CSCI 2670, Fall 2012

Introduction to Theory of Computing

Department of Computer Science
University of Georgia
Athens, GA 30602

Instructor: Liming Cai
www.cs.uga.edu/∼cai

Lecture Note 1
Introduction and Review

A Tentative Schedule

I. Review (less than 1 week)

chapter 0: set, function, relation, string, language, logic, theorem, proof

II. Automata and Languages (7-8 weeks)

chapter 1: regular language, finite automata, nondeterminism, regular expression, non-regular language (3-4 weeks)

The first exam

chapter 2. context-free grammar, context-free language, push-down automata, non-context-free language (3-4 weeks)

The second exam

III. Computability Theory (3 weeks)

chapter 3: Turing machine, Chomsky hierarchy,

chapter 4: decidable language, Halting problem,

chapter 5: undecidable language, reduction

IV. Complexity Theory (2 weeks)

chapter 7: time complexity, P, NP-completness,

chapter 8: space complexity

The final exam

Motivations

Theory of computing:

1. foundation of computer science,

   the theory existed before the first computer model

2. theory and techniques for core CS subdisciplines

   programming language and compiler design, text processing, algorithm design, complexity theory, parallel computing, etc.

3. elegant, simple way to think about computation

   fundamental issues remain regardless advanced technologies

4. new applications

   bio-medical sciences

   non-traditional computation models

What is this course about? goals?

Summarized as the Chomsky Hierarchy and extension

Model	Languages/Grammar	What are they?
finite automata	regular	constant memory
push-down FA	context-free	an additional stack
linear-bounded TMs	context-sensitive	linear memory
Turing machines	decidable	unlimited memory
unknown	undecidable	unknown
Polynomial-time TMs	class P	tractable problems
Polynomial-time NTMs	class NP	intractable
polynomial-space TMs	class PSPACE	P=?NP =?PSPACE

Some explanations for Chomsky Hierarchy

How powerful (powerless) are programs with a limited memory?

e.g.,

```
Program Foo;
Int X, Y, Z, W;
```

what can it do? how can those integers be?

can examine the input but does not memorize much

what can it not do?

can it count or does matching?

cannot recognize even parenthesization!

like $(x + 20) \times (((y - z) \times (w + u) - 40) \times v)$ or simply

`( ) ( ( ( ) ( ) ) )`

The case is “context-free”.

Your program may be able to count the number of ’(’s and ’)’s.

But what if only a limited number **bits** are used?

A stack can help

How?

Remember for now:

stack

= tree structure

= recursion

= “context-free”

= nested + parallel relationships

Can a single stack be powerful enough to recognize "crossing relationships"

`( [ ) ( ] ( [ ] ) )` ?

(*Note: you are only allowed to read the string ONCE*)

A single stack is not that powerful enough for "context-sensitive" thing.

BTW, where did you see this before?

Two stacks would work. But how to recognize

`( [ ) ( { ] ( [ } ] { ) } )` ?

Two stacks can do "anything".

I. Review

Chapter 0. Introduction

sets, basic operations, properties

relations, functions, predicates

strings, languages,

Boolean logic, theorem, proofs

Set: a collection of (related, discrete) objects

elements of a set: $x \in S$
empty set: ϕ
cardinality of a set: $|S|$, infinite set
subset: $A \subseteq B$, and superset, proper subset $A \subset B$
complement of a set: $\bar{S}$
union of two sets: $A \cup B$, intersection of two sets: $A \cap B$
Cartesian product (cross product): $A \times B = \{(a, b) : a \in A, b \in B\}$
Power set: $2^A = \{B : B \subseteq A\}$,

how many elements in 2^A?

Relation and Function: subsets of Cartesian production of two sets

many-many, many-1, 1-many, and 1-1 relations
function $f : A \to B$, a many-1 relation R_f, not 1-many.

$$f(x) = y \text{ if and only } (x, y) \in R_f$$

domain: A, range: B

1-1 function (injection): an 1-1 relation
onto function (surjection) f:
for every $y \in B$, there is an $x \in A$, $(x, y) \in R_f$

bijection: a both 1-1 and onto function.

k-ary relation: a subset of $A \times A \times \ldots \times A$ (k times)

predicate: range is $\{TRUE, FALSE\}$

equivalence relation: a binary relation R satisfying

(1) reflexive: $(x, x) \in R$
(2) symmetric: if $(x, y) \in R$ then $(y, x) \in R$
(3) transitive: if $(x, y) \in R$ and $(y, z) \in R$, then $(x, z) \in R$

Graph: defined by a pair of sets (V, E), in which $E \subseteq V \times V$.

