Chapter-3: Regular Expressions

Solutions for Review Questions
Q.1 Define the following and give suitable examples:
   i) Regular set
   ii) Regular expression

Solution:
   i) Regular set: Refer to the section 3.5.
   ii) Regular expression: Refer to the section 3.2.

Q.2 Prove that the language \( L = \{a^n \ b^{n+1} \mid n > 0 \} \) is non-regular, using pumping lemma.

Solution:
We must not confuse the \( n \) in the language definition with the constant \( n \) of pumping lemma. Hence, we rewrite the language definition as:

\[ L = \{a^m b^{m+1} \mid m > 0 \}. \]

**Step 1:** Let us assume that the language \( L \) is a regular language. Let \( n \) be the constant of pumping lemma.

**Step 2:** Let us choose a sufficiently large string \( z \) such that \( z = a^l b^{l+1} \), for some large \( l > 0 \); the length of \( z \) is given by: \( |z| = 2l + 1 \geq n \). Since we assumed that \( L \) is a regular language and from the language definition it is an infinite language, we can now apply pumping lemma. This means that we should be able to write \( z \) as: \( z = uvw \).

**Step 3:** As per pumping lemma, every string \( uv^i w \), for all \( i \geq 0 \) is in \( L \). Further, \( |v| \geq 1 \), which means that \( v \) cannot be empty, and must contain one or more symbols.

Let us consider the case when \( v \) contains a single symbol from \( \{a, b\} \). Hence, \( z = uvw = a^l b^{l+1} \), which means that the number of \( b \)'s is one greater than number of \( a \)'s in \( z \). Therefore, as per pumping lemma, we would expect \( uv^2 w \) also to be a member of \( L \). However, this cannot be the case, as \( v \) contains only a single symbol, and pumping \( v \) would yield different number of \( a \)'s and \( b \)'s than what is expected by the language definition. Thus, \( uv^2 w \) is not a member of \( L \), contradicting our assumption that \( L \) is regular.

Let us now consider the case when \( v \) contains both the symbols, i.e., \( a \) as well as \( b \). The sample \( v \) could be written as \( 'ab' \), or \( 'aabb' \), and so on. When we try to pump \( v \) multiple times, such as, for example, \( v^2 = abab \), or \( v^3 = aabbaabb \), and so on, we find that even \( a \)'s can follow \( b \) in the string, which is against
the language definition ‘$a^m b^{m+1}$’, according to which, $a$’s are followed by $b$’s, and not vice versa. Thus, ‘$uv^2w$’ is not a member of $L$, contradicting our assumption that $L$ is regular.

Hence, language $L = \{a^m b^{m+1} \mid m > 0\}$ is non-regular.

Q.3 Explain in brief the applications of finite automata.

Solution:
Refer to the section 3.8.

Q.4 Construct the NFA with $\epsilon$-transitions, which accepts the language defined by:

$$(ab + ba)^* aa (ab + ba)^*$$

Also convert this to a minimized DFA.

Solution:
Refer to the example 3.27 form the book.

Q.5 Construct regular expressions defined over the alphabet $\Sigma = \{a, b\}$, which denote the following languages:

i) All strings without a double $a$.

ii) All strings in which any occurrence of the symbol $b$, is in groups of odd numbers.

iii) All strings in which the total number of $a$’s is divisible by 2.

Solution:

i) Strings without double $a$ means strings without two consecutive $a$’s. Hence, the required RE is,

$$(a + \epsilon) \cdot (b + ba)^*$$

ii) Here, $b$’s exist in groups of odd numbers, i.e., 1, 3, 5 and so on. Hence, the RE is,

$$a^* b (bb)^* a^*$$

iii) Here, we require even number of $a$’s. The required RE is,

$$ (b^* \cdot a \cdot b^* \cdot a \cdot b^*)^* + b^*$$
Q.6 Check the following regular expressions for equivalence and justify:

(i) \( R_1 = (a + bb)^* (b + aa)^* \)
\( R_2 = (a + b)^* \)

(ii) \( R_1 = (a + b)^* abab^* \)
\( R_2 = b^* a (a + b)^* ab^* \)

Solution:

i) Let us write languages denoted by \( R_1 \) and \( R_2 \) as below.

