

1. Recall, for a string w , we let w^R denote its reversal.

Let L be an arbitrary regular language over the alphabet $\Sigma = \{0, 1\}$. Prove that the following languages over the alphabet $\Sigma' = \{0, 1, \#\}$ are also regular:

- (a) $\text{BACKTOFRONT}(L) = \{y^R\#x \mid x, y \in \Sigma^* \text{ and } xy \in L\}$.

Solution: Given an arbitrary DFA $M = (Q, s, A, \delta)$ for L , we construct an **NFA** $N = (Q', S', A', \delta')$ with multiple start states for $\text{BACKTOFRONT}(L)$ as follows. All unspecified transitions go to $\emptyset$.

$$Q' = Q \times (Q \cup \{\perp\})$$

$$S' = A \times \{\perp\}$$

$$A' = \{(h, h) \mid h \in Q\}$$

$$\delta'((q, \perp), a) = \{(p, \perp) \mid q = \delta(p, a)\} \quad \text{for all } q \in Q \text{ and } a \in \Sigma$$

$$\delta'((q, \perp), \#) = \{(s, q)\} \quad \text{for all } q \in Q$$

$$\delta'((q, h), a) = \{(\delta(q, a), h)\} \quad \text{for all } q, h \in Q \text{ and } a \in \Sigma$$

Machine M' reads in string y^R by simulating M backwards as done for language reversal in lecture. When M' encounters $\#$, it starts simulating M forward to see if it reaches the same state encountered when going backwards.

Suppose M' reads $y^R\#x$. State $(q, \perp)$ means a suffix of y takes state q in M to an accept state. State (q, h) means a prefix of x takes s to q , and y takes h to an accept state. We start at any $(r, \perp)$ with $r \in A$ to represent xy being accepted at r . We accept at any (h, h) to represent x taking s to h and then y taking h to an accept state. ■

Rubric: 5 points, standard language transformation rubric (scaled). This is not the only correct solution.

(b) $\text{FRONTTOBACK}(L) = \{yx^R \mid x, y \in \Sigma^* \text{ and } xy \in L\}$

Solution: Given an arbitrary DFA $M = (Q, s, A, \delta)$ for L , we construct an **NFA** $N = (Q', S', A', \delta')$ with multiple start states and ϵ -transitions for $\text{FRONTTOBACK}(L)$ as follows. All unspecified transitions go to $\emptyset$.

$$Q' = Q \times (Q \cup \{\perp\})$$

$$S' = \{(h, h) \mid h \in Q\}$$

$$A' = \{(s, \perp)\}$$

$$\delta'((q, h), a) = \{(\delta(q, a), h)\} \quad \text{for all } q, h \in Q \text{ and } a \in \Sigma$$

$$\delta'((r, h), \epsilon) = \{(h, \perp)\} \quad \text{for all } h \in Q \text{ and } r \in A$$

$$\delta'((q, \perp), a) = \{(p, \perp) \mid q = \delta(p, a)\} \quad \text{for all } q \in Q \text{ and } a \in \Sigma$$

Machine M' guesses the state h that machine M reaches after reading x for some $xy \in L$. Machine M' reads in string y by simulating M from h . If it reaches an accept state after reading y , then it reads x^R by simulating M backwards from h as done for language reversal in lecture.

Suppose M' reads yx^R . State (q, h) means a prefix of y takes state h in M to q . State $(q, \perp)$ means a suffix of x takes q to h , and y takes h to an accept state. We start at any (h, h) to represent x taking s to h . We accept at any $(s, \perp)$ to represent x taking s to some h and then y taking h to an accept state. ■

Rubric: 5 points, standard language transformation rubric (scaled). This is not the only correct solution.

2. Describe context-free grammars for the following languages, and clearly explain how they work.

- (a) All strings in which every prefix has at least as many 1s as 0s. More formally:

$$\{w \in \Sigma^* \mid \text{for all } x, y \text{ such that } w = xy, \#(1, x) \geq \#(0, x)\}$$

Solution:

$$S \rightarrow \varepsilon \mid 1S0S \mid 1S \quad \{w \in \Sigma^* \mid \text{for all } x, y \text{ such that } w = xy, \#(1, x) \geq \#(0, x)\}$$

Every string in the language is either 1) empty, 2) has a shortest prefix x with $\#(0, x) = \#(1, x)$, or 3) is non-empty with no such shortest prefix. The production rules distinguish between these three cases. ■

Rubric: 3 points, standard context free grammar rubric (scaled). This is not the only correct solution.

