CSE 2001:
Introduction to Theory of Computation
Summer 2013

Yves Lesperance
lesperan@cse.yorku.ca

Office: LAS 3052A

Course page: http://www.cse.yorku.ca/course/2001

Slides are mostly taken from Suprakash Datta’s for Winter 2013

Textbook:

Michael Sipser.
Introduction to the Theory of Computation,

Administrivia

Lectures: Mon, 7–10 pm (CLH K)

Office hours: Mon & Thu 5-6 pm, (CSEB 3052A), or by appointment.

TA: Paria Mehrani, who will lead tutorials (problem-solving sessions) and Wendy Ashlock

http://www.cse.yorku.ca/course/2001

Webpage: All announcements/handouts will be published on the webpage -- check often for updates)
Administrivia – contd.

Grading:

2 Midterms : 20% + 20% (in class)
Final: 40%
Assignments (4 sets): 20%

Grades will be on ePost (linked from the webpage)

Notes:

1. All assignments are individual.
2. There MAY be an extra credit quiz. This will be announced beforehand.

Plagiarism: Will be dealt with very strictly. Read the detailed policies on the webpage.

Handouts (including solutions): in /cs/course/2001, or on the webpage

Slides: Will usually be on the web the morning of the class. The slides are for MY convenience and for helping you recollect the material covered. They are not a substitute for, or a comprehensive summary of, the textbook.

Resources: We will follow the textbook closely.

There are more resources than you can possibly read – including books, lecture slides and notes.
Recommended strategy

• This is an applied Mathematics course -- practice instead of reading.
• Try to get as much as possible from the lectures.
• If you need help, get in touch with me early.
• If at all possible, try to come to the class with a fresh mind.
• Keep the big picture in mind. ALWAYS.

Course objectives - 1

Reasoning about computation
• Different computation models
  – Finite Automata
  – Pushdown Automata
  – Turing Machines
• What these models can and cannot do
Course objectives - 2

• What does it mean to say “there does not exist an algorithm for this problem”?

• Reason about the hardness of problems

• Eventually, build up a hierarchy of problems based on their hardness.

Course objectives - 3

• We are concerned with solvability, NOT efficiency.

• CSE 3101 (Design and Analysis of Algorithms) efficiency issues.

• Learn to make and prove assertions about computational models.
Reasoning about Computation

Computational problems may be
• Solvable, quickly
• Solvable in principle, but takes an infeasible amount of time (e.g. thousands of years on the fastest computers available)
• (provably) not solvable

Theory of Computation: parts

• Automata Theory (CSE 2001)
• Complexity Theory (CSE 3101, 4115)
• Computability Theory (CSE 2001, 4101)
Reasoning about Computation - 2

• Need formal reasoning to make credible conclusions
• Mathematics is the language developed for formal reasoning
• As far as possible, we want our reasoning to be intuitive

Next:

Ch. 0: Set notation and languages
• Sets and sequences
• Tuples
• Functions and relations
• Graphs
• Boolean logic: $\lor$, $\land$, $\neg$, $\Leftrightarrow$, $\Rightarrow$

• Review of proof techniques
• Construction, Contradiction, Induction…

Some of these slides are adapted from Wim van Dam’s slides (www.cs.berkeley.edu/~vandam/CS172/) and from Nathaly Verwaal (http://cpsc.ucalgary.ca/~verwaal/313/F2005)
Topics you should know:

- Elementary set theory
- Elementary logic
- Functions
- Graphs

Set Theory review

- Definition
- Notation: \( A = \{ x \mid x \in \mathbb{N}, x \mod 3 = 1 \} \)
  \( \mathbb{N} = \{1,2,3,\ldots\} \)
- Union: \( A \cup B \)
- Intersection: \( A \cap B \)
- Complement: \( \overline{A} \)
- Cardinality: \( |A| \)
- Cartesian Product:
  \( A \times B = \{ (x,y) \mid x \in A \text{ and } y \in B \} \)
Some Examples

\[ L_{\leq 6} = \{ x \mid x \in \mathbb{N}, x \leq 6 \} \]
\[ L_{\text{prime}} = \{ x \mid x \in \mathbb{N}, x \text{ is prime} \} \]
\[ L_{\leq 6} \cap L_{\text{prime}} = \{2, 3, 5\} \]

\[ \Sigma = \{0, 1\} \]
\[ \Sigma \times \Sigma = \{(0,0), (0,1), (1,0), (1,1)\} \]