\[ L(R_1) = \{ \varepsilon, a, b, aa, ab, bb, baa, bba, bbaa, babb, abaa, abab, ababba, ... \} \]

\[ L(R_2) = \{ \varepsilon, a, b, aa, ab, ba, bb, ... \} \]

Given regular expressions \( R_1 \) and \( R_2 \) are not equal as the strings produced by them are not same. For example, string ‘ba’ cannot be generated using regular expression \( R_1 \) which can be produced by \( R_2 \).

ii) Let us write languages denoted by \( R_1 \) and \( R_2 \) as below.

\[ L(R_1) = \{ aba, aaba, baba, abab, ababb, ababa, ... \} \]

\[ L(R_2) = \{ aa, baa, baaa, baba, baab, ... \} \]

Given regular expressions \( R_1 \) and \( R_2 \) are not equal as the strings produced by them are not same. For example, string ‘aa’ cannot be produced by Regular Expression \( R_1 \) which can be produced by \( R_2 \).

Q.7 Describe in English the sets denoted by the following regular expressions:

(i) \( (a + \varepsilon) (b + ba)^* \)

(ii) \( (0*1*)^* \)

Solution:

i) Let us write language denoted by the given RE.

\[ L(R) = \{ \varepsilon, a, b, ab, ba, bb, aba, abb, bbb, baba, abab, ... \} \]

Given language consists of strings where two consecutive \( a \)'s cannot occur.

ii) Let us write language denoted by the given RE.

\[ L(R) = \{ \varepsilon, 0, 1, 00, 11, 01, 10, 000, 111, ... \} \]

Given language consists of strings where any combination of 0's and 1's can be observed.

Q.8 Construct an NFA with \( \varepsilon \)-moves, which accepts the language defined by:

\[ [(0 + 1)^* 10 + (00)^* (11)^*]^* \]
Solution:
The NFA with $\varepsilon$-moves is,

![NFA Diagram]

Q.9 Let $R_1$ and $R_2$ be two regular expressions. With the help of transition diagrams, illustrate the three operations ($+$, $\cdot$, $*$) on $R_1$ and $R_2$.

Solution:
Refer to the section 3.4.2.1.

Q.10 Show that the regular expressions, $(a^* bbb)^* a^*$ and $a^* (bbba^*)^*$, are equivalent.

Solution:
Let, $R_1 = (a^* bbb)^* a^*$ and $R_2 = a^* (bbba^*)^*$.

Let us write language denoted by $R_1$ as,
$L(R_1) = \{ \varepsilon, a, aa, bbb, baaa, abbb, bbba abbbba, \ldots \}$

Let us write language denoted by $R_2$ as,
$L(R_2) = \{ \varepsilon, a, aa, bbb, baaa, abbb, bbba, abbbba, \ldots \}$

As we can see that languages denoted by regular expressions are same, i.e., $L(R_1) = L(R_2)$. Therefore, regular expressions $R_1$ and $R_2$ are equivalent.
Q.11 Give a regular expression for representing all strings over \{a, b\} that do not include the sub-strings ‘bba’ and ‘abb’.

Solution:
This essentially requires no consecutive b’s. The RE can be written as,

\[(a + \varepsilon)(b + ba)^*\]

Q.12 Consider the two regular expressions:
\[R_1 = a^* + b^*\]
\[R_2 = ab^* + ba^* + b^*a + (a^* b^*)\]

(i) Find a string corresponding to \(R_1\) but not to \(R_2\).
(ii) Find a string corresponding to \(R_2\) but not to \(R_1\).
(iii) Find a string corresponding to both \(R_1\) and \(R_2\).

Solution:
(i) aaaaaa
(ii) abbbbbbb
(iii) a

Q.13 Construct an NFA for the regular expression, \((a/b)^* ab\). Convert the NFA to its equivalent DFA and validate the answer with suitable examples.

Solution:
It is expected to construct a DFA that recognizes the regular set:

\[R = (a|b)^* \cdot a \cdot b\]

Let us first build the NFA with \(\varepsilon\)-moves and the convert the same to DFA.

The TG for NFA with \(\varepsilon\)-moves is as follows,
Let us convert this NFA with $\varepsilon$-moves to its equivalent DFA using a direct method.

