Problem 1

Which of the following statements are correct? If you think a statement is correct, give a proof. If you think it is incorrect, give a counterexample. You may assume the alphabet is $\Sigma = \{a, b\}$.

(a) If $L$ is regular, then $L' = \{x : ax \in L \text{ or } xb \in L\}$ is regular.

(b) If $L$ is regular, then $L' = \{xx^- : x \in L\}$ is regular.

Here $x^-$ is $x$ without its last symbol, e.g. $(bab)^- = ba$, $b^- = \varepsilon$. (We let $\varepsilon^- = \varepsilon$.)

(c) If $L$ is regular, then $L' = \{x : xy \in L \text{ for some string } y\}$ is regular.

(d) If $L_1L_2$ is regular, then $L_2L_1$ is regular.

($L_1 \text{ and } L_2$ can be any pair of languages, not necessarily regular.)

(e) (Extra credit) If $L$ is regular, then $L' = \{wz : zw \in L \text{ for some strings } w, z\}$ is regular.

Solution

(a) Correct. Assume $L$ is regular. We will show that the languages $L_1' = \{x : ax \in L\}$ and $L_2' = \{x : xb \in L\}$ are both regular. Since $L' = L_1' \cup L_2'$ and regular languages are closed under union, it will follow that $L'$ is regular also.

To show $L_1'$ is regular, we take a DFA $M$ for $L$ and show how to modify it into a DFA $M_1'$ for $L_1'$. The DFA $M_1'$ will be exactly as $M$ except for the start state: The start state of $M_1'$ will be the state reached from the start state of $M$ by the transition labeled $a$.

To show $L_2'$ is regular, we modify $M$ into a DFA $M_2'$ for $L_2'$. The DFA $M_2'$ will be exactly as $M$ except for its accept states: The accept states of $M_2'$ will be those states that can reach some accept state of $M$ by a transition labeled $b$.

(b) Incorrect. For example, if $L$ is the language of the regular expression $a^*b$, then $L'$ is the language

$L' = \{a^nba^n\}$.

We show $L'$ is not regular using the pumping lemma. For contradiction suppose $L'$ is regular and let $n$ be the pumping length of $L'$. Then the string $z = a^nba^n$ is in $L'$, but for any partition $z = uvw$ with $|vw| \leq N$, $|v| \geq 1$, the string $uv^2w$ has the form $a^{m}ba^{n}$ with $m > n$, so it is not in $L'$. By the pumping lemma $L'$ cannot be regular.
(c) Correct. Given a DFA $M$ for $L$, we can construct an NFA $M'$ for $L'$. The states of $M'$ include all the states of $M'$, as well as an additional state $q_{acc}$. The only final state of $M'$ is $q_{acc}$. NFA $M'$ has the following transitions: First, all the transitions of $M$ are present in $M'$. Second, for every state $q$ of $M$ such there exists a path from $q$ to some accepting state of $M$, $M'$ contains an $\varepsilon$-transition from $q$ to $q_{acc}$.

For example, if $M$ is the following DFA:

```
+----------------+-------------------+
| q0             |          q1        |
+----------------+-------------------+
| 0             | 1                 |
|                |                   |
+----------------+-------------------+
| q1             |          q2        |
+----------------+-------------------+
| 1             | 0.1               |
|                |                   |
+----------------+-------------------+
```

then $M'$ will be the following NFA:

```
+----------------+-------------------+
| q0             |          q1        |
+----------------+-------------------+
| 0             | 1                 |
|                |                   |
+----------------+-------------------+
| q1             |          q2        |
+----------------+-------------------+
| 1             | 0.1               |
|                |                   |
+----------------+-------------------+
| q0             |          q2        |
+----------------+-------------------+
| 0             | $\varepsilon$     |
|                |                   |
+----------------+-------------------+
| q2             |          $q_{acc}$ |
+----------------+-------------------+
```

We now argue that the language of $M'$ is $L'$. To do this we show that if $w$ is in $L'$ then it must be accepted by $M'$, and if $w$ is accepted by $M'$ then it must be in $L'$.

If $w$ is in $L'$, then $wx \in L$ for some $x \in \Sigma^*$. So after reading $w$, the DFA $M$ will be in a state $q$ from where some final state of $M$ can be reached. On reading input $w$, $M'$ can end up in the same state $q$, from where it can take the $\varepsilon$-transition to $q_{acc}$ and accept $w$. So $w$ is accepted by $M'$.

If $w$ is accepted by $M'$, then upon reading $w$ the NFA $M'$ must have reached the state $q_{acc}$. Since the only transitions pointing to this state are $\varepsilon$-transitions, $M'$ must also have reached some other state $q$ upon reading $w$. By construction, there must exist a path from $q$ to some final state of $M$. Let $x_1, x_2, \ldots, x_n$ be the sequence of transitions of $M$ that lead from $q$ to this final state. Then $wx_1x_2\ldots x_n$ is in $L$, so $w$ is in $L'$.

