
 - Can do any step-by-step procedure
 - More general languages than other two
- A Turing machine can do anything a computer can do

Is Every Problem Solvable?

- Turing machines can still have limits
 - But there is no more powerful machine
- There are **unsolvable problems**, no matter how much time and resources we have
 - **Computability theory** addresses this
 - Discussed using Turing machine as the computational model

Kinds of Solvable Problems

- There are ***intractable problems***
 - Problems that are solvable theoretically
 - But practical issues due to huge time/resource requirements
 - **Complexity theory** addresses this
- E.g., why are some problems computationally easy and others hard?



Computing Vs The Theory of ...

- Computation by humans has a long history
- But the current state of pervasive computing is a new phenomenon
- Theory of computing older than computers
 - Pioneers (Turing *et al*) saw the power of computers
 - Conceptual models very useful
 - Important field of study relevant to other areas

PART 3

Today's Outline

Lecture 1

- Course Details
 - Objectives
 - Syllabus and Calendar
 - Evaluation
- Introduction to Theory of Computing
 - Overview
 - Mathematical Background

Assumed Background: Basics

- Sets: basic set theory, notations
- Logic: propositional logic, implication, equivalence, quantifiers \forall and \exists
- Functions
- Relations
- Mathematical induction and proofs
- Recursion and recursive definitions
- **READ Chapters 1, 2**

Languages (again)

- A **language**
 - A **set of strings** where the symbols are drawn from an alphabet
- An **alphabet**
 - A finite **set of symbols**, denoted by Σ
- **Length** of a string x over Σ
 - Number of symbols in x , denoted by $|x|$

sigma = set of symbols

string = combination of symbols
from an alphabet (sigma)

language = set of strings

Languages ...contd

- **Null string** (string of length 0) is a string over Σ
 - Denoted by Λ
 - No matter what Σ is
- For any alphabet Σ , the set of all strings over Σ is denoted by Σ^* All of the combinations are here (strings over sigma)
 - A language over Σ is a subset of Σ^*

Null is included

$$\Sigma \subseteq \Sigma^*$$

Languages ...contd

- Example: if $\Sigma = \{a, b\}$, then:
 - Some strings over Σ are $a, baa, aba, aabba$
 - $|a|=1, |baa| = |aba| = 3, |aabaa| = 5$
 - $\Sigma^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, aba, \dots\}$
 - A few languages over Σ are:
 - $\{\Lambda, a, aa, aab\}$
 - $\{x \in \{a, b\}^* \mid |x| \text{ is odd}\}$
 - $\{x \in \{a, b\}^* \mid |x| \leq 8\}$

Languages ...contd

- New languages can be constructed using **set operations** (because languages are sets of strings)
- For any two languages over an alphabet Σ :
 - Their union, intersection, difference are also languages over Σ
- **Complement** L' of a language L over Σ
 - $L' = \Sigma^* - L$
- Language over Σ is a subset of Σ^*

Languages ...contd

- If $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ then both L_1 and L_2 are subsets of $(\Sigma_1 \cup \Sigma_2)^*$
- New languages can also be created using **concatenation**
 - If x and y are elements of Σ^* , concatenation of x and y is the string xy
 - For any string x , $x\Lambda = \Lambda x = x$
 - For any strings x , y and z , $(xy)z = x(yz)$

Languages ...contd

- A string x is a **substring** of a string y if there are strings w and z (either or both can be null) so that $y = wxz$
- A **prefix** of a string is an initial substring
 - E.g., Λ , a , ab , abb and $abba \rightarrow$ prefixes of $abba$
- A **suffix** is a final substring
 - E.g., Λ , a , ba , bba and $abba \rightarrow$ suffixes of $abba$

Languages ...contd

- If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ then
 $L_1 L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$
- For any language L , $L\{\Lambda\} = \{\Lambda\}L = L$
- We use exponential notation to indicate the number of items concatenated
 - Items can be symbols, strings or languages

Languages ...contd

- If Σ is an alphabet, $a \in \Sigma$, $x \in \Sigma^*$, $L \subseteq \Sigma^*$

$$\underline{a^k} = aa \cdot \cdot \cdot a$$

$$x^k = xx \cdot \cdot \cdot x$$

$$\Sigma^k = \Sigma\Sigma \cdot \cdot \cdot \Sigma = \{x \in \Sigma^* \mid |x|=k\}$$

$$L^k = LL \cdot \cdot \cdot L$$

- Special cases:

$$a^0 = \Lambda$$

$$x^0 = \Lambda$$

$$\Sigma^0 = \{\Lambda\}$$

$$L^0 = \{\Lambda\}$$

Languages ...contd

- Unit of concatenation is Λ
- Set of all strings obtained by concatenating any number of elements of L :

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Kleene star

Including 0 number of elements



- The set of all strings obtained by concatenating one or more elements of L :

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

Languages ...contd

- Note that $L^+ = L^*L = LL^*$
- The way we describe how to **construct** an arbitrary string in a language may also be used to **recognize** a string in the language
- **Recognition of languages** is considered in the context of abstract machines

Conclusion

- In this session we discussed ...
 - Course Details
 - Introduction to the Theory of Computing
 - Overview
 - Mathematical background
- Next time...
 - Regular Languages, Regular Expressions, Finite Automata