Contents

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'.

  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.

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

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

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

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

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

Exercise 1.2
Which of the following are languages?

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

□

---

2 Some operations on languages

- Given that languages are sets, ordinary set operations **union**, **intersection** and **difference** are defined for languages.

- 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 $L_1 L_2$, is defined as follows:

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

- 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,

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

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 2.1
State whether true or false:

1. $\{\varepsilon\}^* = \{\varepsilon\}$
2. $\emptyset^* = \{\varepsilon\}$
3. For any alphabet Σ, any L defined over Σ is such that $L \in \mathscr{P}(\Sigma^*)$. ($\mathscr{P}(X)$ denotes the power set of X.)
4. For any language L, $L^* = (L^*)^*$.
5. For any language L, $\emptyset L = L\emptyset = L$
6. For any language L, $\{\varepsilon\}L = \emptyset$

□

Exercise 2.2
Let $L_1 = \{w \in \{a,b\}^* \mid |w| = 2\}$ and $L_2\{w \in \{a,b\}^* \mid |w| = 3 \text{ and } w \text{ ends with } b\}$

1. Give the concatenation of L_1 and L_2.
2. Give their union.

□

Exercise 2.3
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 $L_1 L_2$?

□

---

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

1. Define the language L over $\{a,b\}$ whose strings either start with aa or end with bb.
2. Define the language L over $\{a,b\}$ whose strings have an even length. Also define for odd length.
3. 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.

- Precedence conventions: Kleene star binds most tightly, then comes concatenation, and finally union. For instance, $\{a\}\{b\}^*$ gives $\{a,ab,abb,abbb,\ldots\}$; if you want $\{\varepsilon,ab,abab,ababab,\ldots\}$, you need to have $(\{a\}\{b\})^*$, Again, $\{a\}\{b\} \cup \{c\}$ gives the set $\{ab,c\}$; if you want to have $\{ab,ac\}$ you need $\{a\}(\{b\} \cup \{c\})$.

  Exercise 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.

- Same precedence conventions as above apply.

Exercise 4.4
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 that consist of alternating 0's and 1's (=strings with no consecutive 0's or 1's).
3. The set of strings of 0's and 1's whose third symbol from the right end is 1.
4. The set of strings of 0's and 1's with at most one pair of consecutive 1's.
5. The set of strings of 0's and 1's with no substring 111.
6. $\{w \in \{a,b\}^* \mid w$ has at least three a's$\}$.
7. $\{w \in \{a,b\}^* \mid w$ has at least three a's or at least two b's$\}$.
8. $\{w \in \{0,1\}^* \mid w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$.
9. $\{w \in \{0,1\}^* \mid w$ starts and ends with the same symbol$\}$.
10. $\{w \in \{a,b\}^* \mid w$ does not begin with $aaa\}$.
11. $\{w \in \{a,b,c\}^* \mid w$ has an odd number of occurrences of $ab\}$.
12. $\{w \in \{a,b\}^* \mid w$ has an even length and odd number of b's$\}$.

$\square$

A Answers for selected exercises

1.2 ε is a string, not a language; $\{\varepsilon\}$, $\emptyset$, Σ, Σ^* are all sets of strings, therefore are all languages.

2.1 1: T, 2: T, 3: T, 4: T,5: F, 6: F

2.2 1. $L_1L_2 = \{aaaab, aaabb, aabab, aabbb, abaab, ababb, abbab, abbbb, baaab, baabb, babab, babbb, bbaab, bbabb, bbbab, bbbbb\}$
2. $L_1 \cup L_2 = \{aa, ab, bb, ba, aab, abb, bbb, bab\}$

2.3 Strings with odd number of 0's.

3.2
i. $\varepsilon \in L$.
ii. If $w \in L$, then wab and $wa \in L$.
iii. Nothing other than the strings obtained via i. and ii. above are in L.

