

Computation Theory

David Hicks

Aalborg University Esbjerg

Schedule for lecture day 1

- practical information
- introduction to course content
- history of theory of computability
- get started with Chapter 1....

Practical Info (1)

- instructor: David Hicks
 - email: hicks@cs.aue.auc.dk
 - phone: 7912 7632
 - office: 2.46 (FUV Building)
- course web page: <http://cs.aue.auc.dk/courses>
 - look in the F6S area under “complexity and computability”
 - short course description
 - course syllabus

Practical Info (2)

- lectures will follow “standard” model
 - 45 minutes * 2 + exercise time
- textbook for the course:
 - “Elements of the Theory of Computation” (ETOC)
 - by Lewis, Harry R. and Papadimitriou, Christos H.
 - available in AUE bookstore
- important suggestions:
 - keep up with the reading!
 - do the exercises!
- always refer to web page version for latest changes and updates to course schedule

Introduction (1)

- primary focus of the course: computability and complexity theory
 - the theoretical foundations of computer science
- representative questions:
 - what can a computer do?
 - what can a computer not do?
 - what problems cannot be solved?

Introduction (2)

- to study such questions, will develop *mathematical models* of computation
 - of varying levels of complexity
- will require toolset along the way consisting of:
 - mathematical techniques:
 - set theory, induction, functions, relations, and other elements of discrete math
 - other devices, including those from language theory, useful in addressing computability questions

History of Computability Theory (a very brief overview)*

- field essentially resulted from series of timely coincidences
 - developments from seemingly unrelated fields that occurred around same time
- set theory
 - developed by Georg Cantor
 - early 1900s
 - many “reasonable results” proven
 - however, paradoxes arose
 - especially concerning results for infinite sets

*summarized/adapted from “Introduction to Computer Theory” by Cohen, Daniel.

History

(continued)

- David Hilbert
 - famous mathematician of the day
 - set agenda for mathematical research
 - was quite annoyed with paradoxes in set theory
 - prominent on agenda was to “fix” set theory
 - advocated development of axiomatic system for set theory (much like Euclid’s for geometry)
 - would enable results to be rigorously proven for set theory, and hopefully eliminate paradoxes

History

(continued)

- Hilbert conjectured:
 - *not only is every provable result true, but every true result was provable*
- Hilbert wanted a methodology that would show how to find proofs for true results
 - something analogous to what linear algebra provides for solving systems of linear equations
 - given a problem to be proven, methodology would provide clear and reliable way to find a proof
 - this was essentially “a demand for an automated way to solve mathematical problems”
 - input the problem to be proved, the machine generates a proof

History

(continued)

- mathematicians were perplexed
 - they were used to proving things themselves
 - not creating a “proof generating technique”
- they needed a branch of mathematics that deals with *algorithms* (or procedures or programs)
- note, even though before invention of electronic computing devices, people were starting to think about and discuss *programs*
 - what they could and could not do

History

(continued)

- as research progressed, some difficulties were encountered
 - Kurt Gödel in 1931 proved there was no algorithm possible to provide proofs for all true statements!
 - showed that
 - either some true statements have no proofs, or
 - some false statements existed that did have proofs of correctness
 - forced mathematicians to regroup/rethink their pursuits
 - question now became:
 - “what statements do have proofs, and how can we generate these proofs?”

History

(continued)

- many researchers looked at this question including:
 - A. Church, S. Kleen, E. Post, A. Markov, J. von Neumann, A. Turing
- though they worked mostly independently, each came up with an extraordinarily simple set of building blocks
 - these blocks seemed to be basic atoms from which all mathematical algorithms can be comprised
 - they essentially had developed a universal model for algorithms, a “universal algorithm machine”

History

(continued)

- Turing took it one step further
 - showed there were some questions about the “universal algorithm machine” that the machine itself could not answer
 - means that there was no hope of ever achieving Hilbert’s program for completely mechanizing mathematics
 - though it shattered hopes of pursuing Hilbert’s goal, all was not lost.....

History

(continued)

- people realized that Turing's theoretical algorithm machine was a basic one
 - used only very simple set of mathematical structures
- might make it possible to build a physical realization of this theoretical machine
- if this could be done, human might devise algorithms for solving mathematical problems and have them performed on this (physical) machine
 - would allow machines to automatically and precisely solve problems

History

(continued)

- one of components needed to physically realize such a machine was *vacuum tube*
- vacuum tubes had, coincidentally, been developed around same time (for completely different intended purpose)
- plan to build such a machine was speculative and expensive
 - military finally commissioned project in support of their code-breaking efforts
 - Turing himself took part in construction of machine
- “...the rest is history....”
- what began in the theoretical world of mathematical abstraction became one of most successful inventions ever