1 Let L be an arbitrary regular language. Prove that the language $\mathit{reverse}(L) := \left\{ w^R \mid w \in L \right\}$ is regular. *Hint:* Consider a DFA M that accepts L and construct a NFA that accepts $\mathit{reverse}(L)$.

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 $\mathit{reverse}(L)$ as follows.

$$\begin{aligned} Q' &:= Q \cup \{t\} \quad \text{(here } t \text{ is a new state not in } Q\text{)} \\ s' &:= t \\ A' &:= \{s\} \\ \delta'(t, \epsilon) &= A \\ \forall q \in Q, a \in \Sigma \quad \delta'(q, a) &= \left\{ q' \in Q \mid \delta(q', a) = q \right\} \end{aligned}$$

M' is obtained from M by reversing all the directions of the edges, adding a new state t that becomes the new start state that is connected via ϵ edges to all the original accepting states. There is a single accepting state in M' which is the start state of M. To see that M' accepts $\mathit{reverse}(L)$ you need to see that any accepting walk of M' corresponds to an accepting walk of M.

Another way to show that $\mathit{reverse}(L)$ is regular is via regular expressions. For any regular expression r you can construct a regular expression r' such that $L(r') = \mathit{reverse}(L)$ using the inductive definition of regular languages. We ignore the base cases as exercise and consider the inductive cases.

- If r_1 and r_2 are regular expressions and r_1' and r_2' are regular expressions for the reverse languages then the reverse for $r_1 + r_2$ is $r_1' + r_2'$.
- For $r_1 r_2$ we have $r_2' r_1'$.
- For $(r_1)^*$ we have $(r_1')^*$.

---

2 Let L be an arbitrary regular language. Prove that the language $\mathit{insert}1(L) := \{x1y \mid xy \in L\}$ is regular.

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

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 $\mathit{insert}1(L)$ as follows:

$$\begin{aligned} Q' &:= Q \times \{\mathit{before}, \mathit{after}\} \\ s' &:= (s, \mathit{before}) \\ A' &:= \left\{ (q, \mathit{after}) \mid q \in A \right\} \end{aligned}$$

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

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

M' nondeterministically chooses a 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.

Solutions for extra problems

---

3 Let L be an arbitrary regular language. Prove that the language $\mathit{delete}1(L) := \{xy \mid x1y \in L\}$ is regular.

Intuitively, $\mathit{delete}1(L)$ 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 $\mathit{delete}1(L) = \{01101, 10101, 10110\}$.

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 $\mathit{delete}1(L)$ as follows:

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

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

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

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:

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

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

- $\mathit{stutter}(PRESTO) = PPRREESSTTOO$
- $\mathit{stutter}(HOCUS\square POCUS) = HHOOCCUUSS\square\square PPOOCCUUSS$

Let L be an arbitrary regular language.

4.A. Prove that the language $\mathit{stutter}^{-1}(L) := \{w \mid \mathit{stutter}(w) \in L\}$ 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 $\mathit{stutter}^{-1}(L)$ as follows:

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

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

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

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

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

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

- State $(q,\bullet)$ means M' has just read an even symbol in $\mathit{stutter}(w)$, 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 symbol in $\mathit{stutter}(w)$, 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 $\mathit{stutter}(w)$ for any string w.

Solution:

Let R be an arbitrary regular *expression*. We recursively construct a regular expression $\mathit{stutter}(R)$ as follows:

$$\mathit{stutter}(R) := \begin{cases} \varnothing & \text{if } R = \varnothing \\ \mathit{stutter}(w) & \text{if } R = w \text{ for some string } w \in \Sigma^* \\ \mathit{stutter}(A) + \mathit{stutter}(B) & \text{if } R = A + B \text{ for some regular expressions } A \text{ and } B \\ \mathit{stutter}(A)\,\mathit{stutter}(B) & \text{if } R = A\,B \text{ for some regular expressions } A \text{ and } B \\ (\mathit{stutter}(A))^* & \text{if } R = A^* \text{ for some regular expression } A \end{cases}$$

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

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

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

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

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

---

5 Consider the following recursively defined function on strings:

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

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

- $\mathit{evens}(\mathit{EXPELLIARMUS}) = \mathit{XELAMS}$
- $\mathit{evens}(\mathit{AVADA}\text{␣}\mathit{KEDAVRA}) = \mathit{VD}\text{␣}\mathit{EAR}$.

Once again, let L be an arbitrary regular language.

5.A. Prove that the language $\mathit{evens}^{-1}(L) := \{w \mid \mathit{evens}(w) \in L\}$ 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 $evens^{-1}(L)$ 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).

5.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:

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

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 character of w.