CS 373 Wrapup

Theory of Computation

Spring 2010

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

Theory of Computation

Primary aim of the course: *What is "computation"?*

- *Can we define computation without referring to a modern c computer?*

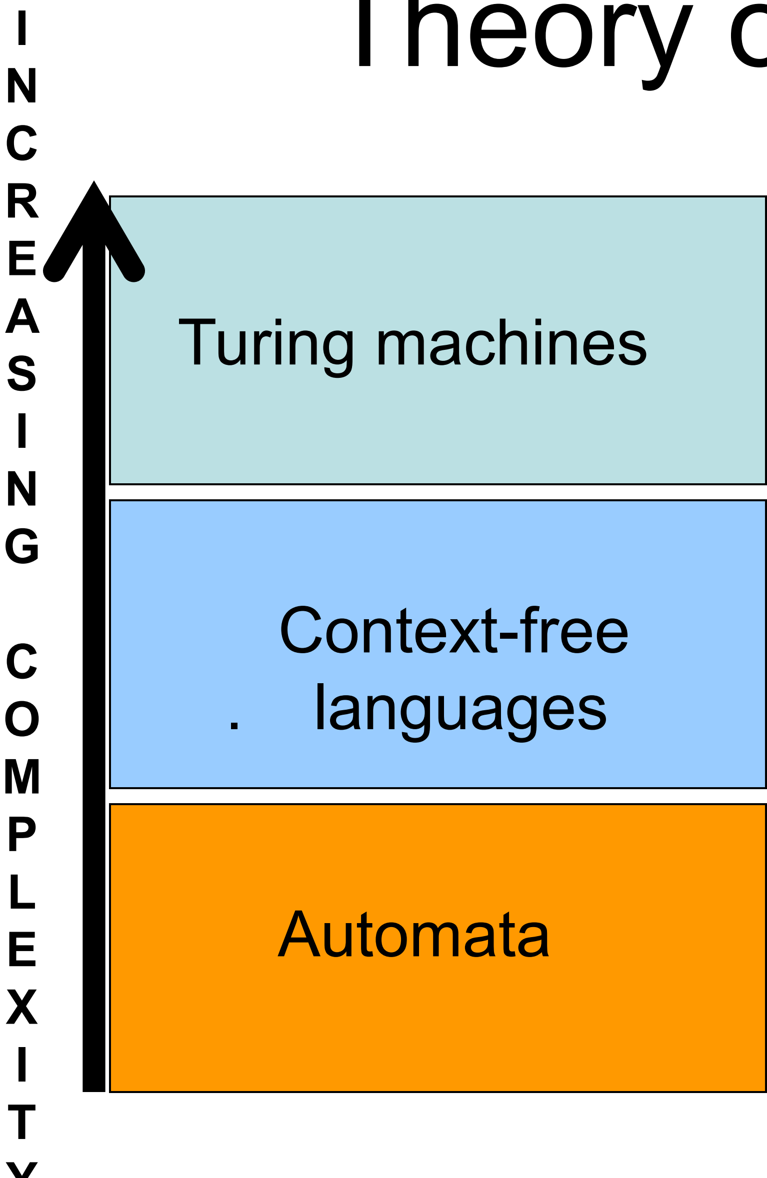
- *Can we define, mathematically, a computer?*
  (yes, Turing machines)

- *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?

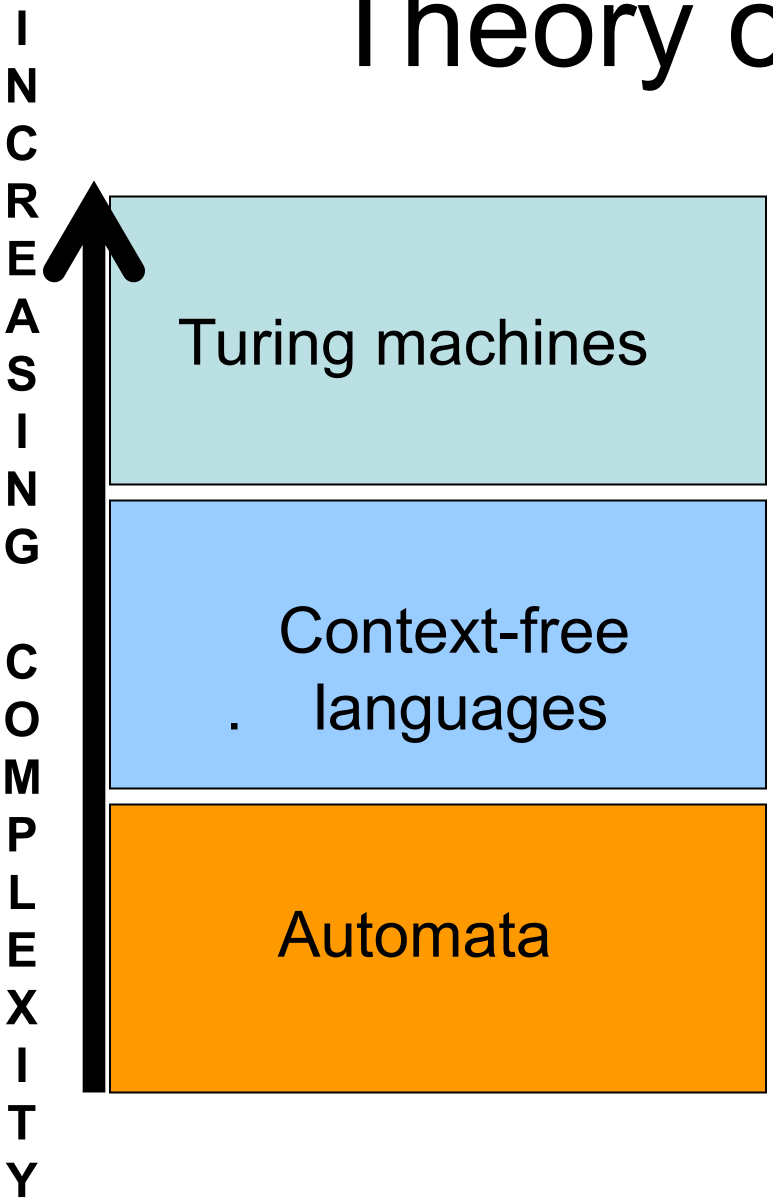
In short, *understand the mathematics of computation*

