Introduction

CSC363: Computational Complexity and Computability

Mohammad Mahmoud

This lecture is co-hosted by one of the TAs:

Paul Zhang

- **Mohammad Mahmoud**

Cairo University
BSc, 2010
MSc, 2014

PhD, 2019

Post-doc

Master's of Statistics

Ryerson University

UNIVERSITY OF TORONTO MISSISSAUGA

About the Course

Course Outline

- You can find it at the Office of the Registrar Timetable page https://student.utm.utoronto.ca/timetable/

- It includes all information regarding **Office hours**, **grading**, and **deadlines**

- Also includes a **course description**, **learning outcomes**, and a list of **references**

Assessments

- 5 easy short quizzes (3% each): The first will be on Friday the 22nd

- 5 assignments (10% each): The first will be on Friday 29th

- No midterm
- No programming
- Final exam (35%). You need to score at least 40% on the exam itself.

Computability

When we were kids

- Learned how to count: 1,2,3 …

- Then, how to add + : through counting (oo + ooo = ooooo)

- Later, how to multiply x: repeated addition

It was computation (calculation)

Main characteristics:

- There is a procedure
- There is some initial primitive operation/function (e.g. successor)

Now we are older

- Our procedures are now algorithms

- We build algorithms using some primitive functions

- We compute functions using algorithms

Strong Intuition

- Most of the time we are comfortable with our intuitive understanding of what an **algorithm** is and what **computable** is.

- Informally, *an **algorithm** is a finite sequence of implementable instructions*.

- *A function is **computable** if there is an algorithm to compute it.*

Do we need formality?

- That moment when mathematicians realized that they were trying to find an algorithm to solve a problem which is unsolvable by an algorithm.

Hilbert's 10^{th} Problem (H10)

- In 1900, David Hilbert published 23 problems. The 10^{th} was:

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

H10 in modern terms

- Find an algorithm which, for any given Diophantine equation, it can decide whether the equation **has** a solution with all unknowns taking integer values.

- Note that the algorithm here is meant to be general. In other words, it is the same for any given Diophantine equation.

- Also note that it does not even ask for finding a solution. Just decide if exists or not.

70 years after Hilbert’s request

- The 23 year old Yuri Matiyasevich proved that THERE IS NO such algorithm

Remarks

- Yuri just added the final touch to a work built by many others (Martin Davis, Hilary Putnam, and Julia Robinson).

- The feeling that the problem is unsolvable had been around since 1944

- That feeling was declared by Emil Leon Post

- H10 is not the first problem of that kind, but it is one we can introduce at the moment without much background

The need for formality

- Proving unsolvability by an algorithm requires a solid formulation of which processes we can accept as algorithms

- We should be able to create a list of all algorithms $\{P_e : e \in N\}$

(algorithms can be thought of as strings, the set of strings is countable)

- Then show that: for all e, P_e does not solve the given problem.

Formal Approaches

- One approach is formalizing the definition of *computable function* directly (Kurt Gödel)

- Another approach is “mechanical”: an algorithm is a computer program (Alan Turing, Emil Leon Post)

- A computer here is an abstraction of how “any” computer works

- There are several such abstract computer models. We will discuss the Turing Machine in this course.

More about H10

I will not be asking you details about this topic. The goal is the big picture and to introduce some knowledge which should ring a bell later in this course. Also to make you look things up and read around the topic.

That being said, many proofs will be omitted.

Some of the content will be hard at first, and that’s ok. You will take your time later to digest it. It will grow your muscles.

Diophantine Equations

- *Diophantine* refers to the mathematician Diophantus of Alexandria (3rd century)

- A Diophantine equation is a polynomial equation in more than one unknown in which all the coefficients are integers (in **Z**).

Recall: **Z** = {…,-3,-2,-1,0,1,2,3,….}

- We only care about integer solutions

Diophantine Equations: Examples

- $3x^2 - 2xy - y^2z - 7 = 0$ has the following integer solution

x = 1, y = 2, z = -2

Indeed, $3(1)^2 - 2(1)(2) - (2)^2(-2) - 7 = 3 - 4 + 8 - 7 = 0$

- $x^2 + y^2 + 1 = 0$ has no integer solutions

Careful, this isn't the circle equation

Diophantine Equation: General Form

- $p(a_1, \ldots, a_k, x_1, \ldots, x_l) = 0$

where $p(\ldots)$ stands for a polynomial function, the x_i's can be the unknowns and the a_i's can be some (or all) of the coefficients (parameters of particular interest).

Example: $3x^2 - 2xy - y^2z - 7 = 0$ can be $p(3, -2, -1, x, y, z) = 0$

where $p(a_1, a_2, a_3, x, y, z) = a_1x^2 + a_2xy + a_3y^2z - 7$

- Or can be $q(7, x, y, z) = 0$ where

$$q(a, x, y, z) = 3x^2 - 2xy - y^2 z - a$$

- It can take many forms

Diophantine Sets

- A Diophantine set is a subset S of N^k(tuples of natural numbers) for which there exists a Diophantine equation $p(\bar{x}, \bar{y}) = 0$ such that

$$S = \{\bar{a} : (\exists \bar{m} \in N^l)(p(\bar{a}, \bar{m}) = 0)\}$$

- Here $\bar{x}$ is a tuple of unknowns of length k, and $\bar{y}$ is a tuple of unknowns of length l. Also, *N* = {0,1,2,…}.

