CSCI 320:
Introduction to Theory of Computation

Spring 2015

Gara Pruesse

Gara.Pruesse@viu.ca

Office: 315/218

Course page:
http://www.csci.viu.ca/~gpruesse/teaching/320

Administrivia

Lectures: Tue, Thu 1 - 2:30 pm (200/105)
Seminar: Wed either 9-10 (355/108) or 4-5 (315/216

Office hours:
Tues, 2:30-3:30 or Wed 2-3 or by appointment.

http://www.csci.viu.ca/~gpruesse/teaching/320

Webpage: All announcements/handouts will be published on the webpage -- check often for updates) and I will email to the classlist re: urgent matters – using your “registration” email

Textbook:

Wayne Goddard. *Introducing the Theory of Computation*

Michael Sipser. *Introduction to the Theory of Computation, Third Edition*. Cengage Learning, 2013.

Administrivia – contd.

Grading:

4 Midterms : 10% + 10% + 10% + 10% = 40% (in class)

Final: 40% You must pass the final to pass the course

Assignments (4 or 5 sets): 20%

Notes:

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

Administrivia – contd.

Plagiarism: Will be dealt with very strictly. Read the detailed policies 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: I will post links.

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.

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
- Near the end of the course we will consider efficiency

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

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

Ch. 0:Set notation and languages

Topics

- Sets and sequences
- Tuples
- Functions and relations
- Graphs
- Boolean logic: $\vee \ \wedge \ \neg \ \Leftrightarrow \ \Rightarrow$

- **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) and Suprakesh Datta

Topics you should know:

- Elementary set theory
- Elementary logic
- Functions
- Graphs

Set Theory review

- Definition
- Notation: $A = \{ x \mid x \in \mathbb{N} , x \bmod 3 = 1\}$

  $\mathbf{N} = \{1,2,3,\ldots\}$
- 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

$L_{<6} = \{ x \mid x \in \mathbb{N} , x<6 \}$

$L_{prime} = \{x \mid x \in \mathbb{N}, x \text{ is prime}\}$

$L_{<6} \cap L_{prime} = \{2,3,5\}$

$\Sigma = \{0,1\}$

$\Sigma\times\Sigma = \{(0,0), (0,1), (1,0), (1,1)\}$

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

Power set

"Set of all subsets"

Formal: $\mathcal{P}(A) = \{ S \mid S \subseteq A \}$

Example: $A = \{x,y\}$

$\mathcal{P}(A) = \{ \{\} , \{x\} , \{y\} , \{x,y\} \}$

Note the different sizes: for finite sets

$$|\mathcal{P}(A)| = 2^{|A|}$$

$$|A \times A| = |A|^2$$

Graphs: review

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

Logic: review

Boolean logic: $\vee \ \wedge \ \neg$

Quantifiers: $\forall$, $\exists$

<u>statement</u>: Suppose $x \in \mathbb{N}, y \in \mathbb{N}$.

Then $\forall x \ \boxed{\exists y \ y > x}$

for any integer, there exists a larger integer

$\Rightarrow$: $a \Rightarrow b$ "is the same as" (is logically equivalent to) $\neg a \vee b$

$\Leftrightarrow$: $a \Leftrightarrow b$ is logically equivalent to

$(a \Rightarrow b) \wedge (b \Rightarrow a)$

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 $P(y,x)$: $x \le y$.

