1 Alphabets, strings, languages

- An **alphabet** is a finite set of **symbols**.

  E.g. the Roman alphabet $\{a,\ldots,z\}$, $\{a,b\}$, $\{0,1\}$,...

- A **string** over an alphabet is a finite sequence of symbols from that alphabet.

- The **empty string** consists of zero symbols. We will denote it by the symbol 'ε'.[1]

  Examples of strings on alphabet $\{a,b\}$ are ε, a, $abbaba$, ...

  The symbols u,v,w,x,y,z are used to name strings – therefore we avoid them as symbols of alphabets.

- The set of all strings – including ε– over an alphabet Σ is denoted by Σ^*.

- The **length** of a string w is denoted by $|w|$.

- The **concatenation** of two strings w and v is formed by sequencing the strings in the given order; it is denoted as wv, $w \circ v$, or $w^\frown v$. Concatenation is associative: $(xy)z = x(yz)$, and $\varepsilon w = w\varepsilon = w$.

- A string v is a **substring** of a string w, if there exists strings x and y such that $w = xvy$. Either or both of x and y can be ε.

- The **reverse** of a string w, denoted as w^R, is defined as follows:

  Definition 1.1 (Reverse of a string)**.**

  i. If $w = \varepsilon$, then $w^R = w$.

  ii. If $w = va$ for some $a \in \Sigma$, then $w^R = av^R$.

- The notation w^n stands for concatenating w to itself for n times. $w^0 = \varepsilon$, $w^1 = w$, $w^2 = ww$, and so on.

- A **language** is a set of strings over a certain alphabet.

[1] Sudkamp uses 'λ' for the empty string, you may also encounter 'e' or other symbols in other books. As 'λ' has another ubiquitous use in computer science and linguistics, I will use 'ε'.

- Therefore a language L on an alphabet Σ is a subset of Σ^*.

  Exercise 1.2.
  Which of the following are languages?

  $$\varepsilon \qquad \{\varepsilon\} \qquad \emptyset \qquad \Sigma \qquad \Sigma^*$$

- Some examples over $\Sigma = \{a,b\}$:

  $$\{b, aa, ab\}$$

  $$\{w \in \Sigma^* \mid w \text{ has equal number of } a\text{'s and } b\text{'s}\}$$

  $$\{w \in \Sigma^* \mid w = w^R\}$$

2 Some operations on languages

- Given that languages are sets, ordinary set operations **union**, **intersection** and **difference** are defined for languages. For the moment we are interested in the union operation.

- There also are operations specific to languages. One is **concatenation of languages**. Given any languages L_1 and L_2 over Σ, their concatenation, designated as $L_1 \circ L_2$, $L_1^\frown L_2$, or simply L_1L_2, is defined as follows:

  $$L_1L_2 = \{w \in \Sigma^* \mid w = xy, \text{ for some } x \in L_1 \text{ and } y \in L_2\} \tag{1}$$

  Exercise 2.1.
  Let $L_1 = \{w \in \{0,1\}^* \mid w \text{ has an even number of 0's}\}$ and
  $L_2 = \{w \in \{0,1\}^* \mid w \text{ starts with a 0 followed by any number of 1's}\}$.
  Which language is L_1L_2?

- Our final and third operation is **closure** (or **star**, or **Kleene closure**) of a language L, denoted as L^*, which is the set of expressions formed by concatenating zero or more strings from L. Formally,

  $$L^* = \{w \in \Sigma^* \mid w = w_1w_2\ldots w_k \text{ for some } k \geq 0 \text{ where } w_1, w_2, \ldots w_k \in L\}$$

or,

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

where $L^0 = \{\varepsilon\}$, $L^1 = L$, and, $L^i = LL\cdots L$, with i-many Ls

- We write L^+ in place of LL^*, which is:

$$L^* = \{w \in \Sigma^* \mid w = w_1 w_2 \ldots w_k \text{ for some } k \geq 1 \text{ where } w_1, w_2, \ldots w_k \in L\}$$

Exercise 2.2.
Compute L^* for (i) $L = \{0,1\}$; (ii) L is the set of strings of 0's, and (iii) $L = \emptyset$.