We have relabelled the states as well. Let us see if we can minimize it. The STF for the DFA looks like,

<table>
<thead>
<tr>
<th>Q $\cup \sum$</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
<td>S</td>
</tr>
<tr>
<td>R</td>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>*S</td>
<td>Q</td>
<td>R</td>
</tr>
</tbody>
</table>

We can see that states $P$ and $R$ are equivalent. Hence, we can replace $R$ by $P$ and get rid of $R$. The reduced STF is,

<table>
<thead>
<tr>
<th>Q $\cup \sum$</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
<td>S</td>
</tr>
<tr>
<td>*S</td>
<td>Q</td>
<td>P</td>
</tr>
</tbody>
</table>
The TG for the equivalent DFA is,

Q.14 Define the term: regular language.

Solution:
Refer to the section 3.5.

Q.15 Write short note on: pumping lemma for regular sets.

Solution:
Refer to the section 3.6.

Q.16 Construct an NFA \((Q, \Sigma, \delta, q_0, F)\) for the following regular expression:

\[ 01[((10)^+ + 111)^* + 0]^* 1 \]

Solution:
NFA can be drawn as below.
Q.17  Prove that the regular expressions given below are equivalent.

(i)  \((a^* bbb)^* a^*\)

(ii)  \(a^* (bbb a^*)^*\)

Solution:

Let, \(R_1 = (a^* bbb)^* a^*\) and \(R_2 = a^* (bbb a^*)^*\).

Let us write language denoted by \(R_1\) as,
\[
L(R_1) = \{ \varepsilon, a, aa, aaa, bbb, aaaa, abbb, bbba abbb, ... \}
\]
Let us write language denoted by \(R_2\) as,
\[
L(R_2) = \{ \varepsilon, a, aa, aaa, bbb, aaaa, abbb, bbba, abba, ... \}
\]

As we can see that languages denoted by regular expressions are same, i.e., \(L(R_1) = L(R_2)\). Therefore, regular expressions \(R_1\) and \(R_2\) are equivalent.

Q.18  Describe the language accepted by the following finite automaton.

![Example DFA](image)

Figure 3.38: Example DFA

Solution:

Regular expression can be written as,
\[
(a + b) \cdot ( (a + b) (a + b) )^*
\]

Q.19  Describe as simply as possible in English the language represented by: \((0/1)^* 0\).

Solution:

Let us write the language denoted by the regular expression.
\[
L(R) = \{ 0, 00, 10, 000, 110, 0000, 1110, 010, 0110, ... \}
\]

Given language consists of all the strings over \(\{0, 1\}\) that ends with a 0.

Q.20  Construct an NFA that recognizes the regular expression \((a / b)^* \cdot a \cdot b\). Convert it to a DFA, and draw the state transition table.
Solution:
Refer to the answer for Q.13 above.

Q.21 Construct a regular expression corresponding to the state diagram shown below, using Arden’s theorem.

![Figure 3.39: Example FA](image)

Solution:
Refer to example 3.41 from the book.

Q.22 Is the following language regular? Justify.

\( L = \{0^p 1^p p^{p+q} \mid p \geq 1, q \geq 1\} \)

Solution:
We need to show that the following language is non-regular using Pumping lemma,

\( L = \{0^p 1^p p^{p+q} \mid p \geq 1, q \geq 1\} \)

As we observe the length of every string from the language \( L = 2p + 2q = 2 (p + q) \) is even.

**Step 1:** Let us assume that the language \( L \) is a regular language. Let \( n \) be the constant of pumping lemma.

**Step 2:** Let us choose a sufficiently large string \( z \), such that \( z = xx \), where \( x = 0^p 1^q P^{P+n} \), for some large \( p, q \) > 0; the length of \( z \) is given by: \( |z| = 2 (p + q) \geq n \).

Since we assumed that \( L \) is a regular language and from the language definition it is an infinite language, we can now apply pumping lemma. Hence, we should be able to write \( z \) as: \( z = uvw \).

**Step 3:** As per pumping lemma, every string ‘\( uv^iw \)’, for all \( i \geq 0 \), is in \( L \). Further, \(|v| \geq 1\), which means that \( v \) cannot be empty, and must contain one or more symbols.
Let us consider the case when \( v \) contains a single symbol from \{0, 1\}. We assume \( z = uvw = xx = 0^p1^qP^{pq}0^p1^qP^{pq} \). As per pumping lemma, we would expect \( uv^2w \) also to be a member of \( L \). However, this cannot be the case as \( v \) contains only a single symbol; hence, pumping \( v \) would cause the first \( x \) in string \( xx \) to end with \( v \), and the second \( x \) of string \( xx \) to begin with \( v \). For example, for \( z = 0^p1^qP^{pq}0^p1^qP^{pq} \), after pumping \( v = 0 \) once, we get, \( z_1 = 0^p1^qP^{pq}000^p1^qP^{pq} \), which cannot be represented as a concatenation of two equal sub-strings. Thus, \( uv^2w \) is not a member of \( L \), as it modifies the string of the form \( xx \) to \( xvxv \) rather than \( xvxv \). This contradicts our assumption that \( L \) is regular.

