Problem 1

Let $M$ be a non-deterministic finite automaton. Show that there exists a non-deterministic finite automaton $M'$ with the same language as $M$, the same number of states as $M$, but without any $\varepsilon$-transitions.

Solution Sketch

Let $M$ be an NFA over some alphabet $\Sigma$.

Construction of $M'$: The NFA $M'$ has the same set of states and the same start state as $M$ (Only the transitions and accept states of $M$ and $M'$ are different.) For all states $q, q'$ and every alphabet symbol $x \in \Sigma$, we add an $x$-transition in $M'$ from $q$ to $q'$ if we can go from $q$ to $q'$ in the state diagram of $M$ via some number of $\varepsilon$-transitions and exactly one $x$-transition. By construction $M'$ has only transitions labeled with alphabet symbols (in particular, $M'$ has no $\varepsilon$-transitions). We make a state $q$ an accept state in $M'$ if we can go from $q$ to an accept state of $M$ by some number of $\varepsilon$-transitions. (In particular, accept states of $M$ are accept states in $M'$.)

$L(M) \subseteq L(M')$: We are to show that every word $w$ accepted by $M$ is also accepted by $M'$. Consider an accepting computation path for $w$ in $M$. This path consists of a sequence of $\varepsilon$-transitions interspersed with transitions corresponding to alphabet symbols. We can replace any $\Sigma$-transition and all directly preceding $\varepsilon$-transitions by a transition in $M'$. The last part of the path consists of a sequence of $\varepsilon$-transitions from a state $q$ to an accept state of $M$. That means that $q$ is an accept state of $M'$. It follows that there exists an accepting computation path for $w$ in $M'$. 
$L(M') \subseteq L(M)$: We are to show that every word $w$ accepted by $M'$ is also accepted by $M$. Consider an accepting computation path for $w$ in $M'$. By construction of $M'$, we can replace every $x$-transition in $M'$ by a sequence of $\varepsilon$-transitions and an $x$-transition in $M$. If the computation path ends in an accept state $q$ of $M'$ that is not accepting in $M$, we can extend the computation path by a sequence of $\varepsilon$-transition to an accept state of $M$ (by definition of the accept states of $M'$).

Problem 2

For a word $w = x_1 \ldots x_n \in \Sigma^*$, define the reverse of $w$, $w^R$, to be $x_n \ldots x_1$.

Let $L \subseteq \Sigma^*$ be a regular language.

Show that the reverse language $L^R := \{w^R \mid w \in L\}$ is also regular.

Solution

Since $L$ is regular, it is the language accepted by some NFA $M_L = (Q, \delta, q_0, F)$. Construct an NFA $M_{L,R} = (Q', \delta', q'_0, F')$ by reversing the direction of all transitions, making the former start state the unique accepting state and adding a new starting state with epsilon transitions to all former accepting state. Formally, this may be defined as follows:

$$Q' = Q \cup \{s\};$$
$$\delta'(q, c) = \begin{cases} F & \text{if } q = s \text{ and } c = \varepsilon \\ \{t \in Q \mid q \in \delta(t, c)\} & \text{else} \end{cases};$$
$$q'_0 = s;$$
$$F' = \{q_0\}.$$

This automaton accepts exactly $L^R$: for if a string is in $L$, it must be accepted by $M_L$ after following some sequence of transitions from the starting state to some state $f \in F$, i.e. $q_0 \xrightarrow{c_1} a_1 \xrightarrow{c_2} \ldots \xrightarrow{c_n} f$, and then $s \xrightarrow{\varepsilon} f \xrightarrow{c_n} \ldots \xrightarrow{c_2} a_1 \xrightarrow{c_1} q_0$ is a sequence of transitions from the start state to the accepting state in $M_{L,R}$ which consumes all the characters among the $c_i$ in reverse order. Conversely, if a string is in $L_R$, it must necessarily be accepted by some sequence of transitions of the latter form (as no transitions lead back to $s$) which gives rise to an accepting transition sequence consuming its reverse in $L$.

Problem 3

Let $L \subseteq \Sigma^*$ be a regular language.
Show that the cycle language,
\[ \text{cycle}(L) = \{ w_2w_1 \mid w_1w_2 \in L \}, \]
is also regular.

