

Let L be an arbitrary regular language over the alphabet $\Sigma = \{0, 1\}$. Prove that the following languages are also regular. For each language, we provide a high level intuitive sketch of the proof; your task is to fill in the remaining technical details. (You probably won't get to all of these during the lab session.)

Important note about notation: If a language L is described in set-builder notation as $\{stuff \mid condition\}$, then every finite-state machine that accepts L takes *stuff* as input, and checks whether *stuff* satisfies the stated *condition*.

In particular, every finite-state machine that accepts the language $\{foo(w) \mid bar(w) \in L\}$ takes an **arbitrary string** x as input, *guesses* a string w such that $x = foo(w)$, and determines whether the guessed string w satisfies the condition $bar(w) \in L$.

1. Let $INSERTEDANY1S(L)$ is the set of all strings that can be obtained from strings in L by inserting **any number of 1s** anywhere in the string. For example:

$$INSERTEDANY1S(\{\epsilon, 1, 00\}) = \{\epsilon, 1, 11, 111, \dots, 00, 100, 0111110, 111011111101111, \dots\}$$

Prove that the language $INSERTEDANY1S(L)$ is regular.

Solution: Let $M = (Q, s, A, \delta)$ be an arbitrary DFA that accepts the regular language L . We want to build a machine M' that accepts $INSERTEDANY1S(L)$.

We construct a new **NFA with ϵ -transitions** $M' = (Q', s', A', \delta')$ that accepts $INSERTEDANY1S(L)$ as follows. **The input to M' is the result of inserting 1s into some string w , and M' needs to determine whether that string w is in L .** Intuitively, M' *guesses* the original string w , by nondeterministically guessing which 1s were inserted, and simulates M running on that original string w .

M' has the same states and start state and accepting states as M , but it has a different transition function.

$$Q' = Q$$

$$s' = s$$

$$A' = A$$

$$\delta'(q, 0) = \{ \delta(q, 0) \}$$

$$\delta'(q, 1) = \{ \boxed{q, \delta(q, 1)} \}$$

$$\delta'(q, \epsilon) = \{ \boxed{} \}$$

Yes, the last box is empty. Each time M' reads a 1, M' guesses whether to ignore that 1 (because it was inserted) or to pass that 1 to M (because it was a symbol in w). In hindsight, we don't need any ϵ -transitions, so we can safely delete the final transition $\delta'(q, \epsilon) = \emptyset$ and the phrase “with ϵ -transitions” at the beginning. ■

2. Let $\text{DELETEDANY1S}(L)$ is the set of all strings that can be obtained from strings in L by deleting **any number of 1s** anywhere in the string. For example:

$$\text{DELETEDANY1S}(\{\epsilon, 00, 1101\}) = \{\epsilon, 0, 00, 01, 10, 101, 110, 1101\}$$

Prove that the language $\text{DELETEDANY1S}(L)$ is regular.

Solution: Let $M = (Q, s, A, \delta)$ be an arbitrary DFA that accepts the regular language L .

We construct a new **NFA with ϵ -transitions** $M' = (Q', s', A', \delta')$ that accepts $\text{DELETEDANY1S}(L)$ as follows. **The input to M' is the result of deleting 1s from some string w , and M' needs to determine whether that string w is in L . Intuitively, M' guesses the original string w , by nondeterministically guessing where 1s were deleted, and simulates M running on that original string w .**

M' has the same states and start state and accepting states as M , but a different transition function.

$$Q' = Q$$

$$s' = s$$

$$A' = A$$

$$\delta'(q, 0) = \{ \delta(q, 0) \}$$

$$\delta'(q, 1) = \{ \boxed{\delta(q, 1)} \}$$

$$\delta'(q, \epsilon) = \{ \boxed{\delta(q, 1)} \}$$

M' passes each of its input symbols to M . Between input symbols M' uses ϵ -transitions to guess where 1s have been deleted and passes those deleted 1s to M . ■

3. Let $\text{INSERTEDONE1}(L) := \{x1y \mid xy \in L\}$ denote the set of all strings that can be obtained from strings in L by inserting **exactly one 1**. For example:

$$\text{INSERTEDONE1}(\{\epsilon, 00, 101101\}) = \{1, 100, 010, 001, 1101101, 1011101, 1011011\}$$

Prove that the language $\text{INSERTEDONE1}(L)$ is regular.

Solution: Let $M = (Q, s, A, \delta)$ be an arbitrary DFA that accepts the regular language L .

We construct a new **NFA with ϵ -transitions** $M' = (Q', s', A', \delta')$ that accepts $\text{INSERTEDONE1}(L)$ as follows. The input to M' is the *result* $x1y$ of inserting exactly one 1 into some string xy , and M' needs to determine whether xy is in L . If the input string for M' does not contain a 1, then M' rejects it as invalid. Otherwise, intuitively, M' *guesses* which 1 in its input string was inserted, and simulates M on the rest of the input string.