vertex $v \in V$, edge $(u, v) \in E$
directed edge if $(u, v) \neq (v, u)$
subgraph: $H = (U, F)$ of G, if $U \subseteq V$ and $F \subseteq E \cap (U \times U)$. path: a sequence of vertices in V simple path: the vertices do not repeat
cycle: path in which the start and end are the same
tree: graph without cycles
connected graph: a path between every two vertices

String and Language

alphabet: Σ, a finite set of symbols

string: s, a finite sequence of symbols taken from an alphabet

empty string: ϵ

$\Sigma^0 = \{\epsilon\}$, $\Sigma^k = \{xy : x \in \Sigma^{k-1}, y \in \Sigma\}$, for k=1, 2, ...

transitive closure of Σ: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

string $s \in \Sigma^*$

reverse of string w: $w^{\mathcal{R}}$

string length: the number of symbols in a string

concatenation of strings $x = x_1 \ldots x_m$ and $y = y_1 \ldots y_n$:

$xy = x_1 \ldots x_m y_1 \ldots y_n$

lexicographical order of strings: dictionary order

language L: a set of strings, $L \subseteq \Sigma^*$

Boolean Logic:

boolean values: 0, 1

boolean operands: $\wedge, \vee, \neg$

boolean variables: x, y, z

boolean expressions: P, Q, R, formed by
boolean values, operands, variables, and expressions.

Definition, theorem, and proof:

definition: describing objects precisely
mathematical statement: stating objects that have certain property.
proof: a convincing logical argument that a statement is true
theorem: a mathematical statement proved true
lemma: a theorem assisting the proof of an more significant theorem
corollary: conclusion easily derived from a theorem

Constructing Proofs:

not alway easy

be patient
be logical
be neat/concise

type of proofs:
- proof by construction
- proof by contradiction
- proof by induction

proof by construction

Some theorems claim that some type of object (or property) exists.

Example: For every even number $n > 2$, there is a 3-regular graph[a]

Proof: Construct such a graph for every given $n > 2$ even.

Construct a "cycle" using all n vertices, and create edges $(i, i + n/2)$ for all $i = 1, 2, \ldots, n/2 - 1$.

[a] A k-regular graph is a graph in which every vertex has degree k.

proof by contradiction

Example 1. "Pigeonhole principle": Putting n pigeons in k holes, $k < n$, there is at least one hole hosting more than one pigeon.

Proof:

Assume otherwise.

Then the total number of pigeons is $\leq k < n$. Contradicts.

Example 2: $\sqrt{2}$ is irrational[a]

<u>Proof</u>: Assume otherwise, i.e., $\sqrt{2} = m/n$ for some m and n

(**at least one of m and n is odd!**).

Then $n\sqrt{2} = m$.

$2n^2 = m^2$; m^2 is even. And m is even too.

(**as the square of an odd number is always odd!**)

So m can be written as $m = 2k$ for some integer k. So we have

$$2n^2 = 4k^2 \Longrightarrow n^2 = 2k^2$$

So n^2 is even; so n is also even.

contradicts with that at least one of m and n is odd!

[a] A number is rational if it can be expressed as m/n for two integers m and n.

proof by induction

To show certain property $\mathcal{P}$ holds for every integer $n = 1, 2, \ldots,$

It suffices to show

(1) $\mathcal{P}$ holds for $n = 1$,

(2) $\mathcal{P}$ holds $k \longrightarrow \mathcal{P}$ holds for $k + 1$.

where (2) is "chain reaction" or "property propagation", while (1) is the "starting point".

Consider the following **theorem** (knocking dominos): *If the first domino was knocked down, then for every n, the nth domino will be down.*

Can you use proof by induction to prove?

Example: For all $n \geq 1$, summation $1 + 2 + \ldots + n = \frac{n}{2}(n+1)$

What is $\mathcal{P}$ here?

$\mathcal{P}(n) =$ "$1 + 2 + \ldots + n = \frac{n}{2}(n+1)$"

Proof. $n = 1$, $\mathcal{P}$ holds because $1 = \frac{1}{2}(1+1)$

Assume $\mathcal{P}(k)$ holds, i.e., $1 + 2 + \ldots + k = \frac{k}{2}(k+1)$

We now show $\mathcal{P}(k+1)$ holds as well:

$$1 + 2 + \ldots + k + k + 1 = (1 + 2 + \ldots + k) + (k+1)$$
$$= \frac{k}{2}(k+1) + (k+1) = (k+1)(k/2+1)$$
$$= (k+1)(k/2+2/2) = \frac{k+1}{2}((k+1)+1)$$