Chap 1

Introduction to the theory of computation

1.2 Three Basic Concepts

1. Languages
2. Grammars
3. Automata

[1] Languages

① Σ : (alphabet)

finite, nonempty set of symbols

e.g. $\Sigma = \{a, b\}$

② w : (string over Σ)

finite sequences of symbols from Σ

e.g. $w = abaaa$

Usually we use $a, b, c, \cdots$ for elements of Σ

and $u, v, w, \cdots$ for string names

③ Σ^* : the set of all strings including empty string λ

Σ^+ : $\Sigma^* - \{\lambda\}$

The **concatenation** of two strings w and v is the string obtained by appending the symbols of v to the right end of w, that is, if

$$w = a_1 a_2 \cdots a_n$$

and

$$v = b_1 b_2 \cdots b_m,$$

then the concatenation of w and v, denoted by wv, is

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m.$$

The **reverse** of a string is obtained by writing the symbols in reverse order; if w is a string as shown above, then its reverse w^R is

$$w^R = a_n \cdots a_2 a_1.$$

The **length** of a string w, denoted by $|w|$, is the number of symbols in the string. We will frequently need to refer to the **empty string**, which is a string with no symbols at all. It will be denoted by λ. The following simple relations

$$|\lambda| = 0,$$

$$\lambda w = w\lambda = w$$

hold for all w.

Any string of consecutive symbols in some w is said to be a **substring** of w. If

$$w = vu,$$

then the substrings v and u are said to be a **prefix** and a **suffix** of w, respectively. For example, if $w = abbab$, then $\{\lambda, a, ab, abb, abba, abbab\}$ is the set of all prefixes of w, while bab, ab, b are some of its suffixes.

Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if u and v are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$|uv| = |u| + |v|. \tag{1.6}$$

If w is a string, then w^n stands for the string obtained by repeating w n times. As a special case, we define

$$w^0 = \lambda,$$

for all w.

④ L : (language over Σ)