Let us now consider the case when \( v \) contains both the symbols, i.e., 0 as well as 1. The sample \( v \) could be written as 01, or 100, and so on. When we try to pump \( v \) multiple times, we obtain strings of the form, \( xv^i\lambda \), \( xv^i\lambda \), and so on, which is against the language definition \( xx \)—every string is represented as concatenation of two equal sub-strings. Thus, \( uv^i\lambda \), for all \( i \geq 0 \) is not a member of \( L \). This contradicts our assumption that \( L \) is regular.

Hence, language \( L \) is non-regular.

Q.23 Construct the regular expression and finite automata for: \( L = L_1 \cap L_2 \) over alphabet \{a, b\}, where:

\[ L_1 = \text{all strings of even length} \]
\[ L_2 = \text{all strings starting with b} \]

Solution:

Let us list down \( L_1 \) and \( L_2 \) for given conditions.

\[ L_1 = \{ \epsilon, aa, bb, ab, ba, abab, aaba, aabb, bbaa, bbaa, bbbb, baba, bbbab, ... \} \]
\[ L_2 = \{ b, bb, ba, baa, bbb, baab, baaa, baaa, bbbb, ... \} \]

Now, as \( L = L_1 \cap L_2 \),

\[ L = \{ bb, ba, baaa, bbb, baab, baba, ... \} \]

Hence, regular expression for \( L \) can be given as,

\[ r = b (a + b) ((a + b) (a + b))^* \]

Let us construct the NFA with \( \epsilon - \)moves as shown in the diagram below.
Q.24 Which of the following are true? Explain.

1) \( baa \in a^* b^* a^* b^* \)
2) \( b^* a^* \cap a^* b^* = a^* \cup b^* \)
3) \( a^* b^* \cap b^* c^* = \phi \)
4) \( abcd \in [a (cd)^* b]^* \)

Solution:

i) Let \( L \) be the language denoted by the given RE, then,

\[ L = \{ \varepsilon, a, b, ab, aa, aba, abab, ba, baa, baab, ... \} \]

As 'baa' string belongs to language produced by given RE.

Hence, \( baa \in a^* b^* a^* b^* \) is TRUE.

ii) Let \( R_1 = b^* a^* \) then \( L_1 = \{ \varepsilon, b, a, ba, bb, aa, bbb, aaa, baa, bba, ... \} \).
Let \( R_2 = a^* b^* \) then \( L_2 = \{ \varepsilon, a, b, ab, bb, aa, bbb, aaa, abb, aab, ... \} \).
Therefore, \( L_1 \cap L_2 = \{ \varepsilon, a, b, aa, bb, aaa, bbb, ... \} \).

\[ a^* = \{ \varepsilon, a, a, aa, aaa, ... \} \text{ and } b^* = \{ \varepsilon, b, b, bb, bbb, bbbb, ... \} \]
Hence, \( a^* \cup b^* = \{ \varepsilon, a, b, aa, bb, bbb, ... \} \)

Therefore, \( b^* a^* \cap a^* b^* = a^* \cup b^* \) is TRUE.
iii) Let $R_1 = a^*b^*$ then $L_1 = \{ \epsilon, a, b, ab, bb, aa, bbb, aaa, ... \}$ and
Let $R_2 = b^*c^*$ then $L_2 = \{ \epsilon, b, c, bc, bb, cc, bbb, ccc, ... \}$ then
Therefore, $L_1 \cap L_2 = \{ \epsilon, b, bb, bbb, ... \} \neq \phi$
Hence, $a^*b^* \cap b^*c^* = \phi$ is FALSE.

iv) Let $L$ be the language denoted by the given RE, $[a (cd)^* b]^*$ then,
$L = \{ \epsilon, ab, abab, acdb, acdbcb, abacdb, abacdb, ... \}$
'abcd' does not belong to language $L$.
Therefore, $abcd \in [a (cd)^* b]^*$ is FALSE.

