CS 4410

Automata, Computability, and Formal Language

Dr. Xuejun Liang

Chapter 1

Introduction to the Theory of Computation

1. Mathematical Preliminaries and Notation
    - Sets
    - Functions and Relations
    - Graphs and Trees
    - Proof Techniques
2. Three Basic Concepts
    - Languages
    - Grammars
    - Automata
3. Some Applications

Learning Objectives

At the conclusion of the chapter, the student will be able to:

- Define the three basic concepts in the theory of computation: automaton, formal language, and grammar.
- Solve exercises using mathematical techniques and notation learned in previous courses.
- Evaluate expressions involving operations on strings.
- Evaluate expressions involving operations on languages.
- Generate strings from simple grammars.
- Construct grammars to generate simple languages.
- Describe the essential components of an automaton.
- Design grammars to describe simple programming constructs.

Sets

Representations

$S=\{0, 1, 2\}$

$S=\{i : i>0, i \text{ is even}\}$

Empty set: $\varnothing$

Operations

Union ($\cup$) : $S_1 \cup S_2 = \{x : x \in S_1 \text{ or } x \in S_2\}$

Intersection ($\cap$) : $S_1 \cap S_2 = \{x : x \in S_1 \text{ and } x \in S_2\}$

Difference (-) : $S_1 - S_2 = \{x : x \in S_1 \text{ and } x \notin S_2\}$

Complement : $\overline{S} = \{x : x \in U \text{ and } x \notin S\}$

Subset: $S_1 \subseteq S_2$

Proper subset: $S_1 \subset S_2$

Power set: $P(S) = 2^S = \{A : A \subseteq S\}$

Cartesian product: $S_1 \times S_2 = \{(x, y): x \in S_1 \text{ and } y \in S_2\}$

Sets: Examples

Example 1.1: If S is the set {a, b, c}, then its powerset is

Example 1.2: Let S_1 = {2, 4} and S_2 = {2, 3, 5, 6}. Then

Functions

Function $f: X \rightarrow Y$, $y = f(x)$, $x \in X$

- In general, **domain** of *f* is a subset of X and **range** of *f* is a subset of Y.
- If the domain of *f* is all of X, we say that *f* is a **total function** on X; otherwise f is said to be a **partial function**.

Given two functions *f* and *g* defined on the positive integers,

if there is a positive constant *c* such that for all sufficiently large *n*,

$$f(n) \leq c|g(n)|,$$

f is said to have **order of at most *g***, denoted by $f(n) = O(g(n))$. If

$$|f(n)| \geq c|g(n)|,$$

f is said to have **order of at least *g***, denoted by $f(n) = \Omega(g(n))$. Finally, if there exist constants c_1 and c_2 such that

$$c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|,$$

f and *g* are said to have the **same order of magnitude**, denoted by $f(n) = \theta(g(n))$

Functions: Example

Example 1.3

$$f(n) = 2n^2 + 3n,$$
$$g(n) = n^3,$$
$$h(n) = 10n^2 + 100$$

$$\Longrightarrow$$

$$f(n) = O(g(n)),$$
$$g(n) = \Omega(h(n)),$$
$$f(n) = \Theta(h(n))$$

Relations

Relation $R \subseteq X \times Y$, $(x, y) \in R$ (or $x\ R\ y$)

A function is a particular relation

Equivalence relation $\equiv$ on X ($\equiv\ \subseteq X \times X$), if it satisfies three rules:

1. **Reflexive**: $x \equiv x$ for all x
2. **Symmetric**: $x \equiv y$ then $y \equiv x$
3. **Transitive**: $x \equiv y$ and $y \equiv z$ then $x \equiv z$.

When $\equiv$ is an equivalence relation on X, then equivalence classes for any x in X can be defined as below.

$$\bar{x} = \{y \in X : x \equiv y\}$$

Relations: Example

Example 1.4:

On the set of nonnegative integers, we can define a relation

$$x \equiv y$$

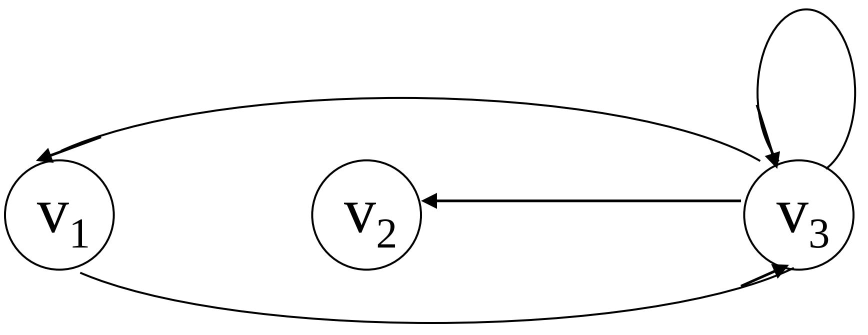
if and only if

$$x \bmod 3 = y \bmod 3$$

Then $\equiv$ is an equivalence relation.

Graphs

G = (V, E), where V = $\{v_1, v_2, \ldots, v_n\}$ and E = $\{e_1, e_2, .., e_m\}$



In **directed graph**

$e_i = (v_j, v_k)$

v_j is a parent of v_k

v_k is a child of v_j

In **undirected graph**

$e_i = \{v_j, v_k\}$

A **walk** from v_i to v_n: a sequence of edges $(v_i, v_j), (v_j, v_k), \ldots (v_m, v_n)$.

The **length** of a walk is the number of edges in the walk.

A **path** is a walk in which no edge is repeated.

A path is **simple** if no vertex is repeated.

A **cycle** with **base** v_i is a path from v_i to v_i.

A **loop** is an edge from a vertex to itself.

Trees

A **tree** is a directed graph that has no cycles, and has one distinct vertex, called the root, such that there is exactly one path from the root to every other vertices.

Leaf

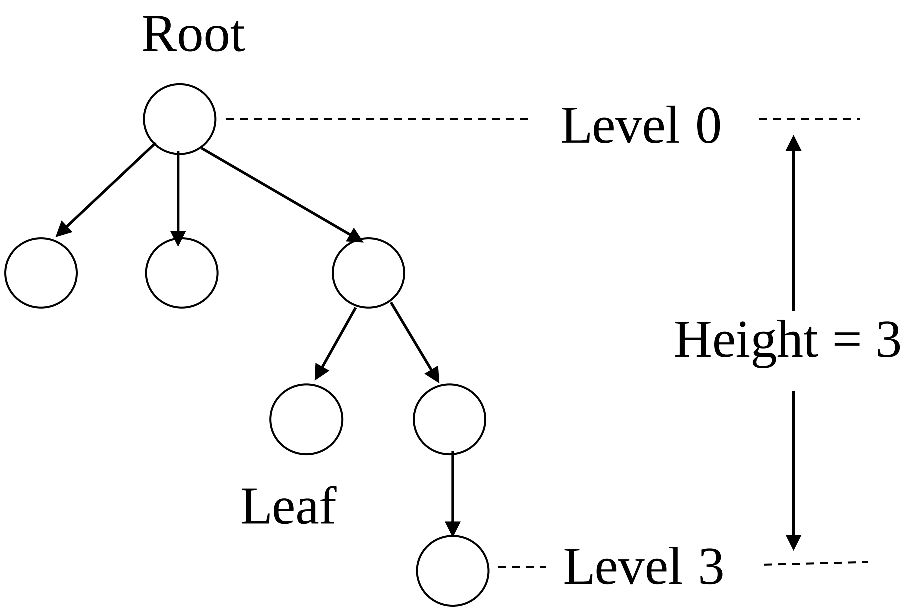
vertex without outgoing edges

Level of a vertex

The number of edges in the path from the root to the vertex

Height of a tree

The largest level number of any vertex



Proof Techniques

Proof by induction

In order to prove P(n) is true for all positive integer n, we need the following three steps of proof:

1. Basis: Verify P(1) is true
2. Induction hypothesis: Assume P(k) (or P(1), …, P(k)) is true
3. Induction proof: Prove P(k+1) is true

Example 1.5: Prove that a binary tree of height n has at most 2^n leaves.

Let L(n) denote the maximum number of leaves of a binary tree of height n, then we want to show that $L(n) \leq 2^n$.

Proof Techniques

Example 1.6: Show that $S_n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Proof Techniques

Proof by contradiction

Want to prove P is true.
Assume P is false, and leads to an incorrect conclusion.
So P cannot be false. That is, P is true.

Example 1.7: Show that $\sqrt{2}$ is an irrational number.