$\forall x\ \exists y\ x \le y$ is TRUE over **N** (set $y = x + 1$)

$\exists y\ \forall x\ x \le y$ is FALSE over **N** (there is no largest number in **N)**

Functions: review

- f: $A \rightarrow C$
- f: $A \times B \rightarrow C$

Examples:

- f: $\mathbf{N} \rightarrow \mathbf{N}$
- f: $\mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$
- f: $A \times B \rightarrow A$, $A = \{a,b\}$, $B = \{0,1\}$

Functions: review

- f: $A \rightarrow C$
- f: $A \times B \rightarrow C$

Examples:

- f: $\mathbf{N} \rightarrow \mathbf{N}$ $f(x) = 2x$
- f: $\mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$
- f: $A \times B \rightarrow A$, $A = \{a,b\}$, $B = \{0,1\}$

Functions: review

- $f: A \to C$
- $f: A \times B \to C$

Examples:

- $f: \mathbf{N} \to \mathbf{N}$ $f(x) = 2x$
- $f: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ $f(x,y) = x + y$
- $f: A \times B \to A$, $A = \{a,b\}$, $B = \{0,1\}$

Functions: review

- f: $A \to C$
- f: $A \times B \to C$

Examples:

- f: $\mathbf{N} \to \mathbf{N}$ $f(x) = 2x$
- f: $\mathbf{N} \times \mathbf{N} \to \mathbf{N}$ $f(x,y) = x + y$
- f: $A \times B \to A$, $A = \{a,b\}$, $B = \{0,1\}$

	0	1
a	a	b
b	b	a

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:

$$\{(a,0,a),(a,1,b),(b,0,b),(b,1,a)\}$$

	0	1
a	a	b
b	b	a

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 Σ
- Is a finite sequence of symbols from Σ
- Length of string w – denoted $|w|$ – length of sequence
- ε – 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 x x x x …x(k times)
- w^R - reversed string; e.g. if w = abcd then w^R = dcba.
- Lexicographic ordering : definition

Languages

- A language over Σ is a set of strings over Σ
- Σ^* is the set of all strings over Σ
- A language L over Σ is a subset of Σ^* ($L \subseteq \Sigma^*$)
- Typical examples:

  - $\Sigma=\{0,1\}$, the possible words over Σ 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:

$A \bullet B = \{ xy \mid x \in A \text{ and } y \in B \}$

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

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

$A \bullet A = \{ 00, 000, 0000 \}$ with $|A \bullet A| = 3$,

$A \times A = \{ (0,0), (0,00), (00,0), (00,00) \}$ with $|A \times A| = 4$

Recursively defined objects

Close relationship to induction

Example: set of all palindromes P

- $\varepsilon \in P$; • $\forall a \in \Sigma$, $a \in P$;
- $\forall a \in \Sigma$, $\forall x \in P$, $axa \in P$
- No other strings are in P

More definitions

Definition of Σ^*:

- $\varepsilon \in \Sigma^*$;
- $\forall a \in \Sigma$, • $\forall x \in \Sigma^*$, $xa \in \Sigma^*$;
- No other strings are in Σ^*.

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 L
- Prove that this is the language of strings

with more a's than b's.

Closure

If we add two natural numbers, do we get another natural number?

Yes! So we say $\mathcal{N}$ is *closed under addition.*

Are they closed under subtraction?

No. The smallest set that contains N and is closed under subtraction is …

$\mathcal{Z}$**.** (The integers.) Hence we say that the *closure of $\mathcal{N}$ under subtraction is $\mathcal{Z}$.*

Closure

Defn: Let *S* be a set, and let *f* be an n-ary function on *S*. Then the closure of *S* under *f* is the smallest set that contains *S* and is closed with respect to *f*.

What is the closure of $\mathcal{N}$ under division?

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

$\boxed{\text{Formal language theory}}$

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.

