Computational Complexity and Computability

Lecture 1 - Introduction, Motivation & History

Koushik Pal

University of Toronto

January 11, 2021

Computability Theory & Computable Functions

Question

- What is an algorithm?
- What is a computable function?

Informally, an algorithm is a sequence of computational steps that converts an input set to an output set.

Informally, a computable function is a function $f : A \to B$ such that there is a *mechanical procedure* for computing the result $f(a) \in B$ for every $a \in A$.

Computability Theory is the study of computable functions and the limits of this notion.

Computable Functions - Examples & Non-examples

Example

- $add : \mathbb{N} \times \mathbb{N} \to \mathbb{N} : (a, b) \mapsto a + b$ ✓
- $exp : \mathbb{N} \to \mathbb{N} : n \mapsto 2^n$ ✓
- $check : String \to \{0, 1\}$, where

$$check(p) = \begin{cases} 1 & \text{if } p \text{ is a syntactically valid Python program} \\ 0 & \text{otherwise.} \end{cases}$$ ✓

- $terminate : String \to \{0, 1\}$, where

$$terminate(p) = \begin{cases} 1 & \text{if } p \text{ is a syntactically valid Python program} \\ & \text{without any user interaction that terminates} \\ 0 & \text{otherwise.} \end{cases}$$ ✗

Computable Functions - Examples & Non-examples

Example

- $diophantine_1$: {Polynomials in 1 variable} $\to \{0, 1\}$, where

$$diophantine_1(p) = \begin{cases} 1 & \text{if } p \text{ has an integer solution} \\ 0 & \text{otherwise} \end{cases} \quad ✓$$

- $diophantine_n$: {Polynomials in n variables} $\to \{0, 1\}$, where

$$diophantine_n(p) = \begin{cases} 1 & \text{if } p \text{ has an integer solution} \\ 0 & \text{otherwise.} \end{cases} \quad ✗$$

Problems in Computability

- Precise definition of "computable" - mathematical definition
- Show that the definition is "correct" - philosophical argument
- Develop methods for showing some functions are computable and some are not - computer science
- When can the calculations be done effectively? - Complexity theory
  - Does a particular problem have a fast solution? - Time Complexity
  - Does a particular problem have a solution requiring a small amount of memory? - Space Complexity
- Can the task of deciding one problem be reduced to deciding another problem? - Reducibility theory

Different areas

- Mathematics
- Philosophy
- Computer Science

By the end of the 19th century, mathematics had become more rigorous and mathematical logic had also been developed.

Complexity theory has its roots in studies in mathematical logic and these earlier questions about maths.

History

Leibnitz (1670) Leibnitz built a first mechanical calculator. He was thinking about building a machine that could manipulate symbols in order to determine the truth values of mathematical statements. He noticed that a first step would be to introduce a precise formal language, and he was working on defining such a language.

Gauss (1798) Gauss in Disquisitiones Arithmeticae wondered about if there is an efficient algorithm for factoring integers.

History

Hilbert (1900) Hilbert announced a list of 23 problems to the mathematics world, many of which proved to be very influential for 20th-century mathematics. The 10th problem on his list turned out to be fundamental for Computability Theory: *Is there an algorithm to decide if a given multivariable polynomial* $p(x_1, \ldots, x_n)$ *has an integer solution?*

Hilbert (1928) Hilbert had already developed a theory for formalizing mathematical proofs. He posed the **Entscheidungsproblem** (decision problem): *Is there an algorithm that decides whether a mathematical formula is a consequence of his theory?*

History

Godel (1931) Godel in his famous Incompleteness Theorem showed that any consistent formal recursive system F within which a certain amount of elementary arithmetic can be carriedout is incomplete, i.e., there are statements of the language of F which can neither be proved nor disproved in F.

By the Church-Turing thesis, this implies that Entscheidungsproblem is unsolvable.