3.4
1. $(\{aa\}\{a,b\}^*) \cup (\{a,b\}^*\{bb\})$;
2. even length: $\{ab,bb,ba,ab\}^*$ or $(\{a,b\}\{a,b\})^*$;
odd length: $\{ab,bb,ba,ab\}^*\{a,b\}$ or $(\{a,b\}\{a,b\})^*\{a,b\}$;
3. $\{0\}^*\{1\}\{0\}^*\{1\}\{0\}^*\{01,\varepsilon\}\{0\}^*$.

4.2 We need to show that the given languages can be defined according to Definition 4.1. There are two tricky points. One, you can only use unit languages in your solution, you are not allowed to use sets like $\{a,b\}$, you need to obtain them by union. Two, you are not given the empty string ε in the definition, so you cannot use the unit language $\{\varepsilon\}$ in your solution. However, you can obtain it from the empty set, since $\emptyset^* = \{\varepsilon\}$.

1. $\{a\}^*\{b\}\{a\}^*(\{a\}^*\{b\}\{a\}^*\{b\}\{a\}^*)^*$
2. $\{a\}^*\{b\}\{a\}^*\{b\}\{a\}^*(\{b\} \cup \emptyset^*)\{a\}^*$

4.4
1. $((a\cup b\cup c)^*a(a\cup b\cup c)^*b(a\cup b\cup c)^*) \cup ((a\cup b\cup c)^*b(a\cup b\cup c)^*a(a\cup b\cup c)^*)$
or $(a\cup b\cup c)^*((a(a\cup b\cup c)^*b) \cup (b(a\cup b\cup c)^*a))(a\cup b\cup c)^*$
or $(a\cup b\cup c)^*((ac^*b) \cup (bc^*a))(a\cup b\cup c)^*$
2. $(1\cup\varepsilon)(01)^*(0\cup\varepsilon)$
3. $(0\cup 1)^*1(0\cup 1)(0\cup 1)$
4. $(0\cup 10)^*(11\cup 1\cup\varepsilon)(0\cup 01)^*$
or $(1\cup\varepsilon)(0\cup 01)^*(10\cup 0)^*(1\cup\varepsilon)$

5. $(0 \cup 10 \cup 110)^*(11 \cup 1 \cup \varepsilon)$
6. $(a \cup b)^* a (a \cup b)^* a (a \cup b)^* a (a \cup b)^*$
7. $((a \cup b)^* a (a \cup b)^* a (a \cup b)^* a (a \cup b)^*) \cup ((a \cup b)^* b (a \cup b)^* b (a \cup b)^*)$
8. $(0(10 \cup 01 \cup 11 \cup 00)^*) \cup ((10 \cup 11)(10 \cup 01 \cup 11 \cup 00)^*)$ or $(0((1 \cup 0)(1 \cup 0))^*) \cup ((10 \cup 11)((1 \cup 0)(1 \cup 0))^*)$
9. $0(1 \cup 0)^* 0 \cup 1(1 \cup 0)^* 1 \cup 0 \cup 1$
10. $\varepsilon \cup a \cup aa \cup (\varepsilon \cup a \cup aa) b (a \cup b)^*$ or $(\varepsilon \cup a \cup aa)(b(a \cup b)^* \cup \varepsilon)$
11. $(a^*c \cup b \cup c)^* a^* ab((a^*c \cup b \cup c)^* a^* ab(a^*c \cup b \cup c)^* a^* ab)^* (a^*c \cup b \cup c)^* a^*$
12. This question is harder than the others. It's quite normal if you were not able to solve it. I included it in the exercises, because it exemplifies the kind of abstract thinking one needs to adopt in dealing with the mathematics of strings and mathematics in general. In short, the difficulty level of this question is above what you will encounter in exams, however studying its solution would be very helpful for understanding and solving simpler questions.