**Solution sketch**

Suppose \( L \) is the language recognised by the NFA \( M = (Q, \delta, q_0, F) \). We can define an NFA \( M' = (Q', \delta', q'_0, F') \) in the following rather opaque way:

\[
\begin{align*}
Q' &= \{0, 1\} \times Q \times Q \uplus \{s\} \\
\delta'(q, c) &= \begin{cases} 
\{(0, q, q) \mid q \in Q\} & \text{if } q = s \text{ and } c = \varepsilon \\
\{(1, l, q_0)\} & \text{if } q = (0, l, r) \text{ for some } l, r, c = \varepsilon \text{ and } r \in F \\
\{p\} \times \delta(r, c) & \text{if } q = (p, l, r) \text{ for some } l, r
\end{cases} \\
q'_0 &= s; \\
F' &= \{(1, q, q) \mid q \in Q\}.
\end{align*}
\]

Informally, we are

- creating two copies (a ‘pre-wraparound’ and ‘post-wraparound’ copy) of the states and transitions of \( M \) for every state of \( M \), corresponding to the possible splitting points between \( w_1 \) and \( w_2 \)
- creating a new non-recurrent start state \( s \)
- adding \( \varepsilon \)-transitions from each ‘pre-wraparound’ copy of the original accepting states to the respective ‘post-wraparound’ copy’s instance of the start state of \( M \) to allow the wrapping around to the beginning of \( w_1 \) after \( w_2 \) has been consumed
- making each ‘post-wraparound’ diagonal state \( (1, q, q) \) accepting, so that the string may be accepted after the original starting point has been reached again

We must now proceed to show that for every \( w \in L \), each \( w_2w_1 \) will be accepted by this NFA no matter how we split \( w \) as \( w_1w_2 \), and conversely that every string \( v \) that is accepted by \( M' \) can be split as \( w_2w_1 \) so that \( w_1w_2 \in L \).

**Problem 4**

Let \( L \subseteq \Sigma^* \) be some language over \( \Sigma \), not necessarily regular. Define an equivalence relation \( \sim_L \) on \( \Sigma^* \), the set of finite strings over \( \Sigma \), by saying that
$u \sim_L v$ if and only if for any $w \in \Sigma^*$, either both $uw$ and $vw$ are in $L$ or neither is.

(a) Suppose that $L_M$ is the language accepted by some DFA $M$. Show that the number of $\sim_{L_M}$-equivalence classes is finite. (Hint: Consider the state that $M$ will be in after reading a given string $u$.)

(b) Consider the language $L_{\text{match}} = \{a^n b^n \mid n \geq 0\}$. Show that there are infinitely many $\sim_{L_{\text{match}}}$-equivalence classes.

(c) Show that if for some language $L$, there are only finitely many $\sim_L$-equivalence classes, then $L$ is regular.

Solution

Parts (a) and (c) of this problem actually asked you to prove the two directions of the Myhill-Nerode theorem, an important result stating that the regular languages are completely characterised by the number of $\sim_L$ equivalence classes being finite. Note that this sets it apart from the Pumping Lemma, whose converse does not hold (consider e.g. $\{11^n0^n1^n \mid n \in \mathbb{N}\}$) nor can even easily be made to hold with the various possible strengthenings.
(a) For any DFA, the state that the DFA will be in after reading a given string is uniquely defined by the transition function $\delta$. Furthermore, if consuming two strings $s$, $s'$ results in $M$ being in the same state $q$, then $s \sim_{L_M} s'$, as any accepting sequence of transitions for a string $sw$ would give rise to an accepting sequence of transitions for the string $s'w$ and vice versa by simply replacing the transitions leading up to $q$ as needed. Hence, the number of $\sim_{L_M}$-equivalence classes is bounded above by the number of states of $M$, which is finite by the definition of a DFA. □

(b) For any $0 < i < j \in \mathbb{N}$, $a^i \not\sim_{L_{\text{match}}} a^j$, for clearly $a^k w \in L_{\text{match}}$ iff $w \in \{a^n b^{k+n} \mid n \geq 0\}$ and all such sets are distinct for different $k$. Hence, each $0 < i \in \mathbb{N}$ gives rise to a different $\sim_{L_{\text{match}}}$-equivalence class and so there must be at least $|\mathbb{N}| = \aleph_0$. □