Q.25 Construct the regular expressions for the following DFAs:

![DFAs](image)

**Solution:**
i) The state equations for the given DFA are:

\[
A = \epsilon + A0 + B0 \\
B = A1 + B1 \\
B = A11^* \quad \text{... using Arden's Theorem}
\]

Substituting for $B$ in $A$,

\[
A = \epsilon + A0 + A11^*0 \\
= \epsilon + A(0 + 11^*0) \\
= \epsilon(0 + 11^*0)^* \quad \text{... using Arden's Theorem}
\]
Hence, $A = (0 + 11^*0)^*$
A being the final state, regular expression for the given DFA is $(0 + 11^*0)^*$.
ii) Let the third state label be C.

The state equations for the given DFA are:

\[ A = \varepsilon + C0 \]
\[ B = A0 + B0 \]
\[ C = A1 + B1 + C1 \]

Let us try to simplify the equations.

\[ B = A0 + B0 \]
\[ = A00^* \quad \text{... using Arden's Theorem} \]

Substituting B in C we get,

\[ C = A1 + A00^*1 + C1 \]
\[ = A (1 + 00^*1) + C1 \]
\[ = A (1 + 00^*1) 1^* \quad \text{... using Arden's Theorem} \]

Substituting C in A we get,

\[ A = \varepsilon + C0 \]
\[ = \varepsilon + A (1 + 00^*1) 1^* 0 \]
\[ = ((1 + 00^*1) 1^* 0)^* \]

Therefore,

\[ B = ((1 + 00^*1) 1^* 0)^* 00^* \]

Both A and B are the final states for the DFA. Hence, the regular expression pertaining to the DFA is,

\[ A + B \]
\[ = ((1 + 00^*1) 1^* 0)^* (\varepsilon + 00^*) \]

Q.26 Which of the following languages are regular sets? Justify your answer.

(i) \( \{0^n \mid n \geq 1 \} \)

(ii) \( \{0^m 1^n 0^{m+n} \mid m \geq 1 \text{ and } n \geq 1 \} \)

Solution:

(i) It is given that \( n \geq 1 \).

For \( n=1 \), \( 0^2 = 0^2 \), length = 2
For $n=2$, $0^{2n} = 0^4$, length = 4
For $n=3$, $0^{2n} = 0^6$, length = 6

Hence, length of each string is multiples of 2 which is even length.

The language $\{0^n \mid n \geq 1\}$ is a regular language that can be denoted by the regular expression, $(00)^*$.  

(ii) $\{0^m \ 1^n \ 0^{m+n} \mid m \geq 1 \text{ and } n \geq 1\}$ is not a regular set. Refer to the answer for the question 3.22 above.

Q.27  Find out whether given languages are regular or not:

(1) $L = \{ww \mid w \in \{0, 1\}^*\}$
(2) $L = \{1^k \mid k = n^2, n \geq 1\}$

Solution:
Both the given language are not regular.
(1) Refer to the example 3.45 from the book.
(2) Refer to the example 3.43 from the book.

Q.28  With the help of a suitable example, prove: ‘regular sets are closed under union, concatenation, and Kleene closure’.

Solution:
Refer to the section 3.5.2.

Q.29  Explain the following applications of regular expressions:

(1) grep utility in UNIX
(2) Finding pattern in text

Solution:
(1) grep utility in UNIX: Refer to the section 3.8.3.
(2) Finding pattern in text: Refer to the section 3.8.2.

Q.30  Construct the NFA and DFA for the following languages:

(i) $L = \{x \in \{a, b, c\}^* \mid x \text{ contains exactly one } b \text{ immediately following } c\}$
(ii) $L = \{x \in \{0, 1\}^* \mid x \text{ starts with } 1 \text{ and } |x| \text{ is divisible by } 3\}$
(iii) \( L = \{ x \in \{a, b\}^* | x \) contains any number of \( a \)'s followed by at least one \( b \}\)

**Solution:**

The regular expressions denoting the languages mentioned are,

(i) \( r = (a + b + cb)^* \)

(ii) \( r = (0+1)(0+1)(0+1)(0+1)(0+1)\]^* \)

(iii) \( r = a*bb* \)

For NFA/DFA construction refer to the section 3.4.2.

