CS 373

Theory of Computation

Spring 2009

Sariel Har-Peled
sariel@cs.uiuc.edu

Madhusudan Parthasarathy (Madhu)
madhu@cs.uiuc.edu

What is computable?

- Examples:
  - check if a number n is prime
  - compute the product of two numbers
  - sort a list of numbers
  - find the maximum number from a list
- Hard but computable:
  - Given a set of linear inequalities, maximize a linear function

    Eg. maximize 5x+2y

    3x+2y < 53

    x < 32

    5x – 9y > 22

Theory of Computation

Primary aim of the course:

- *What is "computation"?*
- *Can we define computation without refering to a modern computer?*
- *Can we define, mathematically, a computer? (yes, Turing machines)*
- *Is computation definable independent of present-day engineering limitations, understanding of physics, etc.?*
- *Can a computer solve any problem, given enough time and disk-space? Or are they fundamental limits to computation?*

In short, *understand the mathematics of computation*

Theory of Computation

Computability	*- What can be computed?* *- Can a computer solve any problem, given enough time and disk-space?*
Complexity	*- How fast can we solve a problem?* *- How little disk-space can we use to solve a problem*
Automata	*-What problems can we solve given really very little space? (constant space)*

Theory of Computation

What problems can a computer solve?

Computability	*Not all problems!!!* *Eg. Given a C-program, we cannot check if it will not crash!*
Complexity	Verification of correctness of programs is hence impossible! (The woe of Microsoft!)
Automata	

Theory of Computation

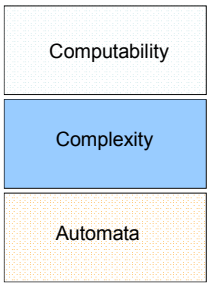
What problems can a computer solve?

Computability	Even checking whether a C-program will halt/terminate is not possible!

Complexity	input n;
	assume n>1;
	while (n !=1) {
	if (n is even)
	n := n/2;
	else
	n := 3*n+1;
	}
	17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Automata	**No one knows whether this terminates on on all inputs!**

Theory of Computation



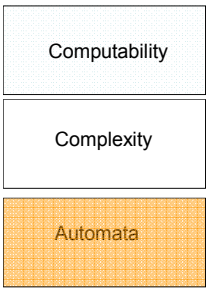
How fast can we compute a function?
How much space do we require?

- **Polynomial time computable**
- **Non-det Poly Time (NP)**
- **Approximation, Randomization**

Functions that cannot be computed fast:

- **Applications to security**
  - **Encrypt fast,**
  - **Decryption cannot be done fast**
- **RSA cryptography, web applications**

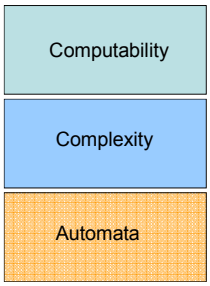
Theory of Computation



Machines with finite memory:-- traffic signals, vending machines hardware circuits

Tractable.
Applications to searching, verification of hardware, etc.

Theory of Computation

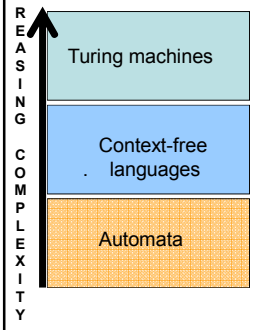


What can we compute?
-- Most general notions of computability
-- Uncomputable functions

What can we compute fast?
-- Faster algorithms, polynomial time
-- Problems that cannot be solved fast:
* Cryptography

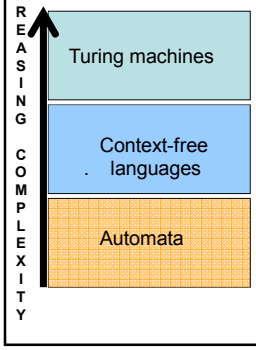
What can we compute with very little space?
-- Constant space (+stack)
* String searching, language parsing, hardware verification, etc.

