Consider the following recursively defined function on strings:

\[
\text{stutter}(w) := \begin{cases} 
\epsilon & \text{if } w = \epsilon \\
 a a \cdot \text{stutter}(x) & \text{if } w = ax \text{ for some symbol } a \text{ and some string } x
\end{cases}
\]

Intuitively, \( \text{stutter}(w) \) doubles every symbol in \( w \). For example:

- \( \text{stutter}(\text{PRESTO}) = \text{PPRREESSTTOO} \)
- \( \text{stutter}(\text{HOCUS\textbulletPOCUS}) = \text{HHOOCCUUSS\textbullet\textbulletPP00CCUUSS} \)

Let \( L \) be an arbitrary regular language.

1. Prove that the language \( \text{Unstutter}(L) := \{ w \mid \text{stutter}(w) \in L \} \) is regular.

**Solution:** Let \( M = (\Sigma, Q, s, A, \delta) \) be an arbitrary DFA that accepts \( L \).

We construct an DFA \( M' = (\Sigma, Q', s', A', \delta') \) that accepts \( \text{stutter}^{-1}(L) \) as follows:

\[
Q' = Q \\
s' = s \\
A' = A \\
\delta'(q, a) = \delta(\delta(q, a), a)
\]

\( M' \) reads its input string \( w \) and simulates \( M \) running on \( \text{stutter}(w) \). Each time \( M' \) reads a symbol, it passes two copies of that symbol to the simulation of \( M \).  

■
2. Prove that the language \( \text{Stutter}(L) := \{ \text{stutter}(w) \mid w \in L \} \) is regular.

**Solution:** Let \( M = (\Sigma, Q, s, A, \delta) \) be a DFA that accepts \( L \).

We construct an DFA \( M' = (\Sigma, Q', s', A', \delta') \) that accepts \( \text{stutter}(L) \) as follows:

\[
\begin{align*}
Q' &= Q \times (\{\bullet\} \cup \Sigma) \cup \{\text{fail}\} \quad \text{for some new symbol } \bullet \notin \Sigma \\
s' &= (s, \bullet) \\
A' &= \{(q, \bullet) \mid q \in A\} \\
\delta'((q, \bullet), a) &= (q, a) \quad \text{for all } q \in Q \text{ and } a \in \Sigma \\
\delta'((q, a), b) &= \begin{cases} 
(\delta(q, a), \bullet) & \text{if } a = b \\
\text{fail} & \text{if } a \neq b
\end{cases} \quad \text{for all } q \in Q \text{ and } a, b \in \Sigma \\
\delta'(\text{fail}, a) &= \text{fail} \quad \text{for all } a \in \Sigma
\end{align*}
\]

\( M' \) reads the input string \( \text{stutter}(w) \) and simulates \( M \) running on input \( w \).

- State \((q, \bullet)\) means \( M' \) has just read an even-indexed\(^a\) symbol in \( \text{stutter}(w) \), so \( M \) should ignore the next symbol (if any).
- For any symbol \( a \in \Sigma \), state \((q, a)\) means \( M' \) has just read an odd-indexed symbol in \( \text{stutter}(w) \), and that symbol was \( a \). If the next symbol is an \( a \), then \( M \) should transition normally; otherwise, the simulation should fail.
- The state \( \text{fail} \) means \( M' \) has read two successive symbols that should have been equal but were not; the input string is not \( \text{stutter}(w) \) for any string \( w \).