    First, let us observe that, as our grammatical strings will be of even length, they are divisible into 2 symbol chunks – you can scan each string from left to right looking at two symbols at each step:

$$x_1 y_1 x_2 y_2 \dots x_n y_n \tag{2}$$

    for some $n \geq 1$. The reason we say n needs to be at least 1 is that we need to have an odd number of b's in our strings and therefore we cannot have the empty string: there needs to be at least one pair of symbols.

    Now let us observe that in every grammatical string there *must* be at least one pair which is either *ab* or *ba*. The reason is that, if all were *aa* or *bb* we would have an even number of b's, therefore scanning the pair of symbols from left to right, we are certain that we will encounter an *ab* or *ba* before the string is over. There may be more than one *ab* or *ba*, the crucial thing is that there must be at least one.

    As we are sure that there will be at least one *ab* or *ba* somewhere in our grammatical strings, any such string will obey the following incomplete regular expression:

$$\dots (ab \cup ba) \dots \tag{3}$$

    Now the question is what should come in places of '...'. Think for some time before reading on, considering that we want our strings to have an even length and odd number of b's.

    As we already have a string of even length and odd b's (namely *ab* or *ba*), whatever we put to the right or left of it should not change this property. Therefore the incomplete parts should contribute strings of even length and an *even* number of b's, *including the empty string*. If both substrings on the left and right of *ab* or *ba* we already specified in (3) have this property, then we will end up with a string of even length and odd number of b's. Therefore, our task is reduced to writing a regular expression for strings with even length and even number of b's. Now try to write this on your own and check with the solution given below.

    How can we write a regular expression for strings with an even length and even number of b's? First observe that, as we don't know the length of those strings in advance, there must be a repetition with Kleene star. The second observation is that this repetition should contribute an even number of symbols including an even number of b's. The strings *aa* and *bb* fulfill this condition, we can repeat them as many times as we want. So we will have something like $(\varepsilon \cup aa \cup bb)^*$ somewhere in our expression. Is this enough? No, because we may encounter *ab*'s or *ba*'s in our even length string with even number of b's. We cannot repeat these, since each repetition contributes and odd number of b's – namely 1 – thereby running the risk of ending up with odd number of b's. Once again, the critical observation is that when we encounter an *ab* or *ba* in the string, we need to have another *ab* or *ba* somewhere down the string, so that we keep the number of b's balanced at an even number. In oder words when you see an *ab* or *ba* you start to owe another b to be re-paid by another *ab* or *ba*. This point of repayment does not need to come immediately after you start owing the b, because there may come *aa*'s and *bb*'s in between. Bringing all these observations together we arrive at the following regular expression that defines all the strings over the alphabet $\{a,b\}$ with an even length and an even number of b's:

$$(aa \cup bb \cup ((ab \cup ba)(aa \cup bb)^*(ab \cup ba)))^* \tag{4}$$

Bringing all together, we arrive at the desired solution:

$$(aa \cup bb \cup ((ab \cup ba)(aa \cup bb)^*(ab \cup ba)))^*(ab \cup ba)(aa \cup bb \cup ((ab \cup ba)(aa \cup bb)^*(ab \cup ba)))^* \tag{5}$$

Now we can look back at our solution once more to grasp the logic behind it. We observed that whatever grammatical string we have will consist of three portions, the first and last of which can be empty.

First portion: a string of even length with even number of b's; crucially a string where every *ab* or *ba* we encounter is matched by another *ab* or *ba*.

Second portion: one *ab* or *ba*, unmatched.

Third portion: same as the first.

Therefore the expression $(ab \cup ba)$ we wrote in (3) corresponds to the unmatched pair in second portion. Note that some strings may have more than one analysis into such portions. For instance *abbabaababbb* can be broken down in two different ways: (1) *abba ba ababbb*; (2) *abbabaab ab bb*

If you couldn't come up with an answer to the question, please study the solution very carefully. Or, if you arrived at the same answer as here, try to find a shorter one.