Theory of Computation



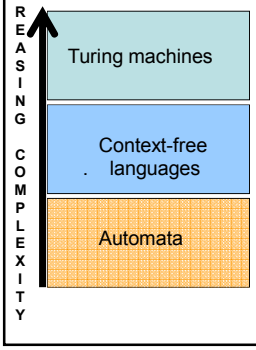
Automata:
--- Foundations of computing
--- Mathematical methods of argument
--- Simple setting

Theory of Computation



Context-free languages
--- Grammars, parsing
--- Machines with stack
--- Still a simple setting; but infinite state

Theory of Computation

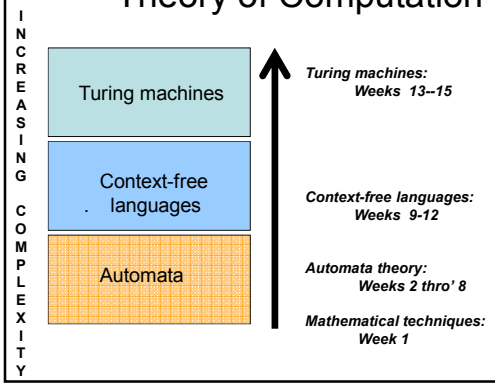


Turing machines (1940s):
-- The most general notion of computing
-- The Church-Turing thesis
-- Limits to computing:
Uncomputable functions

Motivation from mathematics:

- Can we solve any mathematical question *methodically*?
- Godel's theorem: NO!
- "Even the most powerful machines cannot solve some problems."

Theory of Computation



Kurt Gödel

- Logician extraordinaire
- Hilbert, Russel, etc. tried to formalize mathematics
- "Incompleteness theorem" (1931)
  - Cannot prove consistency of arithmetic formally
  - Consequence: unprovable theorems

Since proofs ←→ computation, non-computability was established

Kurt Godel: 1906 - 1978

Alonzo Church

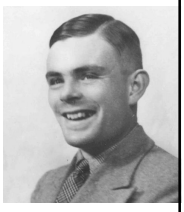
First notions of computable functions

First language for programs
-- lambda calculus
-- formal algebraic language for computable functions

Alonzo Church: 1903 - 1995

Alan Turing

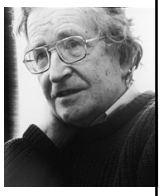
- "father of computer science"
- Defined the first formal notion of a computer (Turing machine) in 1936: "*On Computable Numbers, with an Application to the Entscheidungsproblem*"
- Proved uncomputable functions exist
- Church-Turing thesis: all real world computable
- functions are Turing m/c computable
- Cryptanalysis work breaking Enigma in WW-II



Alan Turing: 1912 - 1954

Noam Chomsky

- Linguist ; introduced the notion of formal languages arguing generative grammars are at the base of natural languages
- Hierarchy of formal languages that coincides with computation
- Eg. Context-free grammars capture most skeletons of prog. languages



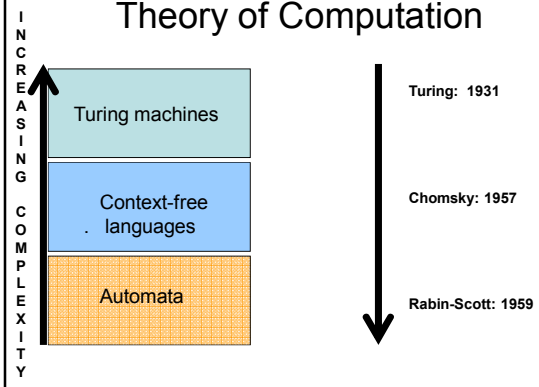
Noam Chomsky: 1928-

"Logical Structure of Linguistic Theory" (1957)

Automata theory

- *Automata: machines with finite memory*
- *"Finite Automata and Their Decision Problem" - Rabin and Scott (1959)*
- *Introduced nondeterministic automata and the formalism we still use today*
- *Initial motivation: modeling circuits*
- *Turing Award (1976)*

Theory of Computation



Goals of the course

- To understand the notion of "computability"
- Inherent limits to computability
- The tractability of weaker models of computation
- The relation of computability to formal languages

- Mathematics of computer science
  - Rigor
  - Proofs

A result you would know at the end...