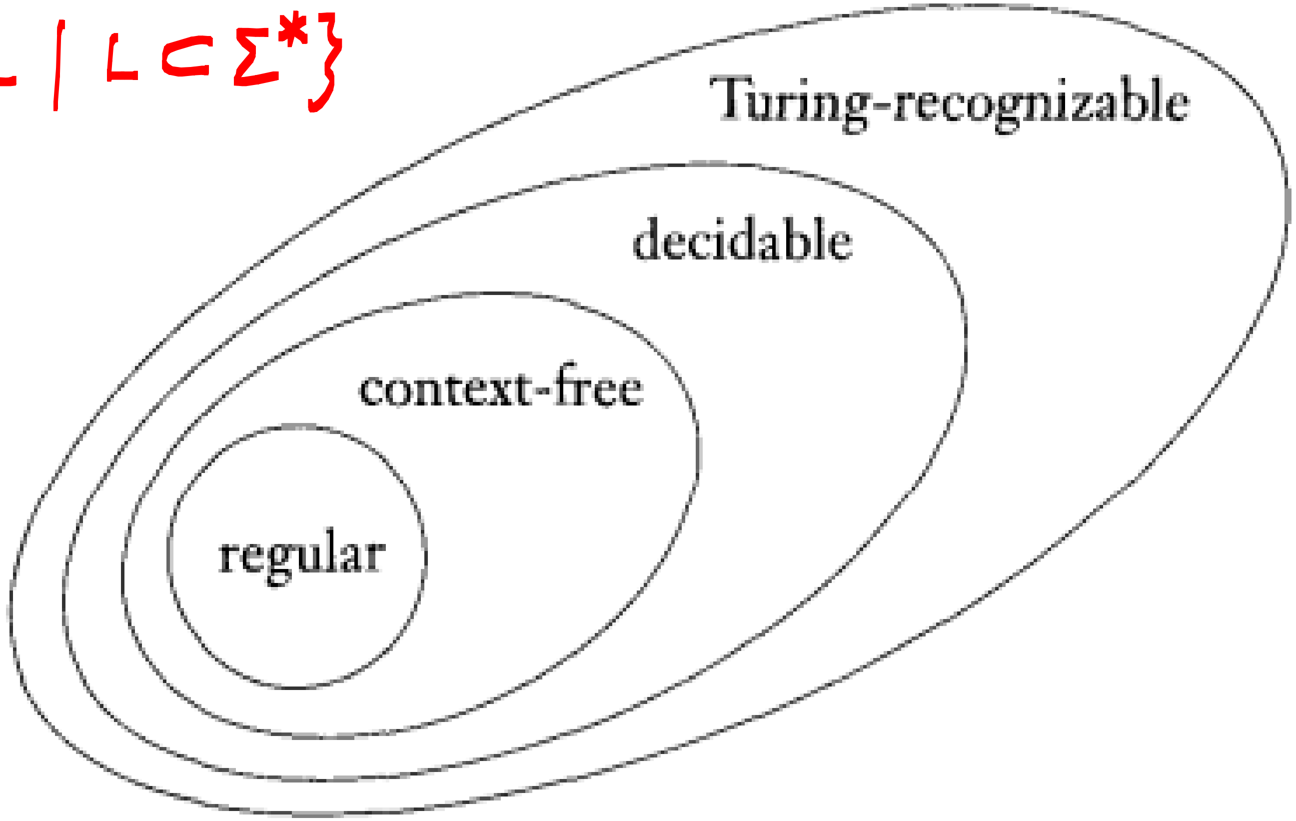
$L \subset \Sigma^*$

⑤ $w \in L$: (a string in L)

: a sentence of L

$$2^{\Sigma^*} = \{L \mid L \subset \Sigma^*\}$$

all languages



$\langle$ Classes of languages $\rangle$

Example 1.9 Let $\Sigma = \{a, b\}$. Then

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, ...\}.$$

The set

$$\{a, aa, aab\}$$

is a language on Σ. Because it has a finite number of sentences, we call it a finite language. The set

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

is also a language on Σ. The strings $aabb$ and $aaaabbbb$ are in the language L, but the string abb is not in L. This language is infinite. Most interesting languages are infinite.

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to Σ^*; that is, the complement of L is

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

The reverse of a language is the set of all string reversals, that is,

$$L^R = \left\{ w^R : w \in L \right\}.$$

The concatenation of two languages L_1 and L_2 is the set of all strings obtained by concatenating any element of L_1 with any element of L_2; specifically,

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

We define L^n as L concatenated with itself n times, with the special cases

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

and

$$L^1 = L$$

for every language L.

Finally, we define the **star-closure** of a language as

$$L^* = L^0 \cup L^1 \cup L^2 \cdots$$

and the **positive closure** as

$$L^+ = L^1 \cup L^2 \cdots.$$

Example 1.10 If

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

then

$$L^2 = \{a^n b^n a^m b^m : n \geq 0, m \geq 0\}.$$

Note that n and m in the above are unrelated; the string $aabbaaabbb$ is in L^2.

The reverse of L is easily described in set notation as

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

but it is considerably harder to describe $\overline{L}$ or L^* this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages. ■

[2] Grammars

Definition 1.1

A grammar G is defined as a quadruple

$$G = (V, T, S, P),$$

where V is a finite set of objects called **variables**,
T is a finite set of objects called **terminal symbols**,
$S \in V$ is a special symbol called the **start variable**,
P is a finite set of **productions**.

It will be assumed without further mention that the sets V and T are nonempty and disjoint.

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$x \to y,$$

where x is an element of $(V \cup T)^+$ and y is in $(V \cup T)^*$. The productions are applied in the following manner: Given a string w of the form

$$w = uxv,$$

we say the production $x \to y$ is applicable to this string, and we may use it to replace x with y, thereby obtaining a new string

$$z = uyv.$$

This is written as

$$w \Rightarrow z.$$

We say that w **derives** z or that z is derived from w. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

$$w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n,$$

we say that w_1 derives w_n and write

$$w_1 \stackrel{*}{\Rightarrow} w_n.$$

The $*$ indicates that an unspecified number of steps (including zero) can be taken to derive w_n from w_1.

Definition 1.2

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

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

is the language generated by G.

If $w \in L(G)$, then the sequence

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

is a **derivation** of the sentence w. The strings $S, w_1, w_2, ..., w_n$, which contain variables as well as terminals, are called **sentential forms** of the derivation.