\(^a\)The first symbol in the input string has index 1; the second symbol has index 2, and so on.
Solution (via regular expressions): Let $R$ be an arbitrary regular expression. We recursively construct a regular expression $\text{stutter}(R)$ as follows:

$$
\text{stutter}(R) := \begin{cases} 
\emptyset & \text{if } R = \emptyset \\
\text{stutter}(w) & \text{if } R = w \text{ for some string } w \in \Sigma^* \\
\text{stutter}(A) + \text{stutter}(B) & \text{if } R = A + B \text{ for some regexen } A \text{ and } B \\
\text{stutter}(A) \cdot \text{stutter}(B) & \text{if } R = A \cdot B \text{ for some regexen } A \text{ and } B \\
(\text{stutter}(A))^* & \text{if } R = A^* \text{ for some regex } A
\end{cases}
$$

To prove that $L(\text{stutter}(R)) = \text{stutter}(L(R))$, we need the following identities for arbitrary languages $A$ and $B$:

- $\text{stutter}(A \cup B) = \text{stutter}(A) \cup \text{stutter}(B)$
- $\text{stutter}(A \cdot B) = \text{stutter}(A) \cdot \text{stutter}(B)$
- $\text{stutter}(A^*) = (\text{stutter}(A))^*$

These identities can all be proved by inductive definition-chasing, after which the claim $L(\text{stutter}(R)) = \text{stutter}(L(R))$ follows by induction. We leave the details of the induction proofs as an exercise for a future semester an exam the reader.

Equivalently, we can directly transform $R$ into $\text{stutter}(R)$ by replacing every explicit string $w \in \Sigma^*$ inside $R$ with $\text{stutter}(w)$ (with additional parentheses if necessary). For example:

$$
\text{stutter}\left((1 + \varepsilon)(01)^*(0 + \varepsilon) + 0^*\right) = (11 + \varepsilon)(0011)^*(00 + \varepsilon) + (00)^*
$$

Although this may look simpler, actually proving that it works requires the same induction arguments.

■
3. Let $L$ be an arbitrary regular language.

(a) Prove that the language $\text{INSERT}1(L) := \{xy \mid xy \in L\}$ is regular.

Intuitively, $\text{INSERT}1(L)$ is the set of all strings that can be obtained from strings in $L$ by inserting exactly one 1. For example:

$$\text{INSERT}1(\{\epsilon, 00, 101101\}) = \{1, 100, 010, 001, 1101101, 1011101, 1011011\}$$

**Solution:** Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts $L$. We construct an NFA $M' = (\Sigma, Q', s', A', \delta')$ that accepts $\text{INSERT}1(L)$ as follows:

$$Q' := Q \times \{\text{before, after}\}$$

$$s' := (s, \text{before})$$

$$A' := \{(q, \text{after}) \mid q \in A\}$$

$$\delta'((q, \text{before}), a) = \begin{cases} 
\{(\delta(q, a), \text{before}), (q, \text{after})\} & \text{if } a = 1 \\
\{(\delta(q, a), \text{before})\} & \text{otherwise}
\end{cases}$$

$$\delta'((q, \text{after}), a) = \{(\delta(q, a), \text{after})\}$$

$M'$ nondeterministically chooses one 1 in the input string to ignore, and simulates $M$ running on the rest of the input string.

- The state $(q, \text{before})$ means (the simulation of) $M$ is in state $q$ and $M'$ has not yet skipped over a 1.
- The state $(q, \text{after})$ means (the simulation of) $M$ is in state $q$ and $M'$ has already skipped over a 1.  

\[\blacksquare\]
(b) Prove that the language $\text{DELETE}_1(L) := \{xy \mid x1y \in L\}$ is regular.

Intuitively, $\text{DELETE}_1(L)$ is the set of all strings that can be obtained from strings in $L$ by deleting exactly one $1$. For example:

$$\text{DELETE}_1(\{\epsilon, 00, 101101\}) = \{01101, 10101, 10110\}$$

Solution: Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts $L$. We construct an NFA $M' = (\Sigma, Q', s', A', \delta')$ with $\epsilon$-transitions that accepts $\text{delete}_1(L)$ as follows:

- $Q' := Q \times \{\text{before, after}\}$
- $s' := (s, \text{before})$
- $A' := \{(q, \text{after}) \mid q \in A\}$
- $\delta'(\{(q, \text{before}), \epsilon\}) = \{(\delta(q, 1), \text{after})\}$
- $\delta'(\{(q, \text{after}), \epsilon\}) = \emptyset$
- $\delta'(\{(q, \text{before}), a\}) = \{(\delta(q, a), \text{before})\}$
- $\delta'(\{(q, \text{after}), a\}) = \{(\delta(q, a), \text{after})\}$

$M'$ simulates $M$, but inserts a single $1$ into $M$’s input string at a nondeterministically chosen location.

