CSC236 Fall 2018
Assignment #3: formal languages
due November 30th, 3 p.m.

The aim of this assignment is to give you some practice with formal languages, FSAs, and regular expressions.

Your assignment must be **typed** to produce a PDF document **a3.pdf** (hand-written submissions are not acceptable). You may work on the assignment in groups of 1 or 2, and submit a single assignment for the entire group on MarkUs

1. Let $x \in \mathbb{N}$. Prove that **term(x)** (below) terminates. **Hint:** Trace the values of x and y for a few different inputs, then devise a loop invariant that helps prove termination.

```
def term(x):
y = x**3
while y != 0:
x = x - 1
y = y - 3*x*x - 3*x - 1
```

sample solution: A good candidate for a strictly decreasing sequence of natural numbers is the sequence of x values after iterations. It should be easy to show they are decreasing, so most of the work will be to show that they are always natural numbers. So part of a reasonable invariant will be that after the ith iteration, x_i is a natural number. The values of x and y are interdependent, so I'll also make the claim that y_i is a natural number part of the invariant. By trying a few small values, it's clear that $x_i^3 = y_i$, so I will include that in the invariant.

Let x_i, y_i be the values of x and y after the ith iteration of the loop, if it occurs.

Define $P(i)$: "At the end of the ith iteration, if it occurs, x_i and y_i are natural numbers and $y_i = x_i^3$."

I prove $\forall i \in \mathbb{N}, P(i)$ by simple induction.

base case: At the end of the 0th iteration the variables are initialized but the loop has not been entered, so $x_0 \in \mathbb{N}$ (from choice of x), and $y_0 = x_0^3 \in \mathbb{N}$, since natural numbers are closed under multiplication.

inductive step: Let $i \in \mathbb{N}$ and assume $P(i)$. I will show that $P(i+1)$ follows

If an $(i+1)$th iteration occurs, then $y_i \neq 0$, by the loop condition. If $y_i \neq 0$ and, by the IH, $y_i = x_i^3$, then $x_i \neq 0$, and since $x_i \in \mathbb{N}$ we must have $x_i \geq 1$, and $x_{i+1} = x_i - 1 \geq 0$. Since x_{i+1} is an integer no smaller than 0, $x_{i+1} \in \mathbb{N}$. By the IH, $y_{i+1} = x_i^3 - 3(x_i-1)^2 - 3(x_i-1) - 1 = (x_i-1)^3 = x_{i+1}^3$, and $y_{i+1} = x_{i+1}^3 \in \mathbb{N}$, since $\mathbb{N}$ is closed under multiplication. ■

If an $(i+1)$th iteration occurs, then $x_{i+1} = x_i - 1$, so $x_{i+1} < x_i$.

Thus the sequence $\langle x_i \rangle$ is a strictly decreasing sequence of natural numbers, and has a smallest element. Since its smallest element is also its last element, the loop terminates. ■

2. Let $\Sigma = \{a, b\}$, $L_a = \{a^k \mid k \in \mathbb{N}\}$, $L_b = \{b^j \mid j \in \mathbb{N}\}$, and $L_2 = \{x \in \{a,b\}^* \mid |x| \text{ is even}\}$. **Do not** draw the machines below. Specify them with the quintuple $(Q, \Sigma = \{a,b\}, \delta, Q_0, F)$, where transition function δ is a table with symbols on the rows and states on the columns.

   (a) Construct DFA M_a such that $L(M_a) = L_a$. Be sure to include all states, including dead states. Devise a state invariant for M_a and use it to prove that M_a accepts L_a.

   sample solution: Here is M_a:

$$Q = \{q_a, q_b\}$$
$$\Sigma = \{a, b\}$$
$$F = \{q_a\}$$

start: q_a

δ	q_a	q_b
a	q_a	q_b
b	q_b	q_b

M_a is designed to accept strings that have only the character a. Here is a state invariant that says that:

$$P(s) : \delta^*(q_a, s) = \begin{cases} q_a & \text{if } s \text{ has no } b \\ q_b & \text{if } s \text{ has at least one } b \end{cases}$$