3 Finite representation of languages

- In the theory of computation and its applications we are interested in representing languages of our interest with *finite* means. This is easy when the language is finite, but it is a challenge for nonfinite languages.

- One method is constructing an **inductive** definition:

Example 3.1 (Inductive definition of a language)**.**
The language L over $\{a,b\}$, where each string begins with an a and has an even length.

i. aa and $ab \in L$.
ii. If $w \in L$, then waa, wab, wba, $wbb \in L$.
iii. Nothing other than the strings obtained via i. and ii. above are in L.

Exercise 3.2.
Write an inductive definition for the language L over $\{a,b\}$ in which every occurrence of b is immediately preceded by an a.

- Now let us see a more transparent and direct way of specifying the above languages. This method involves applying the operations set union, concatenation and closure on sets.

Example 3.3.
The language L over $\{a,b\}$ which has bb as a substring can be defined as $\{a,b\}^*\{bb\}\{a,b\}^*$.

Exercise 3.4.

i. Define the language L over $\{a,b\}$ whose strings either start with aa or end with bb.
ii. Define the language L over $\{a,b\}$ whose strings have an even length. Also define for odd length.
iii. Define the language L over $\{0,1\}$ whose strings have two or three occurrences of 1 the second and third of which are not consecutive.

4 Regular languages

- Another central point of interest in the theory of computation is classes of languages – the set of all languages that share a certain mathematically specifiable property.

- The first class we will look at is the class (or set) of **regular languages** (or **regular sets**).

Definition 4.1 (Regular Languages)**.**
Given an alphabet Σ:

1. $\emptyset$ is a regular language.
2. For any symbol $a \in \Sigma$, $\{a\}$ is a regular language.
3. If A and B are regular languages, so is $A \cup B$.
4. If A and B are regular languages, so is AB.
5. If A is a regular language, so is A^*.
6. Nothing is a regular language unless it fits the above definition.

- In other words, a language is regular if it can be constructed from unit languages like $\{a\}$, $\{b\}$ etc. and the empty language $\emptyset$ by the repeated application of union, concatenation and closure.

Example 4.2.
Show that the following languages are regular.

1. $L = \{x \in \{a,b\}^* \mid x \text{ contains an odd number of } b\text{'s}\}$
2. $L = \{x \in \{a,b\}^* \mid x \text{ contains exactly two or three } b\text{'s}\}$

- **Regular expressions** are notational devices to represent regular languages.

Definition 4.3 (Regular Expressions)**.**
For each regular expression E, the language denoted by it is designated as $L(E)$. The set of regular expressions can be inductively defined as follows.

1. The constants ε and $\emptyset$ are regular expressions, where $L(\varepsilon) = \{\varepsilon\}$ and $L(\emptyset) = \emptyset$.
2. If a is a symbol, $\mathbf{a}$ is a regular expression, where $L(\mathbf{a}) = \{a\}$.
3. If E and F are regular expressions, so is $E \cup F$, where $L(E \cup F) = L(E) \cup L(F)$.
4. If E and F are regular expressions, so is EF, where $L(EF) = L(E)L(F)$.
5. If E is a regular expression, so is E^*, where $L(E^*) = L(E)^*$
6. If E is a regular expression, so is (E), where $L((E)) = L(E)$
7. If E is a regular expression, then it can be shown to be so by 1–6.

Example 4.4.
Let us write a regular expression for the set of strings that consist of alternating 0's and 1's.

Example 4.5.
Write regular expressions for the following languages:

1. The set of strings over alphabet $\{a,b,c\}$ containing at least one a and at least one b.
2. The set of strings of 0's and 1's whose tenth symbol from the right end is 1.
3. The set of strings of 0's and 1's with at most one pair of consecutive 1's.
4. The set of strings of 0's and 1's with no substring 111.

Self study

Sudkamp (1997), Chapter 2 covers most of the material here. Beware that the book uses 'λ' where we use 'ε'.

References

Sudkamp, T. A. (1997). *Languages and Machines: An Introduction to the Theory of Computer Science*. Addison-Wesley, Reading, MA, 2nd edition edition.