- We say S is definable by the formula $\varphi(\bar{x}) = (\exists \bar{m})(p(\bar{x}, \bar{m}) = 0)$ in *N* .

Diophantine Set: Example

- Consider the set

  $$S = \{a \in N: (\exists m_1, m_2 \in N)(m_1 m_2 + 2m_2 + 2m_1 + 4 - a = 0)\}$$

Here

$$p(a, x, y) = xy + 2y + 2x + 4 - a = (x + 2)(y + 2) - a$$

- $p(a, x, y) = 0$ iff $a = (x + 2)(y + 2)$

- S is exactly the set of numbers >2 which are not prime.

- The set S is basically the set of parameter tuples $\bar{a}$ for which the Diophantine equation $p(\bar{a}, \bar{y}) = 0$ has a solution in the **natural** numbers

- Recall that when we stated H10, we were talking about integer solutions (not natural)

- In fact, in the statement of H10, if we restrict the problem to finding natural solutions, then the version we get is still equivalent.

H10 (practical version)

- Find an algorithm which, for any given Diophantine equation, it can decide whether the equation has a solution with all unknowns taking natural values.

- Both versions are equivalent in the sense that, if you find an algorithm for one of them, then you can find an algorithm for the other.

- Proving the equivalence is omitted but not hard. It uses a handy fact called **Lagrange's four-square theorem**

Lagrange’s four-square theorem

- *Every natural number is the sum of the squares of four integers*

- Example: $31 = 5^2 + 2^2 + 1^2 + 1^2$

Time to Recollect

Short break

H10 and Diophantine sets

- There is the following connection between Diophantine sets and Hilbert's problem which is:

The existence of an algorithm as mentioned in H10

implies

We can 'uniformly' decide membership for any Diophantine set

Decidability

- We mean by that, there is an algorithm which when given any Diophantine set S (subset of N^k say), and any $\bar{a} \in N^k$, it can determine if $\bar{a}$ belongs to S or not.

- In general, a set is *decidable* if there is an algorithm that can decide its membership, e.g., the set of prime numbers.

What Yuri did

- Yuri filled the last piece of proving that (known as MRDP theorem):

A set is Diophantine iff it is *listable*

- Listable means there is an algorithm (program) that lists its elements. In other words, pops its elements one after the other.

- Listable is also known as *computably enumerable (c.e.)*

Listable vs Decidable

- Every decidable set is listable (easy to prove)

- The converse is not true (easy once we formally settle the definition of an algorithm or that of a computable function)

- In other words, there exists a set *H* which is listable but not decidable

Let’s trace things back

- *H* is listable but not decidable

- By MRDP, *H* is Diophantine but not decidable

- The existence of a Diophantine set which is not decidable implies that H10 isn't solvable (Contra-positive).

- There is the following connection between Diophantine sets and Hilbert’s problem which is:

The existence of an algorithm as mentioned in H10

implies

We can ‘uniformly’ decide membership for any Diophantine set

Done with the H10 story

Take this

- Formal Computability is either Gödel computability, or Turing computability. Both can be proved equivalent.
- Turing computability is mechanical: *An algorithm is a Turing Machine (TM)*
- A set is *listable* if it can be listed by a TM
- A set is *decidable* if a TM can decide its membership (compute its characteristic/indicator function)
- Every decidable set is listable but not the other way (intuitively kinda obvious, but the proof is through formal computability)
- This implies that H10 is unsolvable (by MRDP which particularly relies on Gödel computability)

Gödel Computability

Recall again when we were little kids

- Counting: **successor** operation

- Then addition: repeating the successor

- Multiplication: repeating addition

- That repetition process can be generally seen as **recursion**

Mimicking baby math

- Gödel decided to mimic early calculation

- Defined some basic initial functions

- Then used some fundamental primitive rules to build with them

Initial Functions

1. The **zero function**: $\mathbf{0}(n) = 0$ for every *n* in *N*

2. The **successor function**: $s(n) = n + 1$ for every *n* in *N*

3. The **projection functions**: U_i^k

The *k* indicates that its input is a tuple of length *k*, and *i* means it outputs the *i*-th component. More precisely,

$$U_i^k(\overline{m}) = m_i$$

Reinforcing Intuition

- Note that those initial functions are directly computable by instinct

- The zero function is like erasing, nothingness

- The successor is what takes us from nothing to something
- The projection is literally choice, ability to distinguish

What could be simpler?!

Beautiful!

Primitive Recursive Functions (PRIM)

- Now it is time to build more complex functions
- We think like induction
- Initial Functions are the base
- Assume we have a bunch of functions that have been built
- We describe how to use the latter to build the next ones (one step more complex)