M' consists of two copies of M , one to process the prefix x and the other to process the suffix y . State (q, FALSE) means (the simulation of) M is in state q and M' has not yet skipped over a 1. State (q, TRUE) means (the simulation of) M is in state q and M' has already skipped over a 1.

$$Q' = Q \times \{\text{TRUE}, \text{FALSE}\}$$

$$s' = (s, \text{FALSE})$$

$$A' = \boxed{A \times \{\text{TRUE}\} = \{(q, \text{TRUE}) \mid q \in A\}}$$

$$\delta'((q, \text{FALSE}), 0) = \{(\delta(q, 0), \text{FALSE})\}$$

$$\delta'((q, \text{FALSE}), 1) = \{(\delta(q, 1), \text{FALSE}), (q, \text{TRUE})\}$$

$$\delta'((q, \text{FALSE}), \epsilon) = \{\boxed{}\}$$

$$\delta'((q, \text{TRUE}), 0) = \{(\delta(q, 0), \text{TRUE})\}$$

$$\delta'((q, \text{TRUE}), 1) = \{(\delta(q, 1), \text{TRUE})\}$$

$$\delta'((q, \text{TRUE}), \epsilon) = \{\boxed{}\}$$

Whenever M' reads a 1 while its “skipped” flag is FALSE, M' guesses whether to ignore that 1 and sets the “skipped” flag to TRUE (because that 1 was inserted into M ’s input string) or to pass it to the simulation of M . Otherwise, M' passes all input symbols directly to M . Finally, M' accepts if and only if its simulation of M accepts and M' has skipped a 1.

In hindsight, we don’t need ϵ -transitions, so we can safely delete those transitions from our solution, along with the phrase “with ϵ -transitions” at the beginning. ■

4. Let $\text{DELETEDONE1}(L) := \{xy \mid x1y \in L\}$ denote the set of all strings that can be obtained from strings in L by deleting exactly one 1. For example:

$$\text{DELETEDONE1}(\{\varepsilon, 00, 101101\}) = \{01101, 10101, 10110\}$$

Prove that the language $\text{DELETEDONE1}(L)$ is regular.

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

We construct an *NFA with ε -transitions* $M' = (\Sigma, Q', s', A', \delta')$ that accepts $\text{DELETEDONE1}(L)$ as follows. The input to M' is the result of deleting one 1 from some string w , and M' needs to determine whether that string w is in L . Intuitively, M' guesses the original string w , by nondeterministically guessing where the 1 was deleted, and simulates M running on that original string w . Equivalently, M' simulates the original DFA M on the prefix x before the deleted 1, then the deleted 1 (which is not part of M' 's input), and finally the suffix y after the deleted 1.

M' consists of two copies of M , one to process the prefix x and the other to process the suffix y . State (q, FALSE) means (the simulation of) M is in state q and M' has not yet reinserted a 1. State (q, TRUE) means (the simulation of) M is in state q and M' has already reinserted a 1.

$$Q' = Q \times \{\text{TRUE}, \text{FALSE}\}$$

$$s' = (s, \text{FALSE})$$

$$A' = A \times \{\text{TRUE}\} = \{(q, \text{TRUE}) \mid q \in A\}$$

$$\delta'((q, \text{FALSE}), 0) = \{(\delta(q, 0), \text{FALSE})\}$$

$$\delta'((q, \text{FALSE}), 1) = \{(\delta(q, 1), \text{FALSE})\}$$

$$\delta'((q, \text{FALSE}), \varepsilon) = \{(\delta(q, 1), \text{TRUE})\}$$

$$\delta'((q, \text{TRUE}), 0) = \{(\delta(q, 0), \text{TRUE})\}$$

$$\delta'((q, \text{TRUE}), 1) = \{(\delta(q, 1), \text{TRUE})\}$$

$$\delta'((q, \text{TRUE}), \varepsilon) = \{\}$$

M' passes each of its input symbols to M . Between input symbols, M' guesses where the single 1 has been deleted and passes that deleted 1 to M using an ε -transition. Finally, M' accepts if and only if its simulation of M accepts and M' has identified the deleted 1. ■

Work on these later: Consider the following recursively defined function on strings:

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

Intuitively, $\text{evens}(w)$ skips over every other symbol in w , starting with the first symbol. For example, $\text{evens}(\text{THE} \diamond \text{SNAIL}) = \text{H} \diamond \text{NI}$ and $\text{evens}(\text{GROB} \diamond \text{GOB} \diamond \text{GLOB} \diamond \text{GROD}) = \text{RBGBGO} \diamond \text{RD}$.

Let L be an arbitrary regular language over the alphabet $\Sigma = \{0, 1\}$.

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

Solution: Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts the regular language L . We need to construct a new machine M' that accepts $\text{UNEVENED}(L)$. The input to M' is an arbitrary string w , and M' needs to determine whether $\text{evens}(w)$ is in L .

We construct a **DFA** $M' = (\Sigma, Q', s', A', \delta')$ that accepts $\text{UNEVENED}(L)$ 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 $\text{evens}(w)$.

- 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).

As usual, I started by trying to construct an NFA with ε -transitions. But when I was finished, I noticed that all my ε -transitions led to $\emptyset$ and that all my real transitions led to a single state, which meant I had actually built a DFA! ■

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

Solution: Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts the regular language L . We need to build a machine M' that accepts $\text{EVENED}(L)$. The input to M' is the output of $\text{evens}(w)$, and M' needs to determine whether the original string w is in L . So intuitively, M' guesses the odd-indexed symbols in w , and then passes the reconstituted string w to M .

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

$$Q' = Q$$

$$s' = s$$

$$A' = A \cup \{q \in Q \mid \delta(q, \textcolor{red}{0}) \in A\} \cup \{q \in Q \mid \delta(q, \textcolor{red}{1}) \in A\}$$

$$\delta'(q, a) = \{\delta(\delta(q, \textcolor{red}{0}), a), \delta(\delta(q, \textcolor{red}{1}), a)\}$$

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 reading $\text{evens}(w)$, it guesses whether w has even or odd length, and in the odd case, it guesses the last symbol in w .

As usual, I started by trying to construct an NFA with ε -transitions. But when I was finished, I noticed that all my ε -transitions led to $\emptyset$, so I erased them. ■