1. (10 points) Define the *shuffle-reverse* operation \( \otimes : \Sigma^* \times \Sigma^* \rightarrow 2^{\Sigma^*} \) for any alphabet \( \Sigma \) by

- \( \varepsilon \otimes (ax) = \{(\varepsilon \otimes x) a\} \),
- \( (ax) \otimes \varepsilon = \{(x \otimes \varepsilon) a\} \),
- \( ax \otimes by = \{x \otimes by\} : \{a\} \cup \{ax \otimes y\} : \{b\} \)

for all \( a, b \in \Sigma, x, y \in \Sigma^* \). For example, \( 01 \otimes ab = \{ba10, b1a0, 1ba0, 10ba, lb0a, b10a\} \). We extend the definition to languages by \( L_0 \otimes L_1 = \bigcup_{xy \in L_0} x \otimes y \). For example,

\[
\{01,0\} \otimes \{ab\} = \{ba10, b1a0, 1ba0, 10ba, lb0a, b10a, ba0, b0a, 0ba\}
\]

Prove that for any regular languages \( A \) and \( B \), language \( A \otimes B \) must be regular. (Hint: Decompose the \( \otimes \) operation into the composition of two operations.)

2. (12 points) A state \( q \) in an NFA \( N = (Q, \Sigma, \Delta, S, F) \) is *useless* if

- \( \neg(\exists z \in \Sigma^*) q \in \hat{\Delta}(S, z) \) \( (\) there doesn’t exist a string taking \( N \) from a start state to \( q \), \( ) or,

- \( \neg(\exists z \in \Sigma^*) \hat{\Delta}(q, z) \cap F \neq \emptyset \) \( (\) there doesn’t exist a string taking \( N \) from \( q \) to a final state). \( )

**a** Can we remove all useless states from any NFA without changing the language that it accepts? Justify your answer.

**b** If we want to remove useless states, we notice that the definition of a useless state involves testing for an infinite set of strings, \( \Sigma^* \). Describe a finite time algorithm to test if \( q \) is useless.

**c** Describe an algorithm which decides whether the language accepted by NFA \( N = (Q, \Sigma, \Delta, S, F) \) is infinite. You may assume that \( N \) does not have \( \varepsilon \)–transitions.

3. (12 points) For each of the following languages over \{a, b, c\}, tell whether or not it is regular and justify your response.

**a** The set of all strings that have a substring of length 4 which starts and ends with the same symbol.

**b** The set of strings with the same number of bs as cs. So \( cab \) and \( a \) each belong to our language, and \( cabab \) does not belong.

**c** The set of all strings that do not have a substring of length 4 which starts and ends with the same symbol.
1. It is easier to break this problem into two steps, by breaking $\otimes$ into the composition of two operations, the first shuffles two strings, and the second takes the reverse of a string. Then we show that regular sets are closed under both of these operations. We first show that applying the *shuffle* operation $\Psi: \Sigma^* \times \Sigma^* \rightarrow 2^{\Sigma^*}$

- $e\Psi x = \{x\}$,
- $x\Psi e = \{x\}$,
- $ax\Psi by = \{a\} \cdot \{x\Psi by\} \cup \{b\} \cdot \{ax\Psi y\}$

for all $a, b \in \Sigma$, $x, y \in \Sigma^*$, to regular languages yields a regular language.

If $L_0$ and $L_1$ are regular, then there are DFAs $M_0 = (Q_0, \Sigma, \delta_0, s_0, F_0)$ and $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ such that $L_0 = L(M_0)$ and $L_1 = L(M_1)$. The idea behind the following machine is to run $M_0$ and $M_1$ in parallel, using nondeterminism to guess which machine should consume the next input symbol and advance its state. A string is accepted by $N_{M_0 \Psi M_1}$, if there exists a path which consumes the string and takes each of $M_0$ and $M_1$ to final states.

$$N_{M_0 \Psi M_1} = (Q_0 \times Q_1, \Sigma, \Delta, \{(s_0, s_1)\}, F_0 \times F_1)$$

where $\Delta((q_i, q_j), a) = \{(\delta_0(q_i, a), q_j), (q_i, \delta_1(q_j, a))\}$. So, by extending the definition of $\Psi$, if $L_0$ and $L_1$ are regular, then so is $L_0 \Psi L_1 = \bigcup_{x \in L_0} \bigcup_{y \in L_1} x\Psi y$, and there must be a DFA to accept $L_0 \Psi L_1$.

As we showed in class, and also as stated in our text, regular languages are closed under reversal. So because there must be a DFA to accept $L_0 \Psi L_1$, there must be a DFA to accept $L_0 \otimes L_1$.

2. **String** $z \in L(N)$ if and only if there is a path labeled $z$ in $N$ from some $s \in S$ to some $f \in F$. If $q$ is useless, then it can’t belong to any such path, either because we can’t get some $s \in S$ to $q$ or we can’t get from $q$ to some $f \in F$ (so no suffix of $z$ can get us from $q$ to some $f \in F$). So removing useless state $q$ from $Q$ can not change the language accepted by $N$.

**b** By the Pigeonhole Principle, if $\exists z \in \Sigma^* q \in \hat{\Delta}(S, z)$ then

$$\exists z \in \Sigma^* (q \in \hat{\Delta}(S, z)) \land (|z| < |Q|).$$

That is, there is a path in $N$ from a state of $S$ to $q$ which does not pass through any state twice. Likewise, if $\exists z \in \Sigma^* \hat{\Delta}(q, z) \cap F = \emptyset$ then

$$\exists z \in \Sigma^* (\hat{\Delta}(q, z) \cap F \neq \emptyset) \land (|z| < |Q|).$$

That is, there is a path in $N$ from $q$ to a state of $F$
which does not pass through any state twice. Hence, to test if $q$ is useless, we only need
test a finite set of strings, $\{z \in \Sigma^* \mid |z| < |Q|\}$.

**Useless?**($q$)

Reachable $\leftarrow$ false

for each $z \in \Sigma^*$, $|z| \leq |Q| - 1$ do

if $q \in \hat{\Delta}(S, z)$ then Reachable $\leftarrow$ true

if $\neg$ Reachable then return “$q$ is useless”

GettoF $\leftarrow$ false

for each $z \in \Sigma^*$, $|z| \leq |Q| - 1$ do

if $F \cap \hat{\Delta}(\{q\}, z) \neq \emptyset$ then GettoF $\leftarrow$ true

if $\neg$ GettoF then return “$q$ is useless”

return “$q$ is not useless”

c Remove useless states from $N$, and there is cycle (loop) in the new NFA if and only if
the language it accepts is infinite.

3. **a** The language is regular because it is described by the regular expression

$$(a+b+c)^* a (a+b+c) (a+b+c) a (a+b+c)^* +$$

$$(a+b+c)^* b (a+b+c) (a+b+c) b (a+b+c)^* +$$

$$(a+b+c)^* c (a+b+c) (a+b+c) c (a+b+c)^*$$

**b** The language is not regular. If it were, then there would be an $n$ such that every string $z$
in the language could be written $z=uvw$ with $|uv| \leq n$, $|v| > 0$, such that every string in
$\{uv^lw \mid l \geq 0\}$ would also belong to the language. Choose $z=b^nc^n$. Any attempt to express
$z$ as $uvw$ with $|uv| \leq n$ and $|v| > 0$ would have $v=b^i$, $i > 0$. So $uv^2w$ contains more $b$s than
cs, which contradicts the conditions of the Pumping Lemma. So the language can not be
regular.

c The language is regular because it is the complement of the regular language in 3a and
regular languages are closed under complement.