Reading Notes of
Elements of the Theory of Computation

Tianyi Cui

October 23, 2011

Introduction

Fundamental questions in computer science answered by theory of computation:

- What is an algorithm?

- What can and what cannot be computed?

- When should an algorithm be considered practically feasible.

The theory of computation is the mathematical abstractions of computers, but its origin is even before the advent of the electronic computer.

> It is based on very few and elementary concepts, and draws its power and depth from the careful, patient, extensive, layer-by-Iayer manipulation of these concepts – just like the computer.

1 Sets, Relations, and Languages

1.1 Sets

Power set: 2^A, the collection of all subsets of set A.

Partition of set A, subset of 2^A whose elements are nonempty and disjoint when contain all elements of A.

1.2 Relations and functions

a and b are called the *components* of the ordered pair (a, b).

The *Cartesian product* of two sets.

Ordered tuples: ordered triples, quadruples, quintuples, sextuples...

n-ary relation: unary, binary, ternary...

The domain, image, *range*, of function; one-to-one + onto = bijection; inverse.

1.3 Special types of binary relations

The relation $R \in A \times A$ is called a *directed graph*.

Properties of relations: reflexive, symmetric, antisymmetric, transitive.

Equivalence relation: r, s, t. Partial order: r, a, t. Total order.

1.4 Finite and infinite sets

Call two sets *equinumerous* if there is a bijection between them.

Finite (equinumerous with $\{1, 2, \ldots n\}$, infinite, countably infinite (equinumerous with $\mathbf{N}$), countable, uncountable.

1.5 Three fundamental proof techniques

The Principle of Mathematical Induction: Let A be a set of natural numbers such that (1) $0 \in A$, and (2) for each natural number n'n', if $\{0, 1, \ldots, n\} \subseteq A$, then $n + 1 \in A$. Then $A = \mathbf{N}$.

The Pigeonhole Principle: if A and B are finite sets and $|A| > |B|$, then there is no one-to-one function from A to B.

The Diagonalization Principle: Let R be a binary relation on a set A, and let D, the diagonal set for R, be $\{a : a \in A \text{ and } (a, a) \notin R\}$. For each $a \in A$, let $R_a = \{b : b \in A \text{ and } (a, b) \in R\}$. Then D is distinct from each R_a. Lemma: the set $2^{\mathbf{N}}$ is uncountable.

1.6 Closures and algorithms

The *reflecxie transitive closure* of a directed graph.

The *rate of growth* of a function f on $\mathbf{N}$.

The proof of correctness of the Floyd algorithm: define *rank of a path* as the biggest index among its intermediate nodes, and prove that after the jth iteration, all path with rank less than or equal to j will be found.

Closure property: Let D be a set, let $n \geq 0$, and let $R \subseteq D^{n+1}$ be a $(n+1)$-ary relation on D. Then a subset B of D is said to be *closed under* R if $b_{n+1} \in B$ whenever $b_1, \ldots, b_n \in B$ and $(b_1, \ldots, b_n, b_{n+1}\} \in R$. Any property of the form "the set B is closed under relation $R_1, R_2, \ldots, R_m$" is called a *closure property* of B.

The minimal set B that contains A and has property P is unique if P is a closure property defined by relations on a set D while $A \subseteq D$. Then we call B the *closure* of A under the relation $R_1, \ldots, R_m$.

Inclusion property: unary closure (take $n = 0$ in definition).

Any closure property over a finite set can be computed in polynomial time (see ex1.6.9).

1.7 Alphabets and languages

Here is the *mathematics of strings of symbols.*

symbol: any object, but often only common characters are used.

alphabet: a finite set of symbols.

string: finite sequence of symbols from the alphabet, which has *length*, operation of *concatenation*($\circ$), *substring*, *prefix*, *suffix*, s^n, operation of *reversal*(s^R) defined

language: any set of strings over an alphabet Σ, that is, any subset of Σ^*. It might be able to be enumerated *lexicographically*. It has *complement* ($\overline{L}$), *concatenation of languages*, *Kleene star* (the set of all strings obtained by concatenating zero or more strings from it). We write L^+ for LL^*, which is the *closure* of L under the function of concatenation.

1.8 Finite representations of languages

This section discusses how to use *regular expressions* to represent languages.

A *regular expression* is the representation of language using empty set, characters in alphabet, concatentaion (symbol usually omitted), function of union (the *or* operator in regex), star, and parentheses. We can define the function $\mathcal{L}$ from regular expressions to lanuages, whose range is called the class of *regular languages.*

A *language recognition device* is an algorithm that is specifically designed to answer questions of the form "is string w a member of L?".

A *language generator* is the description of the way of generating members of a language.

The relation between the above two types of finite language specifications is another major subject of this book.

2 Finite Automata

2.1 Deterministic Finite Automata

DFA is computer with no memory; input a string, output indicate whether it's acceptable.

DFA definition: quituple $M = (K, \Sigma, \delta, s, F)$, where K is a finite set of *states*, Σ is an alphabet, $s \in K$ is the *initial state*, $F \subseteq K$ is the set of *final states*, and δ the *transition function*, is a function from $K \times \Sigma$ to K.

A *configuration of a DFA* is any elements of $K \times \Sigma^*$. For two configuration (q, w) and (q', w'), then $(q, w) \vdash_M (q', w')$ if and only $\vdash_M^*$ if $w = aw'$ for some symbol $a \in \Sigma$, and $\delta(q, a) = q'$. We say that (q, w) **yields** (q', w') **in one step**. Denote the reflexive transitive closure of $\vdash_M$ by $\vdash_M^*$, $(q, w) \vdash_M^* (q', w')$ is read, (q, w) **yields** (q', w').

A string $w \in \Sigma^*$ is said to be *accepted* by M if and only if there is a state $q \in F$ such that $(s, w) \vdash_M^* (q, e)$. *The language accepted* by M, $L(M)$ is the set of all strings accepted by M.