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:

- $stutter(\texttt{PRESTO}) = \texttt{PPRREESSTTOO}$
- $stutter(\texttt{HOCUS}\diamond\texttt{POCUS}) = \texttt{HHOOCCUUSS}\diamond\diamond\texttt{PPOOCCUUSS}$

Let L be an arbitrary regular language.

1. Prove that the language $stutter^{-1}(L) := \{w \mid stutter(w) \in L\}$ 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 $stutter^{-1}(L)$ 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 $stutter(w)$. Each time M' reads a symbol, the simulation of M reads two copies of that symbol. ■

2. Prove that the language $stutter(L) := \{stutter(w) \mid w \in L\}$ 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 $stutter(L)$ as follows:

$$\begin{aligned} Q' &= Q \times (\{\bullet\} \cup \Sigma) \cup \{fail\} \quad \text{for some } \bullet \notin \Sigma \\ s' &= (s, \bullet) \\ A' &= \{(q, \bullet) \mid q \in A\} \end{aligned}$$

$$\begin{aligned} \delta'((q,\bullet), a) &= (q, a) \\ \delta'((q, a), b) &= \begin{cases} (\delta(q, a), \bullet) & \text{if } a = b \\ fail & \text{if } a \neq b \end{cases} \\ \delta'(fail, a) &= fail \end{aligned}$$

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.

■

[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:

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

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 $\mathit{insert1}(L) := \{x\mathtt{1}y \mid xy \in L\}$ is regular.

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

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 $\mathit{insert1}(L)$ as follows:

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

M' nondeterministically chooses a $\mathtt{1}$ 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 $\mathtt{1}$.
- The state (q, after) means (the simulation of) M is in state q and M' has already skipped over a $\mathtt{1}$. ■

(b) Prove that the language $delete1(L) := \{xy \mid x1y \in L\}$ is regular.

Intuitively, $delete1(L)$ is the set of all strings that can be obtained from strings in L by deleting exactly one 1. For example, if $L = \{101101, 00, \varepsilon\}$, then $delete1(L) = \{01101, 10101, 10110\}$.

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 ε-transitions that accepts $delete1(L)$ as follows:

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

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

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

M' simulates M, but inserts a single 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 1.
- The state $(q, after)$ means (the simulation of) M is in state q and M' has already inserted a 1. ■

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

$$Q' = Q \times \{0, 1\}$$
$$s' = (s, 0)$$
$$A' = A \times \{0, 1\}$$

$$\delta'((q, 0), a) = (q, 1)$$
$$\delta'((q, 1), a) = (\delta(q, a), 0)$$

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

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

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.

■