Let L be an arbitrary regular language over the alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$. Prove that the following languages are also regular. (You probably won't get to all of these.)

1. $\textsc{FlipOdds}(L) := \{\mathit{flipOdds}(w) \mid w \in L\}$, where the function *flipOdds* inverts every odd-indexed bit in w. For example:

$$\mathit{flipOdds}(\mathtt{0000111101010101}) = \mathtt{1010010111111111}$$

Solution: Let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct a new DFA $M' = (Q', s', A', \delta')$ that accepts $\textsc{FlipOdds}(L)$ as follows.

Intuitively, M' receives some string $\mathit{flipOdds}(w)$ as input, restores every other bit to obtain w, and simulates M on the restored string w.

Each state (q, flip) of M' indicates that M is in state q, and we need to flip the next input bit if $\mathit{flip} = \textsc{True}$.

$$\begin{aligned}
Q' &= Q \times \{\textsc{True}, \textsc{False}\}\\
s' &= (s, \textsc{True})\\
A' &= A \times \{\textsc{True}, \textsc{False}\}\\
\delta'((q, \textsc{False}), \mathtt{0}) &= \big(\delta(q, \mathtt{0}), \textsc{True}\big)\\
\delta'((q, \textsc{True}), \mathtt{0}) &= \big(\delta(q, \mathtt{1}), \textsc{False}\big)\\
\delta'((q, \textsc{False}), \mathtt{1}) &= \big(\delta(q, \mathtt{1}), \textsc{True}\big)\\
\delta'((q, \textsc{True}), \mathtt{1}) &= \big(\delta(q, \mathtt{0}), \textsc{False}\big)
\end{aligned}$$

By treating $\mathtt{1}$ and $\mathtt{0}$ as synonyms for $\textsc{True}$ and $\textsc{False}$, respectively, we can rewrite δ' more compactly as

$$\delta'((q, \mathit{flip}), a) = \big(\delta(q, a \oplus \mathit{flip}), \ \neg \mathit{flip}\big)$$

■

2. $\textsc{UnflipOdd1s}(L) := \{w \in \Sigma^* \mid \mathit{flipOdd1s}(w) \in L\}$, where the function *flipOdd1* inverts every other 1 bit of its input string, starting with the first 1. For example:

$$\mathit{flipOdd1s}(0000\underline{1}1\underline{1}10\underline{1}010\underline{1}01) = 0000\underline{0}1\underline{0}10\underline{0}010\underline{0}01$$

Solution: Let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct a new DFA $M' = (Q', s', A', \delta')$ that accepts $\textsc{UnflipOdd1s}(L)$ as follows.

Intuitively, M' receives some string w as input, flips every other 1 bit, and simulates M on the transformed string.

Each state (q, flip) of M' indicates that M is in state q, and we need to flip the next 1 bit of and only if $\mathit{flip} = \textsc{True}$.

$$\begin{aligned}
Q' &= Q \times \{\textsc{True}, \textsc{False}\}\\
s' &= (s, \textsc{True})\\
A' &= A \times \{\textsc{True}, \textsc{False}\}\\
\delta'((q, \textsc{False}), 0) &= \big(\delta(q, 0), \textsc{False}\big)\\
\delta'((q, \textsc{True}), 0) &= \big(\delta(q, 0), \textsc{True}\big)\\
\delta'((q, \textsc{False}), 1) &= \big(\delta(q, 1), \textsc{True}\big)\\
\delta'((q, \textsc{True}), 1) &= \big(\delta(q, 0), \textsc{False}\big)
\end{aligned}$$

Once again, by treating 1 and 0 as synonyms for $\textsc{True}$ and $\textsc{False}$, respectively, we can rewrite δ' more compactly as

$$\delta'((q, \mathit{flip}), a) = \big(\delta(q, \neg \mathit{flip} \wedge a),\ \mathit{flip} \oplus a\big) \qquad \blacksquare$$

3. $\textsc{FlipOdd1s}(L) := \{flipOdd1s(w) \mid w \in L\}$, where the function *flipOdd1* is defined as in the previous problem.

Solution: Let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct a new **NFA** $M' = (Q', s', A', \delta')$ that accepts $\textsc{FlipOdd1s}(L)$ as follows.

Intuitively, M' receives some string $flipOdd1s(w)$ as input, ***guesses*** which 0 bits to restore to 1s, and simulates M on the restored string w. No string in $\textsc{FlipOdd1s}(L)$ has two 1s in a row, so if M' ever sees 11, it rejects.