Formal: \( A \cap B = \{ x \mid x \in A \text{ and } x \in B \} \)

13-05-09  CSE 2001, Summer 2013

Power set

“Set of all subsets”
Formal: \( \mathcal{P}(A) = \{ S \mid S \subseteq A \} \)

Example: \( A = \{x, y\} \)
\( \mathcal{P}(A) = \{ \emptyset, \{x\}, \{y\}, \{x, y\} \} \)

Note the different sizes: for finite sets
\[ |\mathcal{P}(A)| = 2^{|A|} \]
\[ |A \times A| = |A|^2 \]
Logic: review

Boolean logic: \( \lor \), \( \land \), \( \neg \)
Quantifiers: \( \forall \), \( \exists \)

statement: Suppose \( x \in \mathbb{N}, \ y \in \mathbb{N} \). Then \( \forall x \ \exists y \ y > x \)
for any integer, there exists a larger integer

\( \Rightarrow \): \( a \Rightarrow b \) “is the same as” (is logically equivalent to) \( \neg a \lor b \)

\( \Leftrightarrow \): \( a \Leftrightarrow b \) is logically equivalent to \( (a \Rightarrow b) \land (b \Rightarrow a) \)

Logic: review - 2

Contrapositive and converse:
the converse of \( a \Rightarrow b \) is \( b \Rightarrow a \)
the contrapositive of \( a \Rightarrow b \) is \( \neg b \Rightarrow \neg a \)

Any statement is logically equivalent to its contrapositive, but not to its converse.
Logic: review - 3

Negation of statements
\[ \neg (\forall x \exists y \ y > x) = \exists x \forall y \ y \leq x \]

(LHS: negation of “for any integer, there exists a larger integer”, RHS: there exists a largest integer)

TRY: \[ \neg (a \Rightarrow b) = ? \]

Logic: review - 4

Understand quantifiers
\[ \forall x \exists y \ P(y, x) \] is not the same as
\[ \exists y \forall x \ P(y, x) \]

Consider \[ P(y,x) : x \leq y. \]
\[ \forall x \exists y \ x \leq y \] is TRUE over \[ \mathbb{N} \] (set \[ y = x + 1 \])
\[ \exists y \forall x \ x \leq y \] is FALSE over \[ \mathbb{N} \] (there is no largest number in \[ \mathbb{N} \])
Functions: review

- \( f: A \rightarrow C \)
- \( f: A \times B \rightarrow C \)

Examples:
- \( f: \mathbb{N} \rightarrow \mathbb{N}, \ f(x) = 2x \)
- \( f: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, \ f(x,y) = x + y \)
- \( f: A \times B \rightarrow A, \ A = \{a,b\}, \ B = \{0,1\} \)

\[
\begin{array}{c|cc}
0 & 1 \\
\hline
a & a & b \\
b & b & a \\
\end{array}
\]

Functions: an alternate view

Functions as lists of pairs or k-tuples
- E.g. \( f(x) = 2x \)
- \( \{(1,2), (2,4), (3,6), \ldots \} \)
- For the function below:
  \( \{(a,0,a),(a,1,b),(b,0,b),(b,1,a)\} \)

\[
\begin{array}{c|cc}
0 & 1 \\
\hline
a & a & b \\
b & b & a \\
\end{array}
\]
Next: Terminology

- Alphabets
- Strings
- Languages
- Problems, decision problems

Alphabets

- An alphabet is a finite non-empty set.
- An alphabet is generally denoted by the symbols Σ, Γ.
- Elements of Σ, called symbols, are often denoted by lowercase letters, e.g., a,b,x,y,..
Strings (or words)

- Defined over an alphabet $\Sigma$
- Is a finite sequence of symbols from $\Sigma$
- Length of string $w$ ($|w|$) – length of sequence
- $\varepsilon$ – the empty string is the unique string with zero length.
- Concatenation of $w_1$ and $w_2$ – copy of $w_1$ followed by copy of $w_2$
- $x^k = x \times x \times x \times \ldots x$ (k times)
- $w^R$ - reversed string; e.g. if $w = abcd$ then $w^R = dcba$.
- Lexicographic ordering: definition

