Introduction

204213 Theory of Computation

Jittat Fakcharoenphol

Kasetsart University

October 29, 2008

Outline

Why? A theory course?

- What is a theory course?

Why? A theory course?

- What is a theory course?
    - Okay, I'll tell you later.

Why? A theory course?

- What is a theory course?
  - Okay, I'll tell you later.
- But why I should be interested in this course? (you ask)

Why? A theory course?

- What is a theory course?
    - Okay, I'll tell you later.
- But why I should be interested in this course? (you ask)
    - Let's see... umm...

Elctronics devices

- People counters
- Expressway gates

[1] source: from amazon product page.

Elctronics devices

- People counters
- Expressway gates
- Floor cleaner robots

[1]

[1] source: from amazon product page.

Regular expressions

```
html = "This is a simple html with <title>Ruby Regex</title> Handling."
/<title>(.*?)<\/title>/.match(html);
print $1,"\n"; ## Print the first match from html string
```

[2]

[2] Taken from http://icfun.blogspot.com/2008/04/ruby-regular-expression-handling.html.

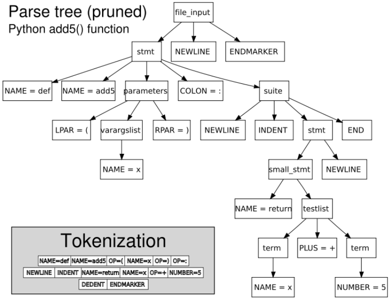
Programming languages

```
def add5(x):
   return x+5

def dotwrite(ast):
   nodename = getNodename()
   label=symbol.sym_name.get(int(ast[0]),ast[0])
   print ' %s [label="%s' % (nodename, label),
   if isinstance(ast[1], str):
      if ast[1].strip():
         print '= %s"];' % ast[1]
      else:
         print '"]'
   else:
      print '"];'
      children = []
      for n, child in enumerate(ast[1:]):
         children.append(dotwrite(child))
      print ' %s -> {' % nodename,
      for name in children:
         print '%s' % name,
```

[3] source: wikipedia, article "Programming languages"

Programming languages

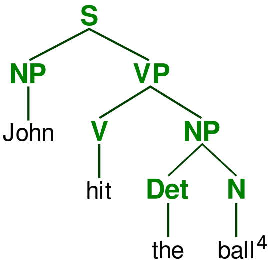
```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print ' %s [label="%s' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print ' %s -> {' % nodename,
        for name in children:
            print '%s' % name,
```



[3]

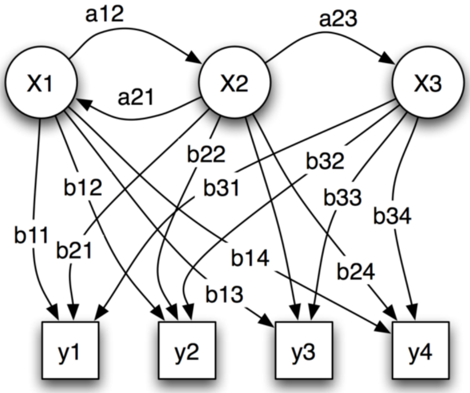
[3] source: wikipedia, article “Programming languages”

Natural languages



[4] source: wikipedia article "Parse tree".

Machine learning



[5]

[5] source: wikipedia: `Image:HiddenMarkovModel.png`.

Main question

What are the fundamental capabilities and limitations of computers?

Main question

What are the fundamental capabilities and limitations of computers?

How are we going to study that BIG question?

This huge bridge


[6]

How did it get designed?

[6] source: wikipedia, article "Golden Gate Bridge"; idea taken from S.A. Seshia's slides

From this simpler model!


7

[7] source: taken from S.A. Seshia's slides

Our turn

- So, instead of this

[8]

- We'll study something much, much simpler.

[8] source: wikipedia, article "Computer".

A theory course

This is a theory course; that means...

A theory course

This is a theory course; that means...

- We'll study something from the mathematical point of view.

A theory course

This is a theory course; that means...

- We'll study something from the mathematical point of view.
- We'll be interested in asserting properties that are definitely true (under a clearly-stated assumption).

A theory course

This is a theory course; that means...

- We'll study something from the mathematical point of view.
- We'll be interested in asserting properties that are definitely true (under a clearly-stated assumption).
- And, we'll **prove** lots of theorems.

