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.

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 (1/4)

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

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

Empty string: λ

Concatenation of two strings w and v: wv

$$w^n = w^{n-1} \cdot w,\ w^0 = \lambda$$

Reverse of a string w: w^R

Substring, Prefix, Suffix

Length of a string w: |w|

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

Languages (2/4)

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

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

Languages (3/4)

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

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

A language: a subset L of Σ^*

A sentence of L: a string in L

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

Languages (4/4)

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

$$L^2 = ?$$

$$L^R = ?$$

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: $x \rightarrow y$, 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 $x \rightarrow y$ then $z = uyv$
- $w \overset{n}{\Rightarrow} z$, $w = w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = z$
- $w \overset{*}{\Rightarrow} z$, there is an $n \geq 0$ such that $w \overset{n}{\Rightarrow} z$

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

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

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 $S, w_1, w_2, \ldots, w_n$ 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 $L(G) = \{a^n b^n : n \geq 0\}$

Examples

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 productions

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

Examples

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

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

generates the language

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

Equivalent Grammars

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

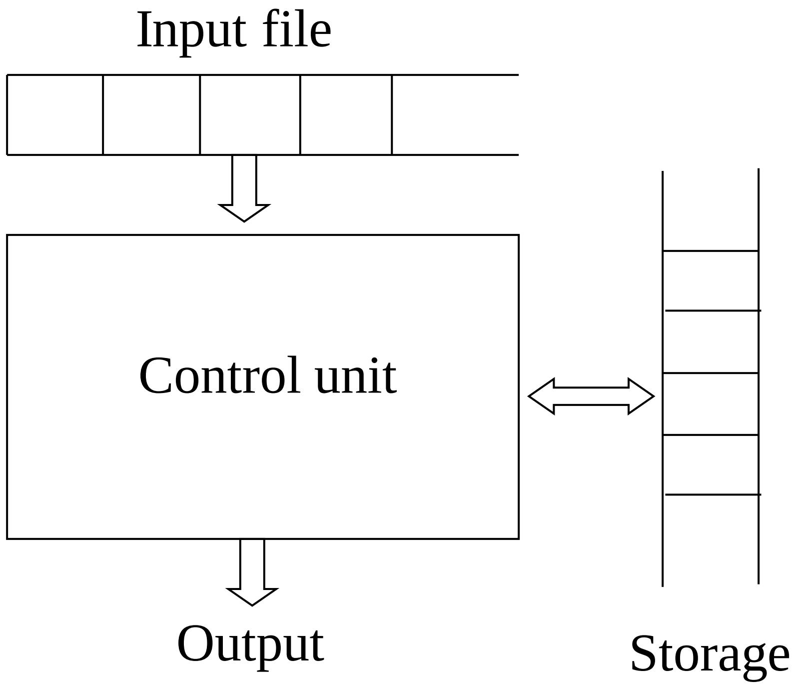
$$S \to aAb \mid \lambda$$
$$A \to aAb \mid \lambda$$

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

So G_1 is equivalent to G in Example 1.11

Automata (1/2)

- An <u>*automaton*</u> is an abstract model of a digital computer
- An automaton consists of
  - An input mechanism
  - A control unit
  - Possibly, a storage mechanism
  - Possibly, an output mechanism
- Control unit can be in any number of internal *states*, as determined by a *next-state* or *transition* function

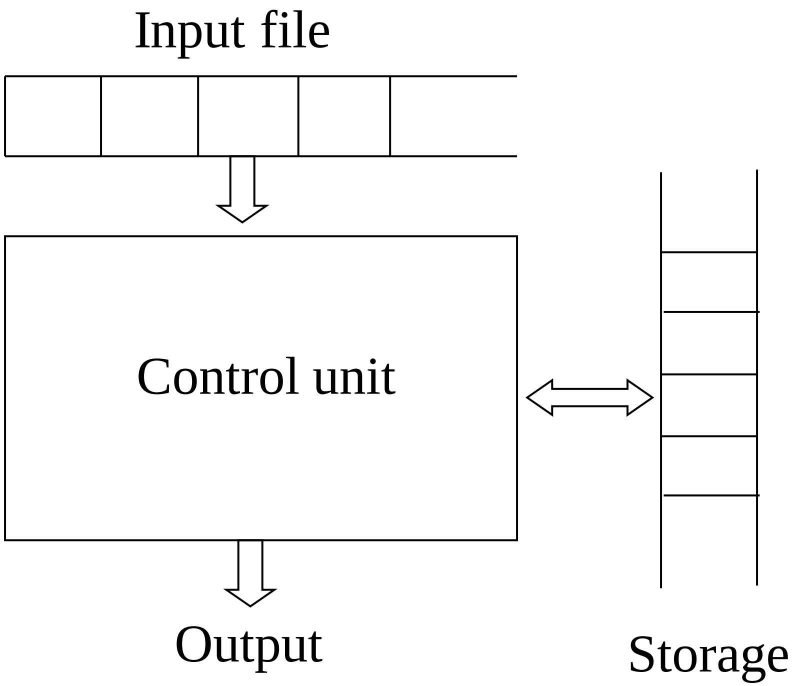


Automata (2/2)

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

- Accepter
- Transducer

- Deterministic automata
- Nondeterministic automata



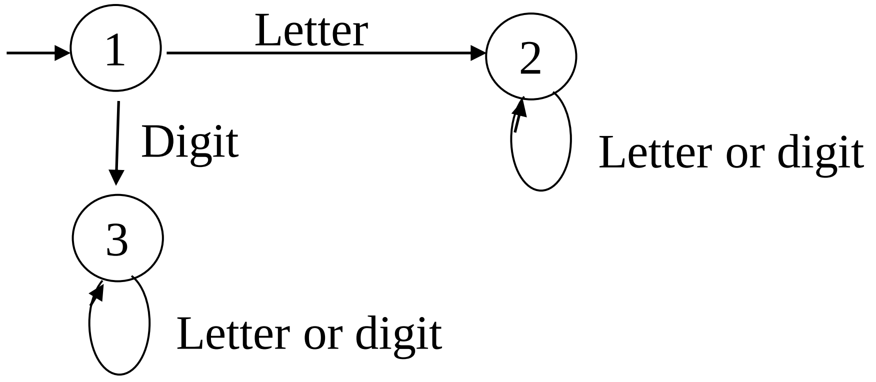
Some Applications (1/2)

Compiler (scanner and 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* | *A* | *B* | … | *Z*
<digit> → *0* | *1* | … | *9*

Example 1.16 Identifiers accepted by an automaton



Some Applications (2/2)

Example 1.17 Serial binary adder

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