E0 224: Computational Complexity Theory

Lecturer: Chandan Saha
Scribe: Mohd Aqil

Lecture #3
13-Aug-2014

1 The Halting Problem

The halting problem takes as input strings α and x and decides if the turing machine M_α represented by α halts on input x within a finite number of steps.In other words given a computer program and an input , can we determine whether the program is going to enter an infinite loop on the input. More formally we define the language L_{HALT} as follows:

$L_{HALT} := \{(\alpha, w) : M_\alpha$ halts on input w $\}$

Claim: L_{HALT} is undecidable

Proof:Suppose for the sake of contradiction there exists a TM M that decides L_{HALT}. We will then show that this implies there is TM M' that decides L_{UC} thus contradicting our earlier theorem where $L_{UC} := \{\alpha \in \{0,1\}^*; M_\alpha(\alpha) = 0$ or M_α does not halt$\}$

M' first calls M as a subroutine with input (α, α).If M ouputs 0 this means that TM represented by α does not halt when given its own string as input so M' outputs 1. If M outputs 1 this means that the TM represented by α halts when given its own string as input.Then M' calls universal TM as a subroutine to simulate M_α on input α . If the universal TM outputs 0 then M' outputs 1 otherwise M' ouputs 0.

2 Complexity Classes

A Complexity Class is a set of functions computable by turing machines under some resource constraints such as Time, Space , Randomness(measured as the amount of random bits needed for computation) or Communication.

To define complexity classes we will be mainly interested in the computation of boolean functions i.e. functions that produce a single bit as their output.Such functions can also be seen as representing decision problems or languages.Let $f : \{0,1\}^* \rightarrow \{0,1\}$ denote the boolean function,then the language represented by this function is $L_f = \{x \in \{0,1\}^* : f(x) = 1\}$

2.1 Class DTIME

Let $T : \mathbb{N} \rightarrow \mathbb{N}$ be some function.We say that a language L is in DTIME(T(n)) if there is a deterministic turing machine M such that for every $x \in \{0,1\}^*$ M(x) halts in time $c.T(\|x\|)$ (where $c > 0$ is some constant depending on M) and M(x) = 1 iff L(x) = 1. Thus class DTIME(T(n)) represents the class of functions that are computable in time c.T(n) where n is the input length.

2.2 Class P

The class P is defined as follows:

$$P := \cup_{c \geq 1} DTIME(n^c) \tag{1}$$

This class denotes the set of all functions that can be computed in polynomial time.The constant $c \geq 1$ implies that we have to read the entire input string at least once which is essential if for any possible input we need to compute the exact answer.

Following are some examples of natural problems in P:

- Graph Connectivity
- Cycle Detection
- Linear Programming
- Primality Testing
- Shortest Path

The class of problems in P are considered to be tractable or efficiently computable. One of the motivating reasons for this is the Strong Church-Turing hypothesis which states that Turing Machine can simulate any physical model of computation with just a polynomial time overhead.This implies that the class of problems defined by P is invariant to the model of the computation.

While defining the class P we have restricted our attention to boolean functions,if we remove this restriction and consider arbitrary functions $f : \{0,1\}^* \rightarrow \{0,1\}^*$ that can be computed in polynomial time then the set of these functions defines a new class called Functional Polynomial Time(FPC).

2.3 Class NP

Informally the class NP denotes the set of all problems whose solution can be efficiently verified.Formally we say that a language $L \subset \{0,1\}^*$ is in NP if there exists a deterministic polynomial time turing machine M and a polynomial function $q : \mathbb{N} \rightarrow \mathbb{N}$ such that $x \in L$ iff $\exists u \in \{0,1\}^{q(\|x\|)}$ such that M(x,u) = 1. M is called the verifier for L and u is called the certificate / witness for x. Following are examples of some natural problems in NP:

- Vertex Cover: Given a graph G and an integer k whether the graph G has vertex cover of size k.Here the certificate is the set of k vertices and given such a set it can be verified in polynomial time whether it forms a vertex cover or not.
- Independent Set:Given a graph G and a number k, find if there exists a k size independent subset of G's vertices.By independent we mean no two vertices in subset should have an edge between them.Here the certificate is the set of k vertices and it can be easily verified in polynomial time whether they form an independent set or not.
- Linear Programming:Given a list of m linear inequalities over n variables with rational coefficients, decide if there is an assignment of variables in rationals that satisfies the inequalities. The certificate is the assignment.
- Integer Linear Programming:Given a list of m linear inequalities over n variables with rational coefficients decide if there is an assignment of variables in integers that satisfies the inequalities.The certificate is the assignment.

- Graph Isomorphism:Given two graphs G_1 and G_2 such that $|V(G_1)| = |V(G_2)|$ decide if there exists a permutation $\sigma : V(G_1) \to V(G_2)$ such that $G_2 = \sigma(G_1)$.The certificate is the permutation and once the permutation is given we can verify in polynomial time whether the graphs are same under the given permutation

- Factoring: Given three numbers N, L, U determine whether N has a factor M between L and U.The certificate is the factor M and one can easily check in ploy nodal time whether M divides N or not.