History

Figure: Stephen Cole Kleene

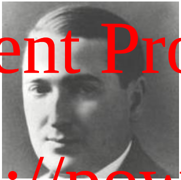


Figure: Emil Post

Figure: Alonzo Church

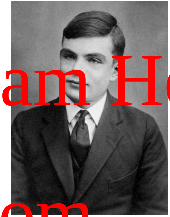


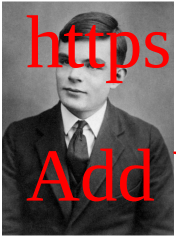
Figure: Alan Mathison Turing

Kleene, Post, Church, Turing (1930s) – different models of computability

Church-Turing Thesis (1936) *All these models are equivalent in computation power!*

History

Church (1936) Church shows the undecidability of equality in the λ-calculus.



Turing (1936) Turing shows the unsolvability of the halting problem. That problem turns out to be the most important undecidable problem.

Post (1944) studies degrees of unsolvability, gives birth to degree theory.

History

Figure: Yuri Matiyasevich

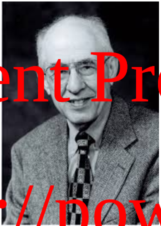


Figure: Hilary Putnam

Figure: Martin Davis

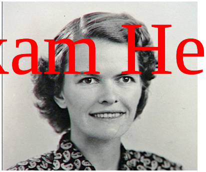


Figure: Julia Robinson

Matiyasevich (1970) Matiyasevich, building on works of Putnam, Davis and Robinson, gave a negative answer to Hilbert's 10th problem.

History

Cook (1971) Cook introduces P and NP, proves NP-Completeness for SAT, and poses the famous question: *is* $P = NP$*?*

Today

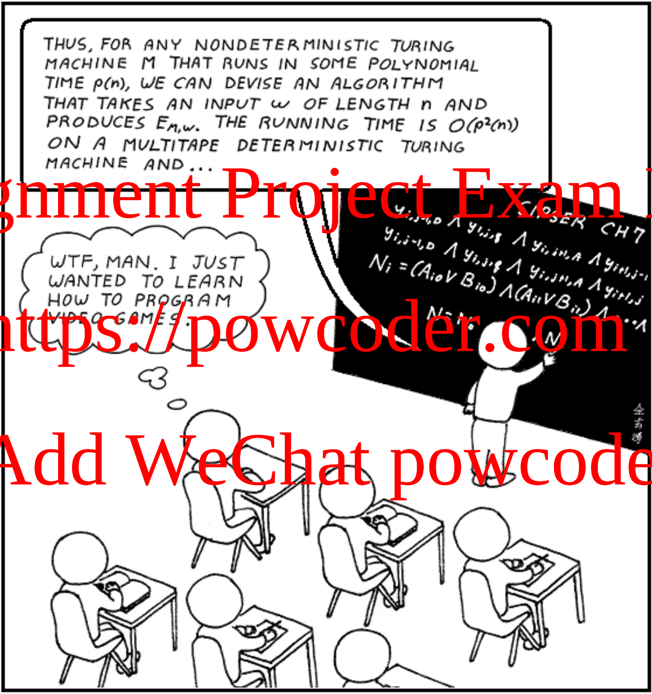
- P = NP? is still an open question.
- Complexity Theory has become a big research area.
- Lots of complexity classes defined for various models of computation.
- Alternative models of computation are studied, for example, quantum computing, genetic algorithms
- Parallel lines of research in mathematical logic, where computability theory is known as recursion theory and computable functions are called recursive functions.

Recap

- Language
- Regular language
- Deterministic Finite automaton (DFA)
- Non-deterministic Finite Automata (NFA)

Theorem

A language is regular
$\iff$ *some regular expression describes it*
$\iff$ *a DFA recognizes it*
$\iff$ *an NFA recognizes it.*


THUS, FOR ANY NONDETERMINISTIC TURING MACHINE M THAT RUNS IN SOME POLYNOMIAL TIME $p(n)$, WE CAN DEVISE AN ALGORITHM THAT TAKES AN INPUT ω OF LENGTH n AND PRODUCES $E_{M,w}$. THE RUNNING TIME IS $O(p^2(n))$ ON A MULTITAPE DETERMINISTIC TURING MACHINE AND...
WTF, MAN. I JUST WANTED TO LEARN HOW TO PROGRAM VIDEO GAMES.
$N_i = (A_{i0} \vee B_{i0}) \wedge (A_{i1} \vee B_{i1}) \wedge \ldots \wedge$