(d) Incorrect. For example let $L_1$ be the language of the regular expression $a^*$ and $L_2 = \{a^nb^m : n \leq m\}$. Then $L_1L_2$ can be described by the regular expression $a^*b^*$ so it is regular.

On the other hand, $L_2L_1$ is the language

$L_2L_1 = \{a^nb^ma^r : n \leq m \text{ and } r \geq 0\}$.

We show this language is not regular. For contradiction suppose it is and let $m$ be its pumping length. Then the string $z = a^mb^m$ is in $L_2L_1$, but for any partition $z = uvw$ with $|vw| \leq n$ and $|v| \geq 1$, the string $uw^2w$ has the form $a^nb^m$ with $n > m$, which is not in $L_2L_1$. By the pumping lemma $L_2L_1$ cannot be regular.
(e) Correct. Let $M$ be a DFA for $L$ (with $N$ states). We’ll construct an NFA $M’$ for $L’$.

First, $M’$ contains $2N$ copies of $M$ and reset all the starting and accepted states. For the $i^{th}$ pair of copies, denote them as $M_i^1$ and $M_i^2$. Add $\epsilon$ transitions from the original accepted states in $M_i^1$ to the original starting state in $M_i^2$. $M’$ contains a new state $q_0$ to be starting state and add $\epsilon$ transitions to the $i^{th}$ state in $M_i^1$ and let the $i^{th}$ state in $M_i^2$ be accepted state.

First, we show for any $x$ which is accepted by $M’$, $x$ is in $L’$, say $x$ goes to the accepted state in the $i^{th}$ pair. Cut the string when it goes from $M_i^1$ to $M_i^2$, say $wz$, then $zw$ can be accepted by $M$.

Now we show the other direction, for any $x \in L’$, $x$ is accepted by $M’$. $x = wz$ and $zw \in L$. Assume after inputting $z$, it is at state $q_j$, claims $wz$ can be accepted in the $j^{th}$ state in $M_i^2$. Start from the $j^{th}$ state in $M_i^1$, after inputting $w$, we can go across the $\epsilon$ transition and go to $M_i^2$, after inputting $z$, we get to the $j^{th}$ state in $M_i^2$, which is accepted state.

Problem 2

Which of the following languages are regular, and which are not? To show a language is regular, describe a DFA, NFA, or regular expression for it (with explanation). To show a language is not regular, prove it using the pumping lemma. The alphabet is $\Sigma = \{a, b, c\}$.

(a) $L_1 = \{wz: |w| = |z|, w \in (a+b)^* \text{ and } z \in (b+c)^*\}$.

(b) $L_2 = \{w: \text{every } a \text{ in } w \text{ is followed by at least one } b \text{ and at least one } c\}$.

For example, $\epsilon, abaacb \in L_2$, but $abacc \notin L_2$.

(c) $L_3 = \{w: w \text{ does not have the same number of } as, bs, \text{ and } cs\}$.

(d) $L_4 = \{w: w \text{ contains the same number of patterns } ac \text{ and } abc\}$.

Solution

(a) $L_1$ is not regular. Suppose it was and let $n$ be its pumping length. Then $z = a^n c^n$ is in $L_1$, but for every partition $z = wvw$ with $|w| \leq n$ and $|v| \geq 1$, $wv^2w$ has more $a$s than $c$s so it is not in $L_1$. By the pumping lemma $L_1$ cannot be regular.

(b) It is regular. There are two cases: Either an $a$ does not appear in the string, or if it does, then the string must end with a pattern of $bs$ and $cs$ containing at least one $b$ and at least one $c$. We write a regular expression for each of these cases. The case of not having any $as$ is represented by the regular expression $(b+c)^*$. Otherwise, the expression must end by $bs$ and $cs$ including at least one of each, that is $bb^*c(b+c)^*$ (if the $b$ comes first) or $cc^*b(b+c)^*$ (if the $c$ comes first). Putting everything together and simplifying a little bit, we get the following regular expression for $L_2$:

$$(b+c)^* + (a+b+c)^*(bb^*c + cc^*b)(b+c)^*$$
Alternatively, we can design an NFA. This NFA accepts all strings that consist of only bs and cs in its start state $q_0$. As soon as an a shows up, it goes to state $q_1$, where it waits until the end of the string, which consists of bs and cs and contains at least one of each. If this part starts with a b, the NFA takes the path $q_1 \rightarrow q_2 \rightarrow q_4$, and if it starts with a c, it takes the path $q_1 \rightarrow q_3 \rightarrow q_4$.

\[
\begin{array}{c}
\text{start} & \xrightarrow{b,c} & q_0 \\
\downarrow{} & & \\
\downarrow{a} & \xrightarrow{a,b,c} & q_1 \\
\downarrow{} & & \\
\downarrow{b} & \xrightarrow{b} & q_2 \\
\downarrow{} & & \\
\downarrow{c} & \xrightarrow{c} & q_3 \\
\downarrow{} & & \\
\downarrow{b} & \xrightarrow{b,c} & q_4
\end{array}
\]

(c) $L_3$ is not regular. The easiest way to show this is to look at its complement language $\overline{L}_3$:

\[\overline{L}_3 = \{w : w \text{ has an equal number of as and bs and cs}\}.
\]

Suppose $\overline{L}_3$ is regular and let $n$ be its pumping length. Then $z = a^n b^n c^n$ is in $\overline{L}_3$, but for every partition $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$, the string $uv^2w$ will have more as than bs and more bs than cs, so it is not in $\overline{L}_3$. By the pumping lemma, $\overline{L}_3$ cannot be regular. Then $L_3$ cannot be regular either: For if $L_3$ was regular, by closure properties $\overline{L}_3$ would be regular too, a contradiction.

