CS 4410

Automata, Computability, and Formal Language

Dr. Xuejun Liang

Spring 2019

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

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

Example 1.1 on p5 Example 1.2 on p5

Functions and Relations

Function f: X → Y, y = f(x), x∈X

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

if there is a positive constant c such that for all n, $\boldsymbol{f(n) \le cg(n)}$,
f is said to has **order of at most *g***, denoted by $\boldsymbol{f(n) = O(g(n))}$.

if $\boldsymbol{|f(n)| \ge c|g(n)|}$, f is said to has **order of at least *g***, denoted by $\boldsymbol{f(n) = \Omega(g(n))}$,

Finally, if there exist constants c_1 and c_2 such that
$\boldsymbol{c_1|g(n)| \le |f(n)| \le c_2|g(n)|}$, f and g are said to have the **same order of magnitude**, denoted by $\boldsymbol{f(n) = \theta(g(n))}$

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

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

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

Functions and Relations

Example 1.3 on p7

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

$$\Rightarrow$$

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

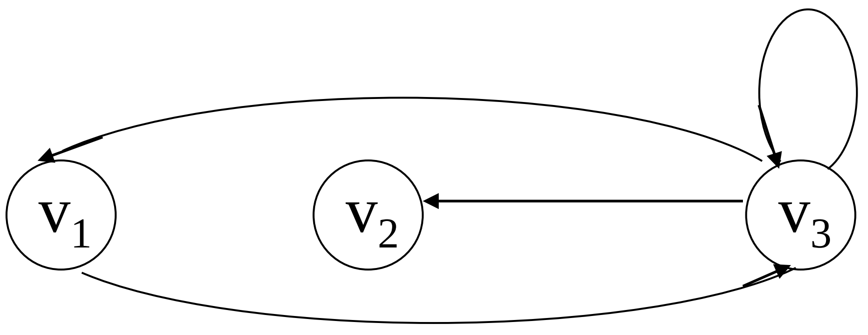
Example 1.4 on p8

x ≡ y if and only if x mod 3 = y mod 3

Then ≡ is an equivalence relation

Graphs and Trees

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.

Graphs and 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

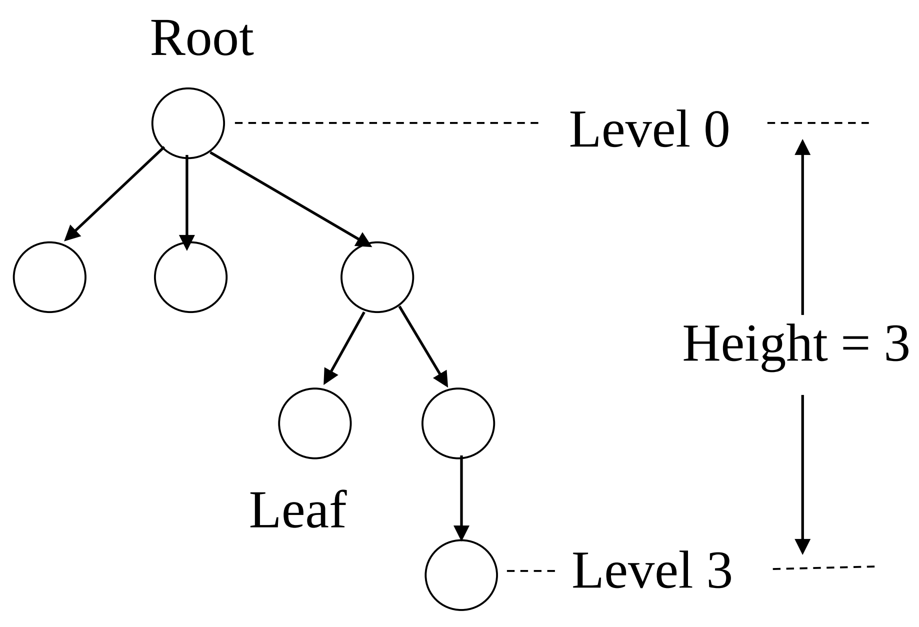
vertex without outgoing edges

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

Want to prove P(n) is true for all positive integer n

Three steps of proof:

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

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

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.

Theory of Computation
Basic Concepts

- *Automaton*: a formal construct that accepts input, produces output, may have some temporary storage, and can make decisions
- *Formal Language*: a set of sentences formed from a set of symbols according to formal rules
- *Grammar*: a set of rules for generating the sentences in a formal language

In addition, the theory of computation is concerned with questions of computability (the types of problems computers can solve in principle) and complexity (the types of problems that can solved in practice).