I will prove $\forall s \in \Sigma^*, P(s)$ by structural induction on Σ^* defined as the smallest set such that:

i. $\varepsilon \in \Sigma^*$

ii. If $s \in \Sigma^*$, then so are sa and sb.

basis: Since q_a is the start state, it is easy to verify

$$\delta^*(q_a, \varepsilon) = \begin{cases} q_a & \text{if } \varepsilon \text{ has no } b \\ q_b & \text{if } \varepsilon \text{ has at least one } b \end{cases}$$

...since the first statement has both hypothesis and conclusion true, and the second statement has a false hypothesis, so any conclusion follows. So $P(\varepsilon)$ holds.

inductive step: Let $s \in \Sigma^*$ and assume $P(s)$. I must show that $P(sa)$ and $P(sb)$ follow.

case sa:

$$\delta^*(\varepsilon, sa) = \delta(\delta^*(\varepsilon, s), a) = \begin{cases} \delta(q_a, a) & \text{if } s \text{ has no } b \\ \delta(q_b, a) & \text{if } s \text{ has at least one } b \end{cases} \quad \text{\# by } P(s)$$
$$= \begin{cases} q_a & \text{if } sa \text{ has no } b \\ q_b & \text{if } sa \text{ has at least one } b \end{cases}$$

So $P(sa)$ follows.

case sb:

$$\delta^*(\varepsilon, sb) = \delta(\delta^*(\varepsilon, s), b) = \begin{cases} \delta(q_a, b) & \text{if } s \text{ has no } b \\ \delta(q_b, b) & \text{if } s \text{ has at least one } b \end{cases} \quad \text{\# by } P(s)$$
$$= \begin{cases} q_a & \text{if } sb \text{ has no } b \\ q_b & \text{if } sb \text{ has at least one } b \end{cases}$$

So $P(sb)$ follows. Notice that, rather than repeating the information second line of the invariant twice by mechanically following the output of $\delta(q_a, b)$, we can form a consistent first line of the first invariant by using vacuous truth.

So $P(sa)$ and $P(sb)$ follow. ■

The first line of the invariant tells us that M_a accepts all strings with no b. The contrapositive of the second line of the invariant tells us that any string that drives M_a to accepting state q_a has no b. ■

(b) Construct DFA M_b such that $L(M_b) = L_b$. Be sure to include all states, including dead states. Devise a state invariant for M_b and use it to prove that M_b accepts L_b.

sample solution: Here is M_b:

$$Q = \{q_a, q_b\}$$
$$\Sigma = \{a, b\}$$
$$F = \{q_b\}$$
$$\text{start: } q_b$$

δ	q_a	q_b
a	q_a	q_a
b	q_a	q_b

M_b is designed to accept strings that have only the character b. Here is a state invariant that says that:

$$P(s) : \delta^*(q_b, s) = \begin{cases} q_a & \text{if } s \text{ has at least one } a \\ q_b & \text{if } s \text{ has no } a \end{cases}$$

I will prove $\forall s \in \Sigma^*, P(s)$ by structural induction on Σ^* defined as the smallest set such that:

i. $\varepsilon \in \Sigma^*$

ii. If $s \in \Sigma^*$, then so are sa and sb.

basis: Since q_b is the start state, it is easy to verify

$$\delta^*(q_a, \varepsilon) = \begin{cases} q_a & \text{if } \varepsilon \text{ has at least one } a \\ q_b & \text{if } \varepsilon \text{ has no } a \end{cases}$$

. . . since the second statement has both hypothesis and conclusion true, and the first statement has a false hypothesis, so any conclusion follows. So $P(\varepsilon)$ holds.

inductive step: Let $s \in \Sigma^*$ and assume $P(s)$. I must show that $P(sa)$ and $P(sb)$ follow.