- State $(q, \text{before})$ means (the simulation of) $M$ is in state $q$ and $M'$ has not yet inserted a $1$.
- State $(q, \text{after})$ means (the simulation of) $M$ is in state $q$ and $M'$ has already inserted a $1$. 

$\blacksquare$
4. Consider the following recursively defined function on strings:

\[
\text{evens}(w) := \begin{cases} 
\epsilon & \text{if } w = \epsilon \\
\epsilon & \text{if } w = \mathit{a} \text{ for some symbol } a \\
b \cdot \text{evens}(x) & \text{if } w = \mathit{abx} \text{ for some symbols } \mathit{a} \text{ and } \mathit{b} \text{ and some string } x
\end{cases}
\]

Intuitively, \text{evens}(w) skips over every other symbol in \( w \). For example:

- \text{evens}('EXPELLIARMUS') = 'XELAMS'
- \text{evens}('AVADA\text{"}KEDAVRA') = 'VD\text{"}EAR'.

Once again, let \( L \) be an arbitrary regular language.

(a) Prove that the language \( \text{Unevens}(L) := \{ w \mid \text{evens}(w) \in L \} \) is regular.

\text{Solution:} Let \( M = (\Sigma, Q, s, A, \delta) \) be a DFA that accepts \( L \). We construct a DFA \( M' = (\Sigma, Q', s', A', \delta') \) that accepts \( \text{evens}^{-1}(L) \) as follows:

\[
\begin{align*}
Q' &= Q \times \{0, 1\} \\
s' &= (s, 0) \\
A' &= A \times \{0, 1\} \\
\delta'((q, 0), a) &= (q, 1) \\
\delta'((q, 1), a) &= (\delta(q, a), 0)
\end{align*}
\]

\( M' \) reads its input string \( w \) and simulates \( M \) running on \( \text{evens}(w) \).

- State \((q, 0)\) means \( M \) is in state \( q \) and \( M' \) has read an even number of symbols, so \( M \) should ignore the next symbol (if any).
- State \((q, 1)\) means \( M \) is in state \( q \) and \( M' \) has read an odd number of symbols, so \( M \) should read the next symbol (if any).
(b) Prove that the language \( \text{Evens}(L) := \{\text{evens}(w) \mid w \in L\} \) is regular.

**Solution:** Let \( M = (\Sigma, Q, s, A, \delta) \) be a DFA that accepts \( L \). We construct an NFA \( M' = (\Sigma, Q', s', A', \delta') \) that accepts \( \text{evens}(L) \) as follows:

\[
\begin{align*}
Q' &= Q \\
s' &= s \\
A' &= A \cup \{q \in Q \mid \delta(q, a) \cap A \neq \emptyset \text{ for some } a \in \Sigma\} \\
\delta'(q, a) &= \bigcup_{b \in \Sigma} \{\delta(q, b), a\}
\end{align*}
\]

\( M' \) reads the input string \( \text{evens}(w) \) and simulates \( M \) running on string \( w \), while nondeterministically guessing the missing symbols in \( w \).

- When \( M' \) reads the symbol \( a \) from \( \text{evens}(w) \), it guesses a symbol \( b \in \Sigma \) and simulates \( M \) reading \( ba \) from \( w \).
- When \( M' \) finishes \( \text{evens}(w) \), it guesses whether \( w \) has even or odd length, and in the odd case, it guesses the last symbol in \( w \).