**Q.31** Let \( \Sigma = \{0, 1\} \). Construct regular expressions for each of the following:

(a) \( L_1 = \{ W \in \Sigma^* | W \) has at least one pair of consecutive zeros\}

(b) \( L_2 = \{ W \in \Sigma^* | W \) has no pair of consecutive zeros\}

(c) \( L_3 = \{ W \in \Sigma^* | W \) starts with either ‘01’ or ‘10’\}

(d) \( L_4 = \{ W \in \Sigma^* | W \) consists of even number of 0’s followed by odd number of 1’s\}

**Solution:**

(a) \( r = [ (1 + 0)^* (00) (1 + 0)^* ]^+ \)

(b) \( r = (0 + \epsilon) (1+10)^* \)

(c) \( r = (01 + 10) (1+0)^* \)

(d) \( r = (00)^* 1 (11)^* \)

**Q.32** Construct a regular expression for the following DFA:

![DFA Diagram](image)

**Figure 3.41: Example DFA**

**Solution:**

The state equations for the given DFA are:

\[ q_0 = q_0 b + q_2 a + \epsilon \]

\[ q_1 = q_0 a \]
Let $L = \{0^n \mid n \text{ is a prime number}\}$; show that $L$ is not regular.

**Solution:**

Length of every string in $L$ is a prime number.

**Step 1:** Let us assume that the language $L$ is a regular language. Let $n$ be the constant of the pumping lemma.

**Step 2:** Let us choose a sufficiently large string $z$ such that $z = 0^l$, for some large $l > 0$; the length of $z$ is given by: $|z| = l \geq n$.

Since we assumed that $L$ is a regular language and from the language definition it is an infinite language, we can now apply pumping lemma. This means that we should be able to write $z$ as: $z = uvw$.

**Step 3:** As per pumping lemma, every string ‘$uvw$’, for all $i \geq 0$, is in $L$. Likewise, $|v| \geq 1$, which means that $v$ cannot be empty and must contain one or more symbols.

Let us consider the case when $v$ contains a single symbol:
In this case, \( z = uvw = 0^l \), which means that the number of 0’s in \( z \) is a prime number. As per pumping lemma, we would expect ‘\( uv^2w \)’ also to be a member of \( L \); however, this cannot be possible, as \( v \) contains only a single symbol, and adding one to the prime number length would not always yield perfect prime length. Thus, pumping \( v \) would yield strings with non-prime lengths. Thus, ‘\( uv^2w \)’ is not a member of \( L \). This contradicts our assumption that \( L \) is regular.

Let us now consider the case when \( v \) contains perfect prime number of 0’s. A sample \( v \) could be written as: ‘000’ (three 0’s), or ‘00000’ (five 0’s), and so on. When we try to pump \( v \) multiple times, such as, for example, \( v^2 = 000000 \) (six 0’s), or \( v^2 = 0000000000 \) (10 0’s), and so on, we find that the length does not remain a perfect prime, and we get a string which is against the language definition, which is ‘0’. Thus, we can say that ‘\( uv^2w \)’ is not a member of \( L \). This contradicts our assumption that \( L \) is regular.

Similarly, if we consider that \( v \) contains any number of 0’s, then on pumping it we will get into a situation where the string has non-prime length, which is against the language definition. For example, if \( v \) contains 2 zeros and if we pump it say 2 times, we will get the string “0000” which does not have a perfect prime length.

Hence, the language \( L = \{ 0^n | n \text{ is a prime number} \} \) is non-regular.

Q.34 Prove or disprove the following for regular expressions \( r, s \) and \( t \).

(a) \((rs + r)^* r = r (sr + r)^*\)
(b) \(s (rs + s)^* r = rr^* s (rr^* s)^*\)
(c) \((r + s)^* = r^* + s^*\)
(d) \((r^* s^*)^* = (r + s)^*\)

Solution:
(a) Let \( r_1 = (rs + r)^* r \), hence \( L(r_1) = \{ r, rsr, rrr, rrrr, rsrsr, rsrsrr, ... \} \)
Let \( r_2 = r (sr + r)^* \), hence \( L(r_2) = \{ r, rsr, rrr, rsrsr, rsrsrr, rrrr, ... \} \)
As the language denoted by \( r_1 \) and \( r_2 \) is same, we can say that \( r_1 = r_2 \). Thus, \((rs + r)^* r = r (sr + r)^*\)