case sa:

$$\begin{aligned} \delta^*(\varepsilon, sa) = \delta(\delta^*(\varepsilon, s), a) &= \begin{cases} \delta(q_a, a) & \text{if } s \text{ has at least one } a \\ \delta(q_b, a) & \text{if } s \text{ has no } a \end{cases} && \text{\# by } P(s) \\ &= \begin{cases} q_a & \text{if } sa \text{ has at least one } a \\ q_b & \text{if } sa \text{ has no } a \end{cases} \end{aligned}$$

So $P(sa)$ follows. Note the use of vacuous truth to verify the second line of the invariant.

case sb:

$$\delta^*(\varepsilon, sb) = \delta(\delta^*(\varepsilon, s), b) \quad = \quad \begin{cases} \delta(q_a, b) & \text{if } s \text{ has at least one } a \\ \delta(q_b, b) & \text{if } s \text{ has no } a \end{cases} \qquad \# \text{ by } P(s)$$

$$= \quad \begin{cases} q_a & \text{if } sb \text{ has at least one } a \\ q_b & \text{if } sb \text{ has no } a \end{cases}$$

So $P(sb)$ follows.

So $P(sa)$ and $P(sb)$ follow. ■

The second line of the invariant tells us that strings with no a drive M_b to accepting state q_b. The contrapositive of the first line of the invariant tells us that any string that drives M_b to accepting state q_b has no a. ■

(c) Construct DFA M_2 such that $L(M_2) = L_2$. Be sure to include all states, including dead states. Devise a state invariant for M_2 and use it to prove that M_2 accepts L_2.

sample solution: Here is M_2:

$Q = \{q_0, q_1\}$

$\Sigma = \{a, b\}$

$F = \{q_0\}$

start: q_0

δ	q_0	q_1
a	q_1	q_0
b	q_1	q_0

M_2 is designed to accept exactly the strings that have an even length. Here is an invariant that says that:

$$P(s) : \delta^*(q_0, s) = \begin{cases} q_0 & \text{if } s \text{ has an even number of characters} \\ q_1 & \text{if } s \text{ has an odd number of characters} \end{cases}$$

I will prove $\forall s \in \Sigma^*, P(s)$ by structural induction on Σ^* defined as the smallest set such that:

i. $\varepsilon \in \Sigma^*$

ii. If $s \in \Sigma^*$, then so are sa and sb.

basis: Since q_0 is the start state it is easy to verify the following:

$$\delta^*(q_0, \varepsilon) = \begin{cases} q_0 & \text{if } \varepsilon \text{ has an even number of characters} \\ q_1 & \text{if } \varepsilon \text{ has an odd number of characters} \end{cases}$$

So $P(\varepsilon)$ follows. Note the use of vacuous truth to verify the second line of the invariant.

inductive step: Let $s \in \Sigma^*$ and assume $P(s)$. I will prove the $P(sa)$ and $P(sb)$ follow.

case sa:

$$\delta^*(q_0, sa) = \delta(\delta^*(q_0, s), a) \quad = \quad \begin{cases} \delta(q_0, a) & \text{if } s \text{ has an even number of characters} \\ \delta(q_1, a) & \text{if } s \text{ has an odd number of characters} \end{cases} \qquad \# \text{ by } P(s)$$

$$= \quad \begin{cases} q_1 & \text{if } sa \text{ has an odd number of characters} \\ q_0 & \text{if } sa \text{ has an even number of characters} \end{cases}$$

So $P(sa)$ follows.

case sb:

$$\delta^*(q_0, sb) = \delta(\delta^*(q_0, s), b) \quad = \quad \begin{cases} \delta(q_0, b) & \text{if } s \text{ has an even number of characters} \\ \delta(q_1, b) & \text{if } s \text{ has an odd number of characters} \end{cases} \qquad \# \text{ by } P(s)$$

$$= \quad \begin{cases} q_1 & \text{if } sb \text{ has an odd number of characters} \\ q_0 & \text{if } sb \text{ has an even number of characters} \end{cases}$$