Languages

- A language over $\Sigma$ is a set of strings over $\Sigma$
- $\Sigma^*$ is the set of all strings over $\Sigma$
- A language $L$ over $\Sigma$ is a subset of $\Sigma^*$ ($L \subseteq \Sigma^*$)
- Typical examples:
  - $\Sigma = \{0,1\}$, the possible words over $\Sigma$ are the finite bit strings.
  - $L = \{ x \mid x \text{ is a bit string with two zeros } \}$
  - $L = \{ a^n b^n \mid n \in \mathbb{N} \}$
  - $L = \{ 1^n \mid n \text{ is prime} \}$
Concatenation of languages

Concatenation of two languages:
\[ A \cdot B = \{ xy | x \in A \text{ and } y \in B \} \]

Caveat: Do not confuse the concatenation of languages with the Cartesian product of sets.

For example, let \( A = \{0, 00\} \) then

\[ A \cdot A = \{ 00, 000, 0000 \} \text{ with } |A \cdot A|=3, \]

\[ A \times A = \{ (0,0), (0,00), (00,0), (00,00) \} \text{ with } |A \times A|=4 \]

Problems and Languages

- Problem: defined using input and output
  - compute the shortest path in a graph
  - sorting a list of numbers
  - finding the mean of a set of numbers.
- Decision Problem: output is yes/no (or 1/0)
- Language: set of all inputs where output is yes
Historical perspective

- Many models of computation from different fields
  - Mathematical logic
  - Linguistics
  - Theory of Computation

Formal language theory

Input/output vs decision problems

Input/output problem: “find the mean of n integers”
Decision Problem: output is either yes or no
“Is the mean of the n numbers equal to k ?”

You can solve the decision problem if and only if you can solve the input/output problem.
Example – Code Reachability

- Code Reachability Problem:
  - Input: Java computer code
  - Output: Lines of unreachable code.
- Code Reachability Decision Problem:
  - Input: Java computer code and line number
  - Output: Yes, if the line is reachable for some input, no otherwise.
- Code Reachability Language:
  - Set of strings that denote Java code and reachable line.

Example – String Length

- Decision Problem:
  - Input: String w
  - Output: Yes, if |w| is even
- Language:
  - Set of all strings of even length.
Relationship to functions

• Use the set of k-tuples view of functions from before.
• A function is a set of k-tuples (words) and therefore a language.
• Shortest paths in graphs – the set of shortest paths is a set of paths (words) and therefore a language.

Recognizing languages

• Automata/Machines accept languages.
• Also called “recognizing languages”.

• The power of a computing model is related to, and described by, the languages it accepts/recognizes.

• Tool for studying different models
Recognizing Languages - 2

• Let $L$ be a language $\subseteq S$

• A machine $M$ recognizes $L$ if

  $x \in S \rightarrow M$

  “accept” if and only if $x \in L$

  “reject” if and only if $x \notin L$

Recognizing languages - 3

• Minimal spanning tree problem solver

  cost

  tree

  Yes/no
Recognizing languages - 4

- Tools from language theory
- Expressibility of languages
- Fascinating relationship between the complexity of problems and power of languages

Graphs: review

- Nodes, edges, weights
- Undirected, directed
- Cycles, trees
- Connected
Proofs

- What is a proof?
- Does a proof need mathematical symbols?
- What makes a proof incorrect?
- How does one come up with a proof?

Proof techniques (Sipser 0.4)

- Proof by cases.
- Proof by contrapositive
- Proof by contradiction
- Proof by construction
- Proof by induction
- Others ….
Proof by cases
If n is an integer, then n(n+1)/2 is an integer
• Case 1: n is even.
  or n = 2a, for some integer a
  So n(n+1)/2 = 2a*(n+1)/2 = a*(n+1),
  which is an integer.
• Case 2: n is odd.
  n+1 is even, or n+1 = 2a, for an integer a
  So n(n+1)/2 = n*2a/2 = n*a,
  which is an integer.

Proof by contrapositive - 1
If x^2 is even, then x is even
• Proof 1 (DIRECT):
  x^2 = x*x = 2a
  So 2 divides x.
• Proof 2: prove the contrapositive!
  if x is odd, then x^2 is odd.
  x = 2b + 1. So x^2 = 4b^2 + 4b + 1 (odd)
Proof by contrapositive - 2

If $\sqrt{(pq)} \neq (p+q)/2$, then $p \neq q$

Proof 1: By squaring and transposing

$(p+q)^2 \neq 4pq$, or
$p^2+q^2 +2pq \neq 4pq$, or
$p^2+q^2 -2pq \neq 0$, or
$(p-q)^2 \neq 0$, or
$p-q \neq 0$, or $p \neq q$.