(b) Let \( r_1 = s (rs + s)^* r \), hence \( L(r_1) = \{ sr, srsr, srsrsr, srsrsrr, srrrr, sssssr, ssrsrr, ... \} \)
Let \( r_2 = rr^* s (rr^* s)^* \), hence \( L(r_2) = \{ rs, rrsrs, rsrsrrs, rrrrrs, ... \} \)
As the languages denoted by \( r_1 \) and \( r_2 \) are not same. Hence, \( s (rs + s)^* r \neq rr^* s (rr^* s)^* \)
(c) Let \( r_1 = (r + s)^* \), hence, \( L(r_1) = \{ r, s, rr, rs, ss, rrr, sss, ... \} \)

Let \( r_2 = r^* + s^* \), hence, \( L(r_2) = \{ r, s, rr, ss, rrr, sss, rrrr, ... \} \)

The strings like ‘rs’, ‘sr’ ‘rss’ and so on are not part of \( L(r_2) \). Hence, \( (r + s)^* \neq r^* + s^* \)

(d) Let \( r_1 = (r^* s^*)^* \), hence, \( L(r_1) = \{ r, s, rrs, rrr, sss, rrs, sss, rrrs, sssr, ... \} \)

Let \( r_2 = (r^* + s^*) \), hence, \( L(r_2) = \{ r, s, rs, ss, rss, rrr, sss, ssss, rrsr, ... \} \)

As the language denoted by \( r_1 \) and \( r_2 \) is same, we can say that \( r_1 = r_2 \). Thus, \( (r^* s^*)^* = (r + s)^* \)

Q.35 State whether each of the following statements is true of false. Justify your answer. Assume that all languages are defined over the alphabet \{0, 1\}.

(a) If \( L_1 \subseteq L_2 \) and \( L_1 \) is not regular, then \( L_2 \) is not regular

(b) If \( L_1 \subseteq L_2 \) and \( L_2 \) is not regular, then \( L_1 \) is not regular

(c) If \( L_1 \) and \( L_2 \) are not regular, then \( L_1 \cup L_2 \) is not regular

Solution:

(i) If \( (L_1 \subseteq L_2) \) and \( L_1 \) is not regular, then \( L_2 \) is not regular.

The statement is not always true, that means, it is false. Let us consider the example of following languages \( L_1 \) and \( L_2 \).

Let, \( L_1 = \{ 0^* 1^n \mid n \geq 0 \} \)

\( = \{ \epsilon, 01, 0011, 000111, \ldots \} \)

\( L_1 \) here is not a regular language; \( L_1 \) actually is a CFL.

Let, \( L_2 = 0^* 1^* \)

\( = \{ \epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots, 0011, \ldots \} \)

\( L_2 \) is a regular language as we know.

Thus, even though \( (L_1 \subseteq L_2) \) and \( L_1 \) is not regular, \( L_2 \) is regular.

Hence, the statement is false.

(ii) If \( (L_1 \subseteq L_2) \) and \( L_2 \) is not regular, then \( L_1 \) is not regular.

The statement is false.

Let us consider the example of following languages \( L_1 \) and \( L_2 \).

Let, \( L_2 = \{ \text{set of all palindrome strings over } \{0, 1\} \} \)
= \{ \epsilon, 0, 1, 00, 11, 000, 010, 101, 111, 0000, \ldots \}

$L_2$ here is not a regular language; $L_2$ actually is a CFL.

Let, $L_1 = \{ \epsilon, 0, 1, 00, 11, 000, 111, 0000, \ldots \}$

$L_1$ thus contains strings consisting of all 0’s or 1’s or an empty string. $L_1$ is actually a regular language and we can denote it using a regular expression, $r = 0^* + 1^*$.

Thus, even though ($L_1 \subseteq L_2$) and $L_2$ is not regular, $L_1$ is regular.

Hence, the statement is false.

(iii) If $L_1$ and $L_2$ are not regular, then ($L_1 \cup L_2$) is not regular.

The statement is true. As we know most of the languages are closed under union. For example, if we take union of two CFLs the result is also a CFL.

Q.36 Use pumping lemma to check whether the language, $L = \{ ww \mid w \in \{0,1\}^* \}$ is regular or not.

Solution:

Refer to the example 3.45 from the book.