- (b) Even-length strings whose second half contains an even number of 1s. More formally:

$$\{xy \in \Sigma^* \mid |x| = |y| \text{ and } \#(1, y) \text{ is even}\}$$

Solution:

$$\begin{array}{ll} S \rightarrow E & \{xy \in \Sigma^* \mid |x| = |y| \text{ and } \#(1, y) \text{ is even}\} \\ E \rightarrow \varepsilon \mid 0E0 \mid 1E0 \mid 0O1 \mid 1O1 & \{xy \in \Sigma^* \mid |x| = |y| \text{ and } \#(1, y) \text{ is even}\} \\ O \rightarrow 0O0 \mid 1O0 \mid 0E1 \mid 1E1 & \{xy \in \Sigma^* \mid |x| = |y| \text{ and } \#(1, y) \text{ is odd}\} \end{array}$$

The production rules create characters in pairs to keep the total length even. When a 1 appears in the second half, then the parity needed for the remaining second half symbols is flipped. ■

Rubric: 3 points, standard context free grammar rubric (scaled). This is not the only correct solution.

(c) $\{0^a 1^b 0^c 1^d \mid a, b, c, d \geq 1 \text{ and if } a = d, \text{ then } b = c\}$

Solution:

$$S \rightarrow 0A1 \mid 0C1 \quad \{0^a 1^b 0^c 1^d \mid a, b, c, d \geq 1 \text{ and if } a = d, \text{ then } b = c\}$$

$$A \rightarrow 1B0 \mid 0A1 \quad \{0^a 1^b 0^b 1^a \mid b \geq 1\}$$

$$B \rightarrow \varepsilon \mid 1B0 \quad \{1^a 0^a\}$$

$$C \rightarrow D \mid E \quad \{0^a 1^b 0^c 1^d \mid b, c \geq 1 \text{ and } a \neq d\}$$

$$D \rightarrow 0F \mid 0D \quad \{0^a 1^b 0^c 1^d \mid b, c \geq 1 \text{ and } a > d\}$$

$$E \rightarrow F1 \mid E1 \quad \{0^a 1^b 0^c 1^d \mid b, c \geq 1 \text{ and } a < d\}$$

$$F \rightarrow 1G0 \mid 0F1 \quad \{0^a 1^b 0^c 1^a \mid b, c \geq 1\}$$

$$G \rightarrow \varepsilon \mid 1G \mid G0 \quad \{1^a 0^b\}$$

The production rules for S distinguish between strings $0^a 1^b 0^c 1^d$ with $a = d$ and those without. For those strings with $a = d$, the rules for A create the equal number of outside 0s and 1s before the rules for B creates the equal number of inside 0s and 1s. For those strings with $a \neq d$, the rules for C through F make sure $a \neq d$ like we saw in lecture. The rules for G creates the inner 0s and 1s. ■

Rubric: 4 points, standard context free grammar rubric (scaled). This is not the only correct solution.

* (d) Practice only. Do not submit solutions.

Strings in which the substrings 00 and 11 appear the same number of times. For example, $1100011 \in L$ because both substrings appear twice, but $01000011 \notin L$.

Solution (counting): Let $\#(11, w)$ denote the number of times 11 appears as a substring of w , and let $\#(1^+, w)$ denote the number of runs of 1 s in w . For example:

$$\#(11, \underline{1111}00\underline{11101}) = 5 \quad \#(1^*, \underline{1111}00\underline{11101}) = 3$$

Each 1 in a binary string is the beginning of a 11 substring, except for the last 1 in every run; it follows that

$$\#(11, w) = \#(1, w) - \#(1^+, w)$$

for every string w . Symmetric arguments imply $\#(00, w) = \#(0, w) - \#(0^+, w)$ for every string w . Thus

$$\begin{aligned} \#(00, w) &= \#(11, w) \\ &\iff \\ \#(0, w) - \#(0^+, w) &= \#(1, w) - \#(1^+, w) \\ &\iff \\ \#(0, w) - \#(1, w) &= \#(0^+, w) - \#(1^+, w) \end{aligned}$$

But because runs of 0 s and 1 s in any binary string w alternate, the number of 0 -runs and the number of 1 -runs always differ by at most 1. More specifically:

- If w starts and ends with 0 , then $\#(1^+, w) = \#(0^+, w) - 1$.
- If the first and last symbols in w are different, then $\#(1^+, w) = \#(0^+, w)$.
- If w starts and ends with 1 , then $\#(1^+, w) = \#(0^+, w) + 1$.