Theory of Computation



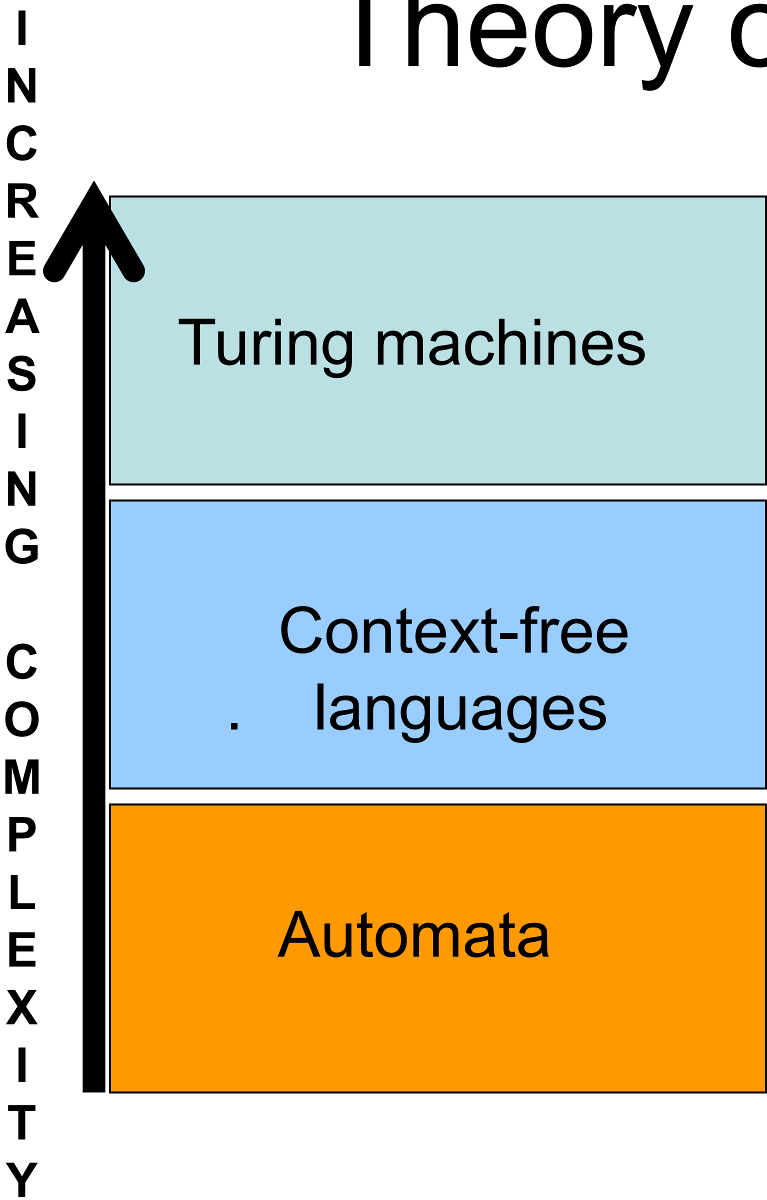
Automata:
--- Foundations of computing
--- Mathematical methods of argument
--- Simple setting

Theory of Computation



Context-free languages
--- Grammars, parsing
--- Finite state machines with recursion (or 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

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

Key classes

- Regular languages
  - Languages decided by finite-state machines
  - Robust, tractable
- Context-free languages
  - Languages expressed by CFGs
  - Decidable by machines
  - Semi-robust, semi-tractable
- Decidable Languages
  - The class of languages decidable using algorithms
  - Turing machine computable
  - Robust, not tractable

Uses

- Turing machines / decidable languages give
  the notion of algorithms and complexity

- Problems that you can model as regular languages (FSMs) or CFGs are more likely to have decidable algorithms.

- Notion of P and NP: classifying complexity of problems.