Three major topics

- Complexity theory
- Computability theory
- Automata theory

Complexity theory

- What makes some problems computationally hard and others easy?
- **Goal:** Distinguishing between hard problems (but maybe solvable) and easy problems

Computability theory

- What can computers do?

Computability theory

- What can computers do?
  - A lot.

Computability theory

- What can computers do?
  - A lot.
- But there are basic problems that cannot be solved by computers.
  - Oh...

Computability theory

- What can computers do?
  - A lot.
- But there are basic problems that cannot be solved by computers.
  - Oh...
- **Goal:** Distinguishing between problems that can be solved by computers and those that cannot be solved.

Automata theory

[9]

[9] source: wikipedia Image:TeaAutomatAndMechanism.jpg.

Automata theory

- Studies **definitions and properties** of mathematical models of computation.
- Basic models:
  - Finite automata — used in text processing, compilers, hardware design
  - Context-free grammar — used in compilers, natural language processing.

Course information

- Homepage: `http://www.cpe.ku.ac.th/ jtf/204213`
- Grading: 25% midterm1, 25% midterm2, 35% final, 15% homework

Notes on the course slides

You've seen that the slides are very sketchy and extremely incomplete. It only provides a guideline for me to proceed, and a rough idea on what's going on in the class for you.

They are not a **substitute** for class attendance.

Mathematical background

Since this is a theory course, everything we conclude will be precise. Every statement we accept must be true, i.e., the argument supporting it must be solid—beyond **any** doubt.

Mathematical background

Since this is a theory course, everything we conclude will be precise. Every statement we accept must be true, i.e., the argument supporting it must be solid—beyond **any** doubt.

- Okay, we'll **prove** lots of theorems.

Basic notions and terminology

- Sets
- Sequences and tuples
- Functions and relations
- Graphs
- Strings and languages
- Boolean logic

Sets (1)

- elements, members

Sets (1)

- elements, members
- subset: $A \subseteq B$ iff for each element $x \in A$, $x \in B$.

Sets (1)

- elements, members
- subset: $A \subseteq B$ iff for each element $x \in A$, $x \in B$.
- proper subset: $A \subsetneq B$ if $A \subseteq B$ and $A \neq B$

Sets (1)

- elements, members
- subset: $A \subseteq B$ iff for each element $x \in A$, $x \in B$.
- proper subset: $A \subsetneq B$ if $A \subseteq B$ and $A \neq B$
- multiset

Sets (1)

- elements, members
- subset: $A \subseteq B$ iff for each element $x \in A$, $x \in B$.
- proper subset: $A \subsetneq B$ if $A \subseteq B$ and $A \neq B$
- multiset
- empty set ($\emptyset$)

Sets (2)

- infinite sets: natural numbers ($\mathcal{N}$), integers ($\mathcal{Z}$), reals ($\mathcal{R}$)

Sets (2)

- infinite sets: natural numbers ($\mathcal{N}$), integers ($\mathcal{Z}$), reals ($\mathcal{R}$)
- set operations: union, intersection, complement

Sets (2)

- infinite sets: natural numbers ($\mathcal{N}$), integers ($\mathcal{Z}$), reals ($\mathcal{R}$)
- set operations: union, intersection, complement
- Venn diagram

Sequences and tuples

- A sequence is an ordered list of objects.
    - $1, 2, 3, 4, 5, \ldots$
    - $3, 4, 2, 6$
    - Sometimes we put them in parentheses, e.g., $(3, 4, 2, 6)$.

Sequences and tuples

- A sequence is an ordered list of objects.
  - $1, 2, 3, 4, 5, \ldots$
  - $3, 4, 2, 6$
  - Sometimes we put them in parentheses, e.g., $(3, 4, 2, 6)$.
- A tuple is a finite sequence. A k-tuple is a sequence of k elements. A 2-tuple is called a pair.

Sequences and tuples

- A sequence is an ordered list of objects.
  - $1, 2, 3, 4, 5, \ldots$
  - $3, 4, 2, 6$
  - Sometimes we put them in parentheses, e.g., $(3, 4, 2, 6)$.
- A tuple is a finite sequence. A k-tuple is a sequence of k elements. A 2-tuple is called a pair.
- A power set of set A is a set of all subsets of A.
  - $\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ is a power set of $\{a, b\}$.