Let $\Delta(w) = \#(1, w) - \#(0, w)$. The preceding case analysis implies that our target language L contains a binary string w if and only if w satisfies one of the following conditions:

- $w = \varepsilon$
- w starts with 0 and ends with 1 , and $\Delta(w) = 0$
- w starts with 1 and ends with 0 , and $\Delta(w) = 0$
- w starts with 0 and ends with 0 , and $\Delta(w) = -1$. In this case, dropping the final 0 leaves a string with equal 0 s and 1 s. So there are two subcases:
 - $w = 0$
 - w contains at least one 1 . Then w must have a prefix x that starts with 0 and ends with 1 , such that $\Delta(x) = 0$. We can write $w = xy0$ for some (possibly empty) string y with $\Delta(y) = 0$.
- w starts with 1 and ends with 1 , and $\Delta(w) = 1$. This case is symmetric to the previous case.

Finally, we can write down a grammar that follows the preceding case analysis.

$S \rightarrow \varepsilon \mid A \mid B \mid C \mid D$	target language L
$A \rightarrow 0E1$	starts with 0, ends with 1, and $\Delta = 0$
$B \rightarrow 1E0$	starts with 1, ends with 0, and $\Delta = 0$
$C \rightarrow 0 \mid AE0$	starts with 0, ends with 0, and $\Delta = -1$
$D \rightarrow 1 \mid BE1$	starts with 1, ends with 1, and $\Delta = +1$
$E \rightarrow \varepsilon \mid EE \mid 0E1 \mid 1E0$	$\Delta = 0$ (from the lecture notes)

This grammar can be written more compactly as follows:

$S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0E1 \mid 1E0 \mid 0E1E0 \mid 1E0E1$	$\#11 = \#00$
$E \rightarrow \varepsilon \mid EE \mid 0E1 \mid 1E0$	$\#1 = \#0$



*3. Practice only. Do not submit solutions.

Let L_1 and L_2 be arbitrary regular languages over the alphabet $\Sigma = \{0, 1\}$. Prove that the following languages are also regular.

- (a) $\text{FARO}(L_1, L_2) := \{\text{faro}(x, z) \mid x \in L_1 \text{ and } z \in L_2 \text{ with } |x| = |z|\}$, where

$$\text{faro}(x, z) := \begin{cases} z & \text{if } x = \varepsilon \\ a \cdot \text{faro}(z, y) & \text{if } x = ay \end{cases}$$

Solution: Let $M_1 = (Q_1, s_1, A_1, \delta_1)$ and $M_2 = (Q_2, s_2, A_2, \delta_2)$ be arbitrary DFAs that accept the regular languages L_1 and L_2 , respectively. We build a **DFA** $M = (Q, s, A, \delta)$ that accepts $\text{FARO}(L_1, L_2)$ using the following modified product construction:

$$Q = Q_1 \times Q_2 \times \{1, 2\}$$

$$s = (s_1, s_2, 1)$$

$$A = A_1 \times A_2 \times \{1\}$$

$$\delta((q_1, q_2, 1), a) = (\delta(q_1, a), q_2, 2)$$

$$\delta((q_1, q_2, 2), a) = (q_1, \delta(q_2, a), 1)$$

M reads the input string and alternately passes its input bits to simulations of M_1 and M_2 . State (q_1, q_2, i) means that machine M_1 is in state q_1 , machine M_2 is in state q_2 , and the next input bit should be passed to M_i . The condition $|w| = |z|$ in the definition of $\text{faro}(w, z)$ implies that M can accept only if it has read an even number of bits. ■

- (b) $\text{SHUFFLES}(L_1, L_2) := \bigcup_{w \in L_1, y \in L_2} \text{shuffles}(w, y)$, where $\text{shuffles}(w, y)$ is the set of all strings obtained by shuffling w and y , or equivalently, all strings in which w and y are complementary subsequences. Formally:

$$\text{shuffles}(w, y) = \begin{cases} \{y\} & \text{if } w = \varepsilon \\ \{w\} & \text{if } y = \varepsilon \\ \{a\} \cdot \text{shuffles}(x, y) \cup \{b\} \cdot \text{shuffles}(w, z) & \text{if } w = ax \text{ and } y = bz \end{cases}$$

Solution: Let $M_1 = (Q_1, s_1, A_1, \delta_1)$ and $M_2 = (Q_2, s_2, A_2, \delta_2)$ be arbitrary DFAs that accept the regular languages L_1 and L_2 , respectively. We build a **NFA** $M = (Q, s, A, \delta)$ that accepts $\text{SHUFFLES}(L_1, L_2)$ using the following modified product construction:

$$Q = Q_1 \times Q_2$$

$$s = (s_1, s_2)$$

$$A = A_1 \times A_2$$

$$\delta((q_1, q_2), a) = \{(\delta_1(q_1, a), q_2), (q_1, \delta_2(q_2, a))\}$$

Each time M reads a bit, it nondeterministically guesses whether to pass that bit to M_1 or M_2 . ■