Regular Languages

- DFAs = NFAs = RegExp
- Closed under union, intersection, complement, concatenation, Kleene-*, reversal, homomorphism, …
- RegExp -> NFAs (closure properties)
- NFAs -> RegExp (constructing gen regexp)
- NFAs -> DFAs (subset construction; 2^n blowup)
- Suffix languages and Myhill-Nerodre thm:
  - L is regular iff L has finitely many suffix languages
    (used to show nonregularity or by using Pumping Lemma)
  - Hence minimal DFAs exist (one state for every suffix language)
  - Efficient minimization of DFAs.
- Decidable problems: L, L1, L2, … given as DFA/NFA/regexp
  - L = empty? ; L = Σ^*
  - L1 $\subseteq$ L2 ?; L1 = L2 ?
  - Closure properties followed by emptiness check.

Regular Lang - Applications

- Lexical analysis in compilers
  - Tokenizing keywords, “print”, “for”, etc.

- Searching for patterns
  - Text search for patterns
  - Datamining
  - Web search

- Modeling systems
  - FSMs describe models of systems and used for analysis
  - E.g. Physical systems (elevator), web browser, etc.
  - Tractability of analysis used: *model-checking*
  - *Hardware and software model-checking*

Context-free languages

- CFG = CFG in CNF = RA = PDA
- Closed under union, concatenation, Kleene-*, reversal, homomorphism, …
- Not closed under intersection, complement
- Membership problem is decidable: CYK algorithm --- parsing
- Decidable problems: L, L1, L2, … given as CFGs/RAs/PDA
  - w in L?
  - L = empty?
- Undecidable problems: $L1 \subseteq L2$; $L1=L2$; $L = \Sigma^*$
- Non-CFL: pumping lemma, corollary to pumping lemma

CFLs: Applications

- Parsing
  - Natural languages (semantic web; understanding speech, understanding text)
  - Programming languages (compilers)
- Recursive automata/PDAs
  - Modeling software control
    - Recursive procedures give recursive automata models
    - Static analysis of software done using these models
    - Compilers use them to check safety (types) and to do optimizations.
- XML
  - XML is basically bracketed text encoding hierarchical data
    - <car> <make> Honda </make> <year> 2002 </year> </car>
  - Data-type definitions – CFGs expressing valid XML documents
  - Conformance checking to DTDs, etc. are solvable.

Decidable languages

- Turing machines that halt
- Captures the class of problems solvable using “algorithms”
- Robust simple mathematical notion
  - independent of current knowledge of physics/engg
  - captures computability without using current prog lang
- Closure under union, intersection, complement, concatenation, Kleene-*, reversal
- Not closed under homomorphisms
- Nothing about the *language of a TM* is decidable (Rice’s thm)
- Undecidable problems
  - w in L?
  - L1=L2? L1 $\subseteq$ L2
  - L = empty? ; L = Σ^*

Undecidability

- A_TM = { <M,w> | M acc w } is undecidable
- Proof: Diagonalize TMs against words;

  Show L_d = { w_i | M_i does not accept w_i}

  is undecidable.

  Reduce L_d to A_TM.

- Reductions:
  - A reduces to B : Using a solution for B, a solution for A exists
  - If A reduces to B and B is decidable, then A is decidable.
  - If A reduces to B and A is undecidabe, then B is undecidable.
  - Use to show many more problems undecidable
    - Rice's theorem: Nothing about the language of a TM is decidable.
    - HALT = { <M> | M writes halts on starting from blank tape }

      is undecidable

A simple undecidable problem that does not refer to TMs

- Post's correspondence problem: Fix an alphabet Σ.

Given $2n$ words in Σ^*:
$w_1, w_2, \ldots, w_n, x_1, x_2, \ldots x_n$,
is there a set of indices $i_1, i_2, \ldots i_k$
($k > 0$ and each i_j between 1 and n) such
that

$$w_{i_1} w_{i_2} \ldots w_{i_k} = x_{i_1} x_{i_2} \ldots x_{i_k}?$$

E.g. If $w_1 = abbb$, $w_2 = b$, $x_1 = a$, $x_2 = bb$,
then it has a solution since
$w_1 w_2 w_2 w_2 = abbbbbb = x_1 x_2 x_2 x_2$.

- Undecidable!

A simple tiling problem that's undecidable

- You're given an infinite set of tiles; each tile is of type t where t belongs to a finite set, say T = {t1, t2, … tn}
- You're also given a set of rules of which tiles can occur horizontally next to each other and vertically next to each other.

This is given by two sets $H \subseteq TxT$ and $V \subseteq TxT$
If (t,t') is in H, a tile of type t and a tile of type t' can occur horizontally next to each other;
similarly for V.

- Is there a way to tile the infinite first quadrant such that all rules are respected?
- Undecidable!

Another simple undecidable problem

- Given a set of polynomial equations
  is there an integer valuation of the vars
  that satisfies the equations.

E.g $4x^3 + 9xy^2 = 49$

$10xy^3 + 7xy = 33$

Are there integer solutions for these eqs.?

Undecidable! (Hilbert's 10^{th} problem)

proved undecidable by Matiyasevich in 1970.