Languages

Alphabet: nonempty set Σ of symbols, E.g. $\Sigma=\{a, b\}$

Strings: finite sequence of symbols, E.g. w = abaaa, v = bbaab

Empty string: λ

Concatenation of two strings w and v: wv, $w^n = w \cdot w \cdots w,\ w^0 = \lambda$

Reverse of a string w: w^R

Length of a string w: |w|

$$|w| = \begin{cases} 0, & \text{if } w = \lambda \\ |u|+1, & \text{if } w = au,\ a \in \Sigma \end{cases}$$

Substring, Prefix, Suffix

$\Sigma^* = \{\text{all strings over } \Sigma\}$

$\Sigma^+ = \Sigma^* - \{\lambda\}$

A language: a subset L of Σ^*

A sentence of L: a string in L

Example 1.8: Prove $|uv| = |u| + |v|$

Example 1.9: Let $\Sigma=\{a,b\}$, then $\Sigma^*=\{\lambda, a, b, ab, ba, aab,\ldots\}$

Languages

Alphabet: nonempty set Σ of symbols, E.g. Σ={a, b}
Strings: finite sequence of symbols, E.g. w = abaaa, v = bbaab
Empty string: λ
Concatenation of two strings w and v: wv, $w^n = w \cdot w \cdots w, \;\; w^0 = \lambda$
Reverse of a string w: w^R
Length of a string w: |w|
$$| w | = \begin{cases} 0, & \text{if } w = \lambda \\ | u | + 1, & \text{if } w = au,\; a \in \Sigma \end{cases}$$
Substring, Prefix, Suffix
Σ* = {all strings over Σ}
$\Sigma^+ = \Sigma^* - \{\lambda\}$
A language: a subset L of Σ*
A sentence of L: a string in L

Example 1.8: Prove $|uv| = |u| + |v|$
Example 1.9: Let Σ={a,b}, then Σ*={λ, a, b, ab, ba, aab,…}

Languages

Complement $\overline{L} = \Sigma^* - L$

Reverse $L^R = \{w^R : w \in L\}$

Concatenation $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

$L^n = LL \cdots L$

$L^0 = \{\lambda\}$

Star-closure $L^* = L^0 \cup L^1 \cup L^2 \cdots$

Positive closure $L^+ = L^1 \cup L^2 \cdots$

Example 1.10 $L = \{a^n b^n : n \geq 0\}$

$L^2 = ?$

$L^R = ?$

Grammars

Definition 1.1 A grammar G is defined as a quadruple

$$G = (V, T, S, P), \text{ where}$$

V is a finite set of variables, T is a finite set of terminal symbols, $S \in V$ is the start variable, and P is a finite set of productions.

Production rule: $x \to y$, where $x \in (V \cup T)^+$ and $y \in (V \cup T)^*$

w derives z (z is derived from w)

- $w \Rightarrow z$, E.g. $w = uxv$ and $x \to y$ then $z = uyv$
- $w \stackrel{n}{\Longrightarrow} z$, $w = w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = z$
- $w \stackrel{*}{\Longrightarrow} z$, there is an $n \geq 0$ such that $w \stackrel{n}{\Longrightarrow} z$

Definition 1.2 Let G=(V, T, S, P) be a grammar. Then the set

$$L(G) = \{w \in T^*: S \stackrel{*}{\Longrightarrow} w\}$$

is the language generated by G. If $w \in L(G)$, then the sequence

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n \Rightarrow w$$

is a derivation of the sentence w. The strings S, w_1, w_2, …, w_n are called sentential forms of the derivation.

Examples

Example 1.11 $G = (\{S\}, \{a, b\}, S, P)$ with P given by

$$S \rightarrow aSb$$
$$S \rightarrow \lambda$$

Then $L(G) = \{a^n b^n : n \geq 0\}$

Example 1.12 Find a grammar that generates $L = \{a^n b^{n+1} : n \geq 0\}$

Solution: $G = (\{S, A\}, \{a, b\}, S, P)$

with products

$$S \rightarrow Ab$$
$$A \rightarrow aAb$$
$$A \rightarrow \lambda$$

Example 1.13 Let Σ={a, b}. The grammar G with productions

$$S \rightarrow SS,$$
$$S \rightarrow \lambda,$$
$$S \rightarrow aSb,$$
$$S \rightarrow bSa,$$

generates the language

L={w∈ Σ*: w contains equal numbers of a's and b's}

Examples

Two grammars G_1 and G_2 are equivalent if they generate the same languages, that is, $L(G_1)=L(G_2)$.