Each state $(q, flip)$ of M' indicates that M is in state q, and we need to flip a 0 bit before the next 1 bit if and only if $flip = \textsc{True}$.

$$
\begin{aligned}
Q' &= Q \times \{\textsc{True}, \textsc{False}\} \\
s' &= (s, \textsc{True}) \\
A' &= A \times \{\textsc{True}, \textsc{False}\} \\
\delta'((q, \textsc{False}), 0) &= \big\{\big(\delta(q, 0),\ \textsc{False}\big)\big\} \\
\delta'((q, \textsc{True}), 0) &= \big\{\big(\delta(q, 0),\ \textsc{True}\big),\ \big(\delta(q, 1),\ \textsc{False}\big)\big\} \\
\delta'((q, \textsc{False}), 1) &= \big\{\big(\delta(q, 1),\ \textsc{True}\big)\big\} \\
\delta'((q, \textsc{True}), 1) &= \varnothing
\end{aligned}
$$

The last transition indicates that we waited too long to flip a 0 to a 1, so we should kill the current execution thread. ■

4. Prove that the language $insert1(L) := \{x1y \mid xy \in L\}$ is regular.

Intuitively, $insert1(L)$ is the set of all strings that can be obtained from strings in L by inserting exactly one 1. For example, if $L = \{\varepsilon, \texttt{OOK!}\}$, then $insert1(L) = \{\texttt{1}, \texttt{1OOK!}, \texttt{O1OK!}, \texttt{OO1K!}, \texttt{OOK1!}, \texttt{OOK!1}\}$.

Solution: Let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct an NFA $M' = (Q', s', A', \delta')$ that accepts $insert1(L)$ as follows.

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

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

$$
\begin{aligned}
Q' &:= Q \times \{before, after\} \\
s' &:= (s, before) \\
A' &:= \big\{(q, after) \bigm| q \in A\big\} \\
\delta'((q, before), a) &= \begin{cases} \big\{(\delta(q,a), before),\ (q, after)\big\} & \text{if } a = 1 \\ \big\{(\delta(q,a), before)\big\} & \text{otherwise} \end{cases} \\
\delta'((q, after), a) &= \big\{(\delta(q,a), after)\big\}
\end{aligned}
$$

■

5. Prove that the language $\mathit{delete}\mathtt{1}(L) := \{xy \mid x\mathtt{1}y \in L\}$ is regular.

   Intuitively, $\mathit{delete}\mathtt{1}(L)$ is the set of all strings that can be obtained from strings in L by deleting exactly one $\mathtt{1}$. For example, if $L = \{\mathtt{101101}, \mathtt{00}, \varepsilon\}$, then $\mathit{delete}\mathtt{1}(L) = \{\mathtt{01101}, \mathtt{10101}, \mathtt{10110}\}$.

Solution: Let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct an NFA $M' = (Q', s', A', \delta')$ with ε-transitions that accepts $\mathit{delete}\mathtt{1}(L)$ as follows.

Intuitively, M' simulates M, but inserts a single $\mathtt{1}$ into M's input string at a nondeterministically chosen location.

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

$$\begin{aligned} Q' &:= Q \times \{\mathit{before}, \mathit{after}\} \\ s' &:= (s, \mathit{before}) \\ A' &:= \big\{(q, \mathit{after}) \bigm| q \in A\big\} \end{aligned}$$

$$\begin{aligned} \delta'((q, \mathit{before}), \varepsilon) &= \big\{(\delta(q, \mathtt{1}), \mathit{after})\big\} \\ \delta'((q, \mathit{after}), \varepsilon) &= \varnothing \end{aligned}$$

$$\begin{aligned} \delta'((q, \mathit{before}), a) &= \big\{(\delta(q, a), \mathit{before})\big\} \\ \delta'((q, \mathit{after}), a) &= \big\{(\delta(q, a), \mathit{after})\big\} \end{aligned}$$

■

6. Consider the following recursively defined function on strings:

$$stutter(w) := \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ aa \bullet stutter(x) & \text{if } w = ax \text{ for some symbol } a \text{ and some string } x \end{cases}$$

Intuitively, $stutter(w)$ doubles every symbol in w. For example:

- $stutter$(PRESTO) = PPRREESSTTOO
- $stutter$(HOCUS◇POCUS) = HHOOCCUUSS◇◇PPOOCCUUSS

(a) Prove that the language $stutter^{-1}(L) := \{w \mid stutter(w) \in L\}$ is regular.