Sequences and tuples

- A sequence is an ordered list of objects.
  - $1, 2, 3, 4, 5, \ldots$
  - $3, 4, 2, 6$
  - Sometimes we put them in parentheses, e.g., $(3, 4, 2, 6)$.
- A tuple is a finite sequence. A k-tuple is a sequence of k elements. A 2-tuple is called a pair.
- A power set of set A is a set of all subsets of A.
  - $\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ is a power set of $\{a, b\}$.
- A Cartesian product of two subsets A and B is a set of all pairs whose first element is a member of A and second element is a member of B.

Functions and relations

- function, mapping
- domain, range
- function arguments
- k-ary functions, binary functions, unary functions
- predicate
- relations
- equivalence relations

Strings and languages (1)

- Strings are basic objects of our study.
- Many "kinds" of strings:
    - DNA sequence: `CGTAGACGATAGACCGGAAG`
    - English sentence: "Hello, I am a student."
    - Binary string: 1010111010010001110101 01

Strings and languages (2)

- When we want to talk about particular set of strings, we have to specify the set of possible symbols in the strings.

Strings and languages (2)

- When we want to talk about particular set of strings, we have to specify the set of possible symbols in the strings.
- An alphabet is a nonempty finite set. Each member of the alphabet is called the symbol of the alphabet.
  - DNA sequence: $\Sigma_1 = \{A, C, G, T\}$
  - English sentence: $\Sigma_2 = \{a, \ldots, z, A, \ldots, Z, space, comma\}$
  - Binary string: $\Sigma_3 = \{0, 1\}$

Strings and languages (2)

- When we want to talk about particular set of strings, we have to specify the set of possible symbols in the strings.
- An alphabet is a nonempty finite set. Each member of the alphabet is called the symbol of the alphabet.
  - DNA sequence: $\Sigma_1 = \{A, C, G, T\}$
  - English sentence: $\Sigma_2 = \{a, \ldots, z, A, \ldots, Z, space, comma\}$
  - Binary string: $\Sigma_3 = \{0, 1\}$
- A string over an alphabet is a finite sequence of symbols from the alphabet. (usually written with no commas).

Strings and languages (2)

- When we want to talk about particular set of strings, we have to specify the set of possible symbols in the strings.
- An alphabet is a nonempty finite set. Each member of the alphabet is called the symbol of the alphabet.
  - DNA sequence: $\Sigma_1 = \{A, C, G, T\}$
  - English sentence: $\Sigma_2 = \{a, \ldots, z, A, \ldots, Z, space, comma\}$
  - Binary string: $\Sigma_3 = \{0, 1\}$
- A string over an alphabet is a finite sequence of symbols from the alphabet. (usually written with no commas).
- The length of string w is the number of symbols in w. The length of w is denoted by $|w|$.

Strings and languages (2)

- When we want to talk about particular set of strings, we have to specify the set of possible symbols in the strings.
- An alphabet is a nonempty finite set. Each member of the alphabet is called the symbol of the alphabet.
  - DNA sequence: $\Sigma_1 = \{A, C, G, T\}$
  - English sentence: $\Sigma_2 = \{a, \ldots, z, A, \ldots, Z, space, comma\}$
  - Binary string: $\Sigma_3 = \{0, 1\}$
- A string over an alphabet is a finite sequence of symbols from the alphabet. (usually written with no commas).
- The length of string w is the number of symbols in w. The length of w is denoted by $|w|$.
- A string of length zero is called the empty string, denoted by ϵ.

Strings and languages (3)

- For a string w of length n, we write $w = w_1 w_2 \cdots w_n$ where each w_i is a symbol.

Strings and languages (3)

- For a string w of length n, we write $w = w_1 w_2 \cdots w_n$ where each w_i is a symbol.
- If $x = x_1 x_2 \cdots x_m$ and $y = y_1 y_2 \cdots y_m$, the concatenation of x and y, denoted by xy is the string $x_1 x_2 \cdots x_n y_1 y_2 \cdots y_m$.