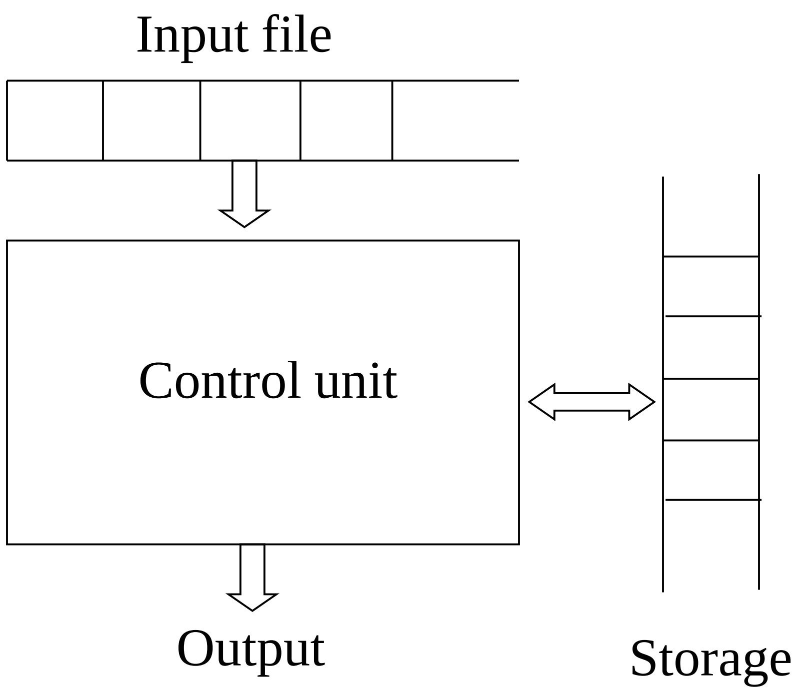
Example 1.14 $G_1 = (\{S\},\{a,b\},S,P_1)$ with P_1 given by

$$S \to aAb \mid \lambda$$
$$A \to aAb \mid \lambda$$

Then $L(G_1) = \{a^n b^n : n \geq 0\}$

So G_1 is equivalent to G in Example 1.11

Automata



Some terms

- Internal states
- Next-state or transition function
- Configuration
- Move

Deterministic automata
Nondeterministic automata

Accepter
Transducer

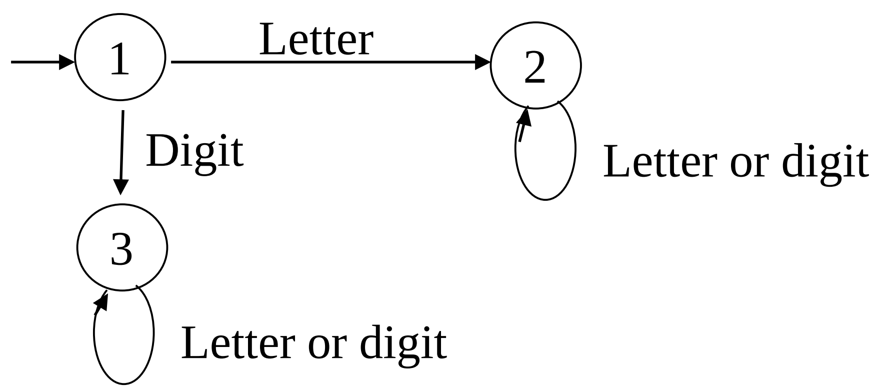
Some Applications

Compiler (parser) design and Digital circuit design

Example 1.15 Identifiers as a language generated by a grammar
(Identifiers: Strings of letters and digits starting with a letter)

<id> → *<letter> <rest>*
<rest> → *<letter> <rest>* | *<digit> <rest>* | λ
<letter> → *a* | *b* | …| *z*
<digit> → *0* | *1* | …|*9*

Example 1.16 Identifiers accepted by an automaton



Some Applications

Example 1.17 Serial binary adder

$$x = a_n a_{n-1} \ldots a_1 a_0 \text{ and } y = b_n b_{n-1} \ldots b_1 b_0$$
$$z = x + y = d_n d_{n-1} \ldots d_1 d_0$$