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

Intuitively, M' reads its input string w and simulates M running on $stutter(w)$. Each time M' reads a symbol, the simulation of M reads two copies of that symbol.

$$\begin{aligned} Q' &= Q \\ s' &= s \\ A' &= A \\ \delta'(q, a) &= \delta(\delta(q, a), a) \end{aligned}$$

■

(b) Prove that the language $stutter(L) := \{stutter(w) \mid w \in L\}$ is regular.

Solution: Let $M = (Q, s, A, \delta)$ be a DFA that accepts L. wWe construct an DFA $M' = (Q', s', A', \delta')$ that accepts $stutter(L)$ as follows.

M' reads the input string $stutter(w)$ and simulates M running on input w.

- State $(q, \bullet)$ means M' has just read an even-indexed[a] symbol in $stutter(w)$, 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 $stutter(w)$, and that symbol was a. If the next symbol is an a, then M should transition normally; otherwise, the simulation should fail.
- The state *fail* means M' has read two successive symbols that should have been equal but were not; the input string is not $stutter(w)$ for any string w.

$$Q' = Q \times (\{\bullet\} \cup \Sigma) \cup \{fail\} \qquad \text{for some new symbol } \bullet \notin \Sigma$$
$$s' = (s, \bullet)$$
$$A' = \{(q, \bullet) \mid q \in A\}$$

$$\delta'((q, \bullet), a) = (q, a) \qquad \text{for all } q \in Q \text{ and } a \in \Sigma$$
$$\delta'((q, a), b) = \begin{cases} (\delta(q, a), \bullet) & \text{if } a = b \\ fail & \text{if } a \neq b \end{cases} \qquad \text{for all } q \in Q \text{ and } a, b \in \Sigma$$
$$\delta'(fail, a) = fail \qquad \text{for all } a \in \Sigma$$ ■

[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 $stutter(R)$ as follows:

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

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

- $stutter(A \cup B) = stutter(A) \cup stutter(B)$
- $stutter(A \bullet B) = stutter(A) \bullet stutter(B)$
- $stutter(A^*) = (stutter(A))^*$

These identities can all be proved by inductive definition-chasing, after which the claim $L(stutter(R)) = stutter(L(R))$ 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 $stutter(R)$ by replacing every explicit string $w \in \Sigma^*$ inside R with $stutter(w)$ (with additional parentheses if necessary). For example:

$$stutter\big((\mathtt{1} + \varepsilon)(\mathtt{01})^*(\mathtt{0} + \varepsilon) + \mathtt{0}^*\big) \;=\; (\mathtt{11} + \varepsilon)(\mathtt{0011})^*(\mathtt{00} + \varepsilon) + (\mathtt{00})^*$$

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

7. Consider the following recursively defined function on strings:

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

Intuitively, $evens(w)$ skips over every other symbol in w. For example:

- $evens(\texttt{EXPELLIARMUS}) = \texttt{XELAMS}$
- $evens(\texttt{AVADA}\diamond\texttt{KEDAVRA}) = \texttt{VD}\diamond\texttt{EAR}$.

Once again, let L be an arbitrary regular language.

(a) Prove that the language $evens^{-1}(L) := \{w \mid evens(w) \in L\}$ is regular.

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

$$\begin{aligned} Q' &= Q \times \{0, 1\} \\ s' &= (s, 0) \\ A' &= A \times \{0, 1\} \end{aligned}$$

$$\begin{aligned} \delta'((q, 0), a) &= (q, 1) \\ \delta'((q, 1), a) &= (\delta(q, a), 0) \end{aligned}$$

M' reads its input string w and simulates M running on $evens(w)$.

- State $(q, 0)$ means M' has just read an even symbol in w, so M should ignore the next symbol (if any).
- State $(q, 1)$ means M' has just read an odd symbol in w, so M should read the next symbol (if any).

■

(b) Prove that the language $evens(L) := \{evens(w) \mid w \in L\}$ is regular.

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

Intuitively, M' reads the input string $evens(w)$ and simulates M running on string w, while nondeterministically guessing the missing symbols in w.

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

$$
\begin{aligned}
Q' &= Q \\
s' &= s \\
A' &= A \cup \big\{q \in Q \;\big|\; \delta(q,a) \cap A \neq \varnothing \text{ for some } a \in \Sigma\big\} \\
\delta'(q,a) &= \bigcup_{b \in \Sigma} \big\{\delta\big(\delta(q,b),a\big)\big\}
\end{aligned}
$$

■