Proof 2: prove the contrapositive!

If $p = q$, then $\sqrt{(pq)} = (p+q)/2$

Easy: $\sqrt{(pq)} = \sqrt{(pp)} = \sqrt{(p^2)} = p = (p+p)/2 = (p+q)/2$.

Proof by contradiction

$\sqrt{2}$ is irrational

• Suppose $\sqrt{2}$ is rational. Then $\sqrt{2} = p/q$, such that $p$, $q$ have no common factors.

Squaring and transposing,

$p^2 = 2q^2$ (even number)
So, $p$ is even (previous slide)
Or $p = 2x$ for some integer $x$
So $4x^2 = 2q^2$ or $q^2 = 2x^2$
So, $q$ is even (previous slide)
So, $p,q$ are both even – they have a common factor of 2. CONTRADICTION.

So $\sqrt{2}$ is NOT rational. Q.E.D.
Proof by construction

There exists irrational $b, c$, such that $b^c$ is rational

Consider $\sqrt{2}^{\sqrt{2}}$. Two cases are possible:

- Case 1: $\sqrt{2}^{\sqrt{2}}$ is rational – DONE ($b = c = \sqrt{2}$).

- Case 2: $\sqrt{2}^{\sqrt{2}}$ is irrational – Let $b = \sqrt{2}^{\sqrt{2}}$, $c = \sqrt{2}$.
  Then $b^c = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = (\sqrt{2})^{\sqrt{2} \cdot \sqrt{2}} = (\sqrt{2})^2 = 2$

Debug this “proof”

For each positive real number $a$, there exists a real number $x$ such that $x^2 > a$

Proof: We know that $2a > a$
  So $(2a)^2 = 4a^2 > a$
  So use $x = 2a$. 
Proof by induction

Format:
- Inductive hypothesis,
- Base case,
- Inductive step.

Prove: For any $n \in \mathbb{N}$, $n^3 - n$ is divisible by 3.

IH: $P(n)$: For any $n \in \mathbb{N}$, $f(n) = n^3 - n$ is divisible by 3.
Base case: $P(1)$ is true, because $f(1) = 0$.
Inductive step:
Assume $P(n)$ is true. Show $P(n+1)$ is true.
Observe that $f(n+1) - f(n) = 3(n^2 + n)$
So $f(n+1) - f(n)$ is divisible by 3.
Since $P(n)$ is true, $f(n)$ is divisible by 3.
So $f(n+1)$ is divisible by 3.
Therefore, $P(n+1)$ is true.
Exercise: give a direct proof.
What is a Proof - continued?

“Everybody knows what a mathematical proof is. A proof of a mathematical theorem is a sequence of steps which leads to the desired conclusion. The rules to be followed by such a sequence of steps were made explicit when logic was formalized early in this century, and they have not changed since“


uid=3739448&uid=2129&uid=70&uid=3737720&uid=4&sid=21101181582701

Proof by contradiction - 2

The Pigeonhole Principle

• If $n+1$ or more objects are placed into $n$ boxes, then there is at least one box containing two or more of the objects

In a set of any 27 English words, at least two words must start with the same letter

• If $n$ objects are placed into $k$ boxes, then there is at least one box containing $\lceil n/k \rceil$ objects
Recursively defined sets

Close relationship to induction
Example: set of all palindromes
- $\varepsilon \in P$; $\forall a \in \Sigma$, $a \in P$;
- $\forall a \in \Sigma \forall x \in P$, axa $\in P$;
- No other strings are in P

More definitions

Definition of $\Sigma^*$:
- $\varepsilon \in \Sigma^*$;
- $\forall a \in \Sigma, \forall x \in \Sigma^*$, xa $\in \Sigma^*$;
- No other strings are in $\Sigma^*$. 
Exercise

Suppose $\Sigma = \{a,b\}$. Define $L$ as
- $a \in L$;
- $\forall x \in L, ax \in L$
- $\forall x, y \in L, bxy, xby$ and $xyb$ are in $L$
- No other strings are in $P$

- Prove that this is the language of strings with more $a$’s than $b$’s.

Next: Finite automata

Ch. 1: Deterministic finite automata (DFA)

Look ahead:
We will study languages recognized by finite automata.