Example 1.11 Consider the grammar

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

with P given by

$$S \to aSb,$$
$$S \to \lambda.$$

Then

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb,$$

so we can write

$$S \overset{*}{\Rightarrow} aabb.$$

The string $aabb$ is a sentence in the language generated by G, while $aaSbb$ is a sentential form.

Example 1.12 Find a grammar that generates

$$L = \{a^n b^{n+1} : n \geq 0\}.$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra b. This can be done with a production $S \to Ab$, with other productions chosen so that A can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G = (\{S, A\}, \{a, b\}, S, P)$, with productions

$$\begin{aligned} S &\to Ab, \\ A &\to aAb, \\ A &\to \lambda. \end{aligned}$$

Derive a few specific sentences to convince yourself that this works.

Example 1.13 Take $\Sigma = \{a, b\}$, and let $n_a(w)$ and $n_b(w)$ denote the number of a's and b's in the string w, respectively. Then the grammar G with productions

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

generates the language

$$L = \{w : n_a(w) = n_b(w)\}.$$

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars G_1 and G_2 **are equivalent** if they generate the same language, that is, if

$$L(G_1) = L(G_2).$$

As we will see later, it is not always easy to see if two grammars are equivalent.

Example 1.14 Consider the grammar $G_1 = (\{A, S\}, \{a, b\}, S, P_1)$, with P_1 consisting of the productions

$$S \to aAb|\lambda,$$
$$A \to aAb|\lambda.$$

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by $|$. In this notation $S \to aAb|\lambda$ stands for the two productions $S \to aAb$ and $S \to \lambda$.

This grammar is equivalent to the grammar G in Example 1.11. The equivalence is easy to prove by showing that

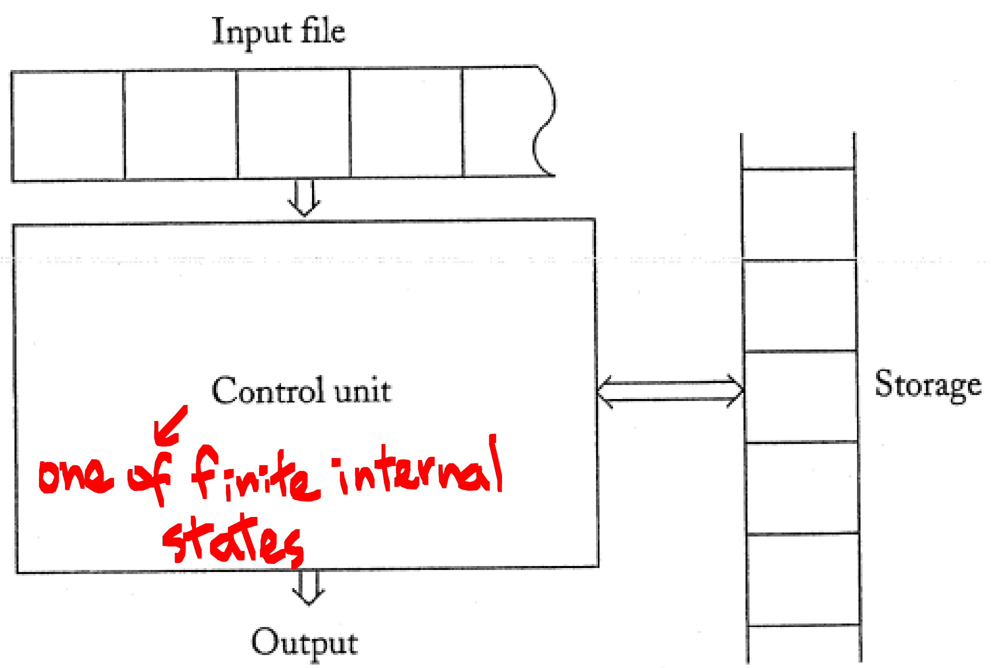
$$L(G_1) = \{a^n b^n : n \geq 0\}.$$

We leave this as an exercise. ■

[3] Automata

An automaton is an abstract model of a digital computer.

Figure 1.4



① next state is determined by transition function.

② configuration is used to refer to a particular state of the control unit, input file, and temporary storage.

③ a move : config. ⇒ config.

④ deterministic automaton
nondeterministic automaton

⑤ accepter : (output response is "yes" or "no".)
transducer : (others)