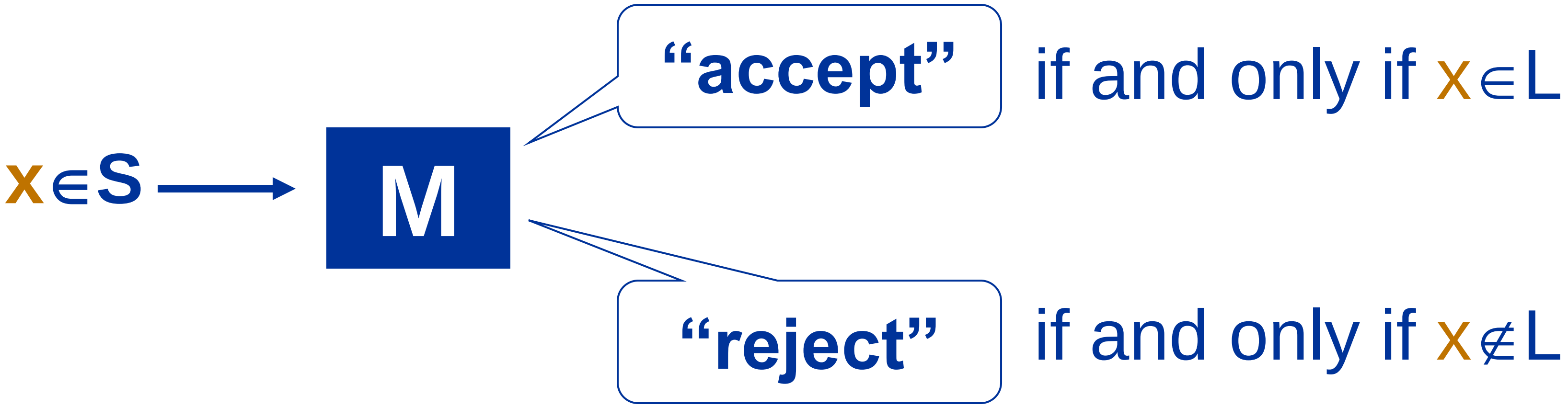
Recognizing languages

- Automata/Machines **accept** languages.

-- i.e., a given automaton accepts or "recognizes" exactly one language.

Recognizing Languages - 2

- Let L be a language $\subseteq$ S
- a machine M <u>recognizes</u> L if



Recognizing languages - 3

- Minimal spanning tree problem solver

Recognizing languages - 4

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

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

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

Proof 1: Suppose $\sqrt{(pq)} \neq (p+q)/2$

$\rightarrow (p+q)^2 \neq 4pq$

$\rightarrow p^2+q^2 +2pq \neq 4pq$

$\rightarrow p^2+q^2 -2pq \neq 0$, or

$\rightarrow (p-q)^2 \neq 0$, or

$\rightarrow p-q \neq 0$, or $p \neq q$.

Proof 2: prove the contrapositive!

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

Proof 2: Suppose $p=q$.

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

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 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 at least one box contains $\lceil n/k \rceil$ objects

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 $b^c = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = (\sqrt{2})^{\sqrt{2}*\sqrt{2}} = (\sqrt{2})^2 = 2$

Q: Do we know if $\sqrt{2}^{\sqrt{2}}$ is rational ?

Exercise: 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.

- Powerful proof technique
- When does it work?
- Why is it useful?

Proof by induction

Let P(n) denote the property that

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

Base case: P(1) is true, because f(1)=0.

Consider some fixed value of n that is ≥ 1

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.

Exercises

- Easy problem: Prove that 21 divides $4^{n+1} - 5^{2n-1}$ whenever n is a positive integer.
- (Q 19, pg 330, Rosen) Let P(n) be the statement that
  $1 + 1/4 + 1/9 + \ldots + 1/n^2 < 2 - 1/n$
- (Q 61, pg 332, Rosen) Show that n lines separate the plane into $(n^2 + n+2)/2$ regions if no two of these lines are parallel and no three of these lines intersect at a point.

Next: Finite automata

Ch. 1: Deterministic finite automata (DFA)

Look ahead:

We will study languages recognized by finite automata.

Summary of last lecture

- Some facts on logic
- Representing functions as input/output lists/tables, decision problems and languages
- Computing devices solving problems by accepting (recognizing) languages

Comparison of computation models

- We would like to claim “Model A can compute function every function B can, and further it can compute a function f, but model B cannot’’ which would imply model A is strictly more powerful than B.
- What we will (typically) prove is “Model A accepts every language model B accepts and in addition a language L that model B cannot accept”