COMP3310/3610: Theory of Computation

Taso Viglas
tasos@it.usyd.edu.au

University of Sydney

July 24, 2007

Outline

1. Introduction
   - about this unit
   - on computation

2. Models of computation
   - Hilbert's tenth

3. Finite state automata
   - Formal languages

Teaching team:
taso viglas, tasos@it

M. Sipser, Introduction to the theory of computation, second edition

Plus material from the lectures

Plus material from the unpublished/in preparation book of Sanjeev Arora, "Computational Complexity: A Modern Approach" available online at:

Assessment (roughly):
- midterm exam: 15%
- two assignments: 10% each, plus 5% for tutorial work
- final exam: 60%
Assignments/Problem sets
Strict policy on collaboration for solving the problems from the assignments:
Please collaborate. a lot.
Two things about collaboration (or is it 3):
1. discuss problems you have not solved yet
2. write down the solutions on your own
3. list the names of the people with whom you discussed the assignment solutions on your assignment/submission

submit all work using webCT pdf, html
Check course homepage/webCT regularly!
http://www.it.usyd.edu.au/ comp3310
lectures and tutorials tuesdays 2-5p in SIT 123

Theory of Computation: 3 questions
What can we compute? (solve computationally)
What can we really compute?
Efficient algorithmic solutions, solve fast hardness
What do we do with hard problems?
- give up, forget about them
- give a lousy solution that we do not really have a clue about how bad it is
- deal with hardness
Theory of Computation

- Theory of Computation: 3 parts
  - Computability
  - Complexity
  - Algorithms

Computability

- models of computation
- from automata to Turing machines
- The halting problem and other not-computable problems
- The Church-Turing thesis
- Connections to logical theories

Complexity

- Complexity theory: what can we compute?
- Easy problems and hard problems
- Computational hardness
  - is it easier to multiply or to add numbers?
- Reductions
- Complexity classes and hierarchies
- The $P$ versus $NP$ problem

Algorithms

- Some problems are very hard. Give up?
- Dealing with hardness
  - Approximation algorithms
  - Randomized algorithms
Useful mathematical background
- Combinatorics, counting
- Probability theory, tail bounds
- Graph theory
- Some logic

Question: What is computation?
- Define simple models of computation
  - Automata, non-determinism, push-down machines, Turing Machines
- Equivalence between models
- Power for each model
- Church-Turing Thesis

Computation
- \( f(x) = y \)
- \( f : x \rightarrow y \)
- \( x \rightarrow (f) \rightarrow y \)
- \( f \): a program computing the mapping
- A sequence of steps that will compute \( y \) from \( x \)
- Definitions of a function
  - \( f(x) = 1 \) iff \( x \) is prime. Otherwise, \( f(x) = 0 \)
- \( f(x, y, z) = 1 \) iff there exists a positive integer \( n \) such that \( x^n + y^n = z^n \). \( f \) is zero otherwise.
- Assume \( x, y, z \) positive integers.
- Non-constructive definition
- What is the computation here?
- Two things to note:
  - Algorithms must always terminate
  - One-sided behaviour
Fermat’s Last theorem

• \( x^n + y^n = z^n \)
  • are there solutions for \( n \geq 3 \)?

Hilbert’s tenth

• David Hilbert, 1862-1943
• influential/famous presentation at the International Congress of Mathematicians in 1900
• Hilbert’s problems including the continuum hypothesis, Riemann hypothesis
• Hilbert’s tenth resolved, Matiyasevich’s theorem

Hilbert’s program

• 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.
• example of a diophantine equation: \( x^n + y^n = z^n \)
• Hilbert’s tenth is unsolvable (QED in 1970)
• Post, Davis, Robinson, Putnam, Matiyasevich

David Hilbert, 1862-1943
influential/famous presentation at the International Congress of Mathematicians in 1900
Hilbert’s problems including the continuum hypothesis, Riemann hypothesis
Hilbert’s tenth resolved, Matiyasevich’s theorem

Hilbert’s program: formalize all existing theories to a finite, complete set of axioms, and provide a proof that these axioms were consistent, and show that there is an algorithm for deciding the truth or falsity of any mathematical statement.

Gödel’s incompleteness theorems prove that this is basically impossible
Finite State Automata

- Finite state automata
- Regular expressions
- The pumping lemma

Computation

- Simple computation models
- Machines or programming models
- Question: "What can we compute?"
- Formalize the notion of computation

Notation

- Defining a set: \( A = \{x | x \text{ is a positive natural number with some property}\} \)
- Union: \( A \cup B \)
- Intersection: \( A \cap B \)
- Complement: \( \bar{A} \)
- Cartesian Product: \( A \times B = \{(x, y) | x \in A, y \in B\} \)
- \( \Sigma = \{0, 1\} \), then \( \Sigma \times \Sigma = \{(0, 0), (0, 1), (1, 0), (1, 1)\} \)
- Power set: \( \mathcal{P}(A) \)
- \( \Sigma = \{0, 1\} \), then \( \mathcal{P}(\Sigma) = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\} \)

Languages

- An alphabet is a (finite) set of symbols, \( \Sigma = \{a, b\} \)
- A word is a concatenation of alphabet symbols (a string over a given alphabet), \( w = abba \)
- A language is a set of words over a given alphabet
  - \( L = \{a, aa, abba\} \), or
  - \( L = \{\text{all words starting with an } a\} \)
  - \( L = \{a^n b^n | n \in \mathbb{N}\} \)
Let $L$ be a language

- A machine $M$ recognizes $L$ if $M(x) = 1$ if $x \in L$, otherwise $M(x) = 0$
- $M$ accepts all strings in the language $L$ and rejects all other strings

A finite state machine, reads its input once (left to right) and accepts or rejects

Finite automaton has fixed size memory

Example of an automaton

**Finite automaton: formal definition**

- Def: A deterministic finite state automaton (DFA) $M$ is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
  - $Q$ finite set of states
  - $\Sigma$ finite alphabet
  - $\delta$ transition function, $\delta : Q \times \Sigma \rightarrow Q$
  - $q_0 \in Q$ starting state
  - $F \subseteq Q$ accepting states

**The power of DFAs**

- Are there languages that are not recognized by DFAs?
- What kind of languages are recognized by DFAs?
- Non-deterministic automata
- FAs and Regular expressions
- Regular operations and closure properties
- Non-regular languages and the pumping lemma