(c) Say the set of equivalence classes is $E = \Sigma^*/\sim_L$, and for any string $s$, denote the unique equivalence class $e \in E$ containing $s$ as $\overline{s}$. We can create a DFA that accepts $L$ as follows:

$$Q = E$$

$$\delta(e, c) = \overline{sc} \text{ for some } s \in e \text{ (})$$

$$q_0 = \overline{\varepsilon}$$

$$F = \{e \in E \mid \exists s \in L. s \in e\}.$$ 

Here, the definition (**) is consistent because if $s \sim_L s'$, then $sc \sim_L s'c$ for any character $c$. This is true because $sw \in L \iff s'w \in L$ for any string $w$ does more specifically imply $scw' \in L \iff s'cw' \in L$ for any $w'$, and hence $\overline{sc}$ does not depend on the choice of $s$.

It is evident (e.g. by induction on string length) that after reading any string $s$, this automaton will be in the state $\overline{s}$, so every string in $l \in L$ will be accepted by it (by definition, $l \in \overline{l}$ and hence $\overline{l} \in F$). Furthermore, no string $x \not\in L$ may be accepted by it, for if $\overline{x} \in F$, then $\exists s \in L. s \in \overline{x}$, and hence $s \sim_L x$, so $s = s \varepsilon \in L \Rightarrow x \varepsilon = x \in L$, resulting in a contradiction. So $L$ is indeed the language accepted by this DFA. □
Problem 5

(a) For languages \( A, B \subseteq \Sigma^* \), the \textit{perfect shuffle} of \( A \) and \( B \) is the language

\[
\{ x_1y_1 \cdots x_ky_k \mid x_1 \cdots x_k \in A \text{ and } y_1 \cdots y_k \in B, \text{ symbols } x_1, \ldots, x_k, y_1, \ldots, y_k \in \Sigma \}
\]

Show that if \( A \) and \( B \) are regular, then the perfect shuffle of \( A \) and \( B \) is also regular.

(b) For languages \( A, B \subseteq \Sigma^* \), the \textit{shuffle} of \( A \) and \( B \) is the language

\[
\{ u_1v_1 \cdots u_kv_k \mid u_1 \cdots u_k \in A \text{ and } v_1 \cdots v_k \in B, \text{ strings } u_1, \ldots, u_k, v_1, \ldots, v_k \in \Sigma^* \}
\]

Show that if \( A \) and \( B \) are regular, then the shuffle of \( A \) and \( B \) is also regular.

Solution sketch

(b) If \( M_A = (Q_A, \delta_A, q_0, A, F_A) \), \( M_B = (Q_B, \delta_B, q_0, B, F_B) \) are NFAs accepting \( A \) and \( B \) respectively, create a new NFA \( M = (Q, \delta, q_0, F) \) as follows:

\[
Q = Q_A \times Q_B \\
\delta((q_A, q_B), c) = \{(q'_A, q_B) \mid q'_A \in \delta_A(q_A, c)\} \\
\cup \{(q_A, q'_B) \mid q'_B \in \delta_B(q_B, c)\} \\
q_0 = (q_0, q_0, 0) \\
F = F_A \times F_B.
\]

Prove correctness mechanically.

(a) Inspired by the thought that we obtain the perfect shuffle if we, precluding \( \varepsilon \)-transitions, force the above automaton to make transitions belonging to \( A \) and \( B \) alternatingly, if \( M_A = (Q_A, \delta_A, q_0, A, F_A) \), \( M_B = (Q_B, \delta_B, q_0, B, F_B) \) are DFAs accepting \( A \) and \( B \) respectively, create (being rather wasteful with states) a new NFA \( M = (Q, \delta, q_0, F) \) as follows:

\[
Q = Q_A \times Q_B \times \{0, 1\} \\
\delta((q_A, q_B, p), c) = \{(q'_A, q_B, 1) \mid q'_A \in \delta_A(q_A, c) \land p = 0\} \\
\cup \{(q_A, q_B, 0) \mid q'_B \in \delta_B(q_B, c) \land p = 1\} \\
q_0 = (q_0, q_0, 0) \\
F = F_A \times F_B \times \{0\}.
\]

Prove correctness mechanically.