(d) $L_4$ is not regular. Suppose it is and let $n$ be its pumping length. Let $z = (ac)^n(abc)^n$, which is in $L_4$. However, if we write $z = uvw$ with $|uv| \leq n$ and $|v| \geq 1$, the string $uv^0w = uv$ would contain fewer patterns of the type ac but the same number of patterns of the type abc, so $uvw$ is not in $L_4$. By the pumping lemma $L_4$ cannot be regular.
Problem 3

This problem concerns the following DFA.

(a) Run the minimization algorithm on this DFA. Clearly show the different stages that the minimization algorithm goes through.

(b) Show that every pair of states in the minimized DFA is distinguishable.

(c) Convert the minimized DFA into a regular expression. You may use the conversion algorithm from class, or write a regular expression by hand and argue that it is equivalent to the DFA.

Solution

(a) To obtain the minimized DFA, we mark all pairs of distinguishable states. First, every state (but \( q_{11} \)) is distinguishable from \( q_{11} \) because it is an accepting state. Then, on transition 1, we note that \( q_1 \) and \( q_{01} \) go to \( q_{11} \), while \( q_0 \), \( q_{00} \) and \( q_{10} \) go to a state distinguishable from \( q_{11} \). So these are all distinguishable pairs. At this point we have the following pairs of distinguishable states:

\[
\begin{array}{c|cccccccc}
q_0 & q & q_0 & q_0 & q_1 & q_0 & q_0 & q_1 & q_1 \\
q_1 & x & x & x & x & x & x & x & x \\
q_{00} & x & x & x & x & x & x & x & x \\
q_{01} & x & x & x & x & x & x & x & x \\
q_{10} & x & x & x & x & x & x & x & x \\
q_{11} & x & x & x & x & x & x & x & x \\
\end{array}
\]

No more pairs of states can be distinguished. The indistinguishable states split into three classes: Class A consisting of states \( \{q, q_0, q_{00}, q_{10}\} \); class B, consisting of \( \{q_1, q_{01}\} \), and class C, consisting of \( q_{11} \). This yields the following minimized DFA:
(b) States $A$ and $B$ can be distinguished by 1; $A$ and $C$ can be distinguished by 0, 1 and $\varepsilon$; and $B$ and $C$ can be distinguished by 0 and $\varepsilon$.

(c) We convert it step by step. First, add a new starting state with no incoming arrows and a new accepting state with no outgoing arrows.

We now turn this 5-state NFA into a 2-state generalized NFA by eliminating states. The elimination can be done in arbitrary order, so let us start by eliminating $q_C$:

We now eliminate $q_B$ to obtain

Finally, we eliminate $q_A$ and obtain

which gives the regular expression $(0+1)^*11(0+1)^*$. This expression represents all strings that contain the pattern 11; the part $(0+1)^*$ represents the part before the first occurrence of 11, and the last part $(0+1)^*$ allows for any ending. An alternative regular expression for the same language is $(0+1)^*11(0+1)^*$.
Problem 4

You have a file `propernames` which contains a list of first names in English. Write `grep` commands (including a short explanation about how they work) that will search for the following information in the file. You can use `grep -E -i` to ignore distinctions between uppercase and lowercase.

(a) Any name that contains exactly two vowels, as in Amy.
(b) Any name that any a is followed by at least two consonants, as in Ahmed.
(c) Any name that does not contain the pattern ale, as in Emma but not Alex.
(d) Any name that contains both the patterns in and ri, as in Kristin (but not Nana).

You can test your search patterns with the file

(a) `grep -E -i '^[aeiouy]*[aeiouy][aeiouy]*{2}$' propernames`
   The expression .*[aeiouy]*[aeiouy]* matches a pattern that contains at least two vowels and we must add < and $ to make sure there are exactly two vowels.

(b) `grep -E -i '^[a]*([a][aeiouy]\{2}[aeiouy]*)?*$' propernames`
   Notice strings that do not contain a's should also be accepted. The same, we need < and $ to make sure bad cases will not happen in the front or back.

(c) `grep -E -i '^[a]*([a][^ae]*|[^ae][a]*)?[a|al]?*$' propernames`
   There are two cases that the name does not contain the pattern ale. First, a is not followed by l. Second, if a is followed by a l, then the l is not followed by e. Also, we need < and $ to make sure that there are not such pattern in the front or back.

(d) `grep -E -i '(in.*ri|ri.*in|rin)*' propernames`
   There are three cases, the name first contains the pattern in and then ri, or first ri and then in, or contains rin.