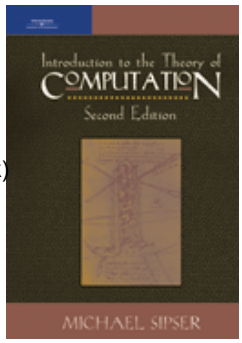
- *Proving that it is impossible to check if a C program will halt.*
- Formal proof!
- No computer *ever* will solve this problem (not even a quantum computer)

Textbook

Michael Sipser

Introduction to Theory of Computation
(2nd ed; 1st ed may be ok)

I will announce chapter readings that you must read before class.



Course logistics

- Section 1: Tu/Thu 11:00-12:15 Sariel Har-Peled
  Section 2: Tu/Thu 12:30-13:45 Madhusudan Parthasarathy.
  Lectures in SC 1105.
- Discussion sections (all in SC 1111): by TAs
  Tue 2:00 PM - 2:50 PM
  Tue 3:00 PM - 3:50 PM
  Tue 4:00 PM - 4:50 PM
  Wed 4:00 PM - 4:50 PM
- Announcements (homework posting announcements, discussions, corrections/clarifications):
  Newsgroup: class.cs373

Teaching assistants

- Micah Hodosh mhodosh2@illinois.edu,
- Aparna Sundar sundar2@illinois.edu,
- Reza Zamani zamani@uiuc.edu

Problem Sets

- Homeworks every week;
  handed out on Thursday,
  due in class by 12:30pm on Thursday.
- Write each problem on a separate sheet of paper.
  (allows distributed grading)
- Homework can be done in groups of at most three people.
- However, each student must hand in their own homework
  (no group submissions; must clearly write your group members)
- There may be additional "quizzes" (15min tests) at discussion sections as well.

Grading

- Two midterms - 20% each
- Final exam - 30%
- Homework/Quizzes - 25%
  (least scored HW not counted)
- Attendance to
  discussion sections - 5%

Curve

- Raw numerical scores tend to run low in theory classes; letter grades will primarily be decided based on relative ranking within the class.

Class Percentile	Grade
95 %	A+
85 %	A
80 %	A--
70 %	B+
60 %	B
50 %	B--
40 %	C+
30 %	C
20 %	C--
15 %	D+
10 %	D
5 %	D-
< 5 %	F

My lectures

- I will use a tablet PC; all class lecture slides will be posted online.
- Additional resources (on course webpage)
  - Sariel's and Margaret's lecture notes (Sp08)
  - Lecture notes (slides) from Fall'08
  - Review notes on main results you should learn/know (by me)
  - Old homeworks/solutions online
  - Probably too many resources!....

Honors?

- Honors students will do extra problems and a project.
- Please contact me after class if you intend taking this course as an honors course.

How to do well…

- This is essentially a math course:
  - you must learn the concepts well; if you don't there's almost no chance of success
  - if you do learn the concepts, there is very little else (facts, etc.) to learn; you can do really well!
  - You must do problems. There's no replacement for this.
  - Attending lectures is highly adviced!
    - It will be very hard to learn the concepts by yourself or from textbook.
  - Don't postpone learning; you will not be able to "make up" later. Topics get quickly hard.
  - Come regularly to discussion sections; you will learn a lot by working out problems and learn from fellow students

How to do well…

- Come to office hours!!
  - We are here to help you learn and do well.
- A new plan:
  - We will categorize homework problems, exam problems into various categories:
    - Machine construction, Proofs, Notation, Conceptual understanding, etc.
    - We will tell you how well you do in each category
  - We will also, at midterms, try to estimate how well you are doing and your projected grade. This will help you gauge your grade and overcome the panic of looking at low scores!