Strings and languages (3)

- For a string w of length n, we write $w = w_1 w_2 \cdots w_n$ where each w_i is a symbol.
- If $x = x_1 x_2 \cdots x_m$ and $y = y_1 y_2 \cdots y_m$, the concatenation of x and y, denoted by xy is the string $x_1 x_2 \cdots x_n y_1 y_2 \cdots y_m$.
- Also, x^k is the string obtained by concatenating x with itself for k times.
    - If x is abc, x^3 is abcabcabc

Strings and languages (3)

- For a string w of length n, we write $w = w_1 w_2 \cdots w_n$ where each w_i is a symbol.
- If $x = x_1 x_2 \cdots x_m$ and $y = y_1 y_2 \cdots y_m$, the concatenation of x and y, denoted by xy is the string $x_1 x_2 \cdots x_n y_1 y_2 \cdots y_m$.
- Also, x^k is the string obtained by concatinating x with itself for k times.
  - If x is abc, x^3 is abcabcabc
- A **language** is a set of strings.

Boolean logic

- boolean operations: negation (NOT), conjuction (AND), and disjunction (OR)
- propositions, predicates

Definitions, theorems, and proofs (1)

- **Definitions**
- **Mathematical statements**

Definitions, theorems, and proofs (2)

- **Proofs** are solid logical arguments. We need proofs beyond any doubt.
- **Theorems** are mathematical statements supported by proofs.
- **Lemmas** are "smaller" mathematical statements used to prove theorems. (But sometimes lemmas get more popular.)
- **Corollaries** are statements that follow easily from some theorem or lemma.

You should be familiar with these concepts from the discrete math class.

Finding proofs

- Read carefully.

Finding proofs

- Read carefully.
- Identify parts.

Finding proofs

- Read carefully.
- Identify parts.
  - $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (P \Leftarrow Q)$
  - For two sets A and B, $A = B \equiv (A \subseteq B) \wedge (B \subseteq A)$

Finding proofs

- Read carefully.
- Identify parts.
  - $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (P \Leftarrow Q)$
  - For two sets A and B, $A = B \equiv (A \subseteq B) \wedge (B \subseteq A)$
- Try with examples

Finding proofs

- Read carefully.
- Identify parts.
  - $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (P \Leftarrow Q)$
  - For two sets A and B, $A = B \equiv (A \subseteq B) \wedge (B \subseteq A)$
- Try with examples
  - Try to find counter examples.

Finding proofs: tips

- Be patient.
- Come back to it.
- Be neat.
- Be concise.

Types of proof

There are many. Here are a few of them...

- Proof by construction
- Proof by contradiction
- Proof by induction

Let's review what they are and see some examples.

Proof by construction

You want to know if something exists?

Proof by construction

You want to know if something exists?
Okay, I'll construct it for you.

Proof by contradiction

You want to know if something is true?

Proof by contradiction

You want to know if something is true?
Okay, let's see what happens if it is not true.

Proof by contradiction

You want to know if something is true?
Okay, let's see what happens if it is not true.

- If that leads to impossibility, you should then believe me that it is true.

Proof by induction (1)

This one is hard...

Proof by induction (1)

This one is hard...
Examples might help.

Proof by induction (2)

- Want to prove that a statement $P(i)$ is true for every $i \in \mathcal{N}$.

Proof by induction (2)

- Want to prove that a statement $P(i)$ is true for every $i \in \mathcal{N}$.
- There are two steps: basis and induction step.
  - **Basis** proves that $P(1)$ is true.
  - **Induction step** proves that for each $i \geq 1$, if $P(i)$ is true, then $P(i+1)$ is true.

Proof by induction (2)

- Want to prove that a statement $P(i)$ is true for every $i \in \mathcal{N}$.
- There are two steps: basis and induction step.
  - **Basis** proves that $P(1)$ is true.
  - **Induction step** proves that for each $i \geq 1$, if $P(i)$ is true, then $P(i+1)$ is true.
- When proving the induction step, the assumption that $P(i)$ is true is called induction hypothesis.

Practice 1

There are 10 students in a class. The average score of one exam is 10, and none of the students gets less than 0 in this exam. Prove that the number of students who get the scores of at least 20 from this exam is at most 5.

Practice 2

Prove that for any natural number $n \geq 1$,

$$1 + 2 + \cdots + n = \frac{(n)(n+1)}{2}.$$

Practice 3

Prove that

$$\sum_{i=1}^{n} i \cdot 2^i = (n-1) \cdot 2^{n+1} + 2.$$

Practice 4

Suppose that we draw n lines on the plane in such a way that no two are parallel and no three intersect in a common point. Prove that the plane is divided into exactly $n(n+1)/2+1$ parts by the lines.