So $P(sb)$ follows. ■

The invariant statement about q_0 shows that all strings with an even number of characters are accepted at state q_0. The contrapositive of the statement about q_1 shows that all strings accepted at q_0 have an even number of characters. ■

(d) Construct DFA $M_{a|b}$ such that $L(M_{a|b}) = L_a \cup L_b$. Use the product construction from week 9 slides. Explain how you can use the proofs for M_a and M_b to establish that $M_{a|b}$ accepts $L_a \cup L_b$.

sample solution: I construct $M_{a|b}$ by creating the Cartesian product of M_a and M_b: each state in $M_{a|b}$ is a pair of states from M_a and M_b, and each transition is a pair of transitions. The accepting states are all those that have either the accepting state for M_a or the accepting state for M_b, and the start state is the pair with the start state for M_a and the start state for M_b:

$$Q = \{(q_{aM_a}, q_{aM_b}), (q_{aM_a}, q_{bM_b}), (q_{bM_a}, q_{aM_b}), (q_{bM_a}, q_{bM_b})\}$$
$$\Sigma = \{a, b\}$$
$$F = \{(q_{aM_a}, q_{aM_b}), (q_{aM_a}, q_{bM_b}), (q_{bM_a}, q_{bM_b})\}$$
$$\text{start: } (q_{aM_a}, q_{bM_b})$$

δ	(q_{aM_a}, q_{aM_b})	(q_{aM_a}, q_{bM_b})	(q_{bM_a}, q_{aM_b})	(q_{bM_a}, q_{bM_b})
a	(q_{aM_a}, q_{aM_b})	(q_{aM_a}, q_{aM_b})	(q_{bM_a}, q_{aM_b})	(q_{bM_a}, q_{aM_b})
b	(q_{bM_a}, q_{aM_b})	(q_{bM_a}, q_{bM_b})	(q_{bM_a}, q_{aM_b})	(q_{bM_a}, q_{bM_b})

The corresponding invariant is:

$$\delta^*((q_{aMa}, q_{bM_b}), s) = \begin{cases} (q_{aM_a}, q_{aM_b}) & \text{if } s \text{ has no } b \text{ and } s \text{ has at least one } a \\ (q_{aM_a}, q_{bM_b}) & \text{if } s \text{ has no } b \text{ and } s \text{ has no } a \\ (q_{bM_a}, q_{aM_b}) & \text{if } s \text{ has at least one } b \text{ and } s \text{ has at least one } a \\ (q_{bM_a}, q_{bM_b}) & \text{if } s \text{ has at least one } b \text{ and } s \text{ has no } a \end{cases}$$

By inspecting the transition function δ I verify that each transition is a pair of transitions from M_a and M_b, and each line of the invariant represents a conjunction of invariant lines from M_a and M_b. These were already proved in the previous parts, so the conjunctions follow without further proof. ■

The invariants says that all strings that do not contain at least one a and at least one b are accepted, and the contrapositive of the third line says that any string that drives this machine to an accepting state that lacks either any a or any b characters, or both. ■

(e) Construct DFSA $M_{a|b \text{ even}}$ such that $L(M_{a|b \text{ even}}) = (L_a \cup L_b) \cap L_2$. Explain how you can use the proofs for the preceding machine to establish that $M_{a|b \text{ even}}$ accepts $(L_a \cup L_b) \cap L_2$.

sample solution: I construct $M_{a|b \text{ even}}$ by taking the Cartesian product of M_a, M_b and M_2, so the new machines states are triples of the original machines' states. For notational convenience, I indicate the triples of original states in the subscripts only, e.g. $(q_{aM_a}, q_{bM_b}, q_{0M_2}) = q_{(a,b,0)}$. The accepting states for $M_{a|b \text{ even}}$ are those states corresponding to q_0 in M_2 and either q_a in M_a or q_b in M_b. $M_{a|b \text{ even}}$ starts in $q_{(a,b,0)}$:

$$Q = \{q_{(a,a,0)}, q_{(a,a,1)}, q_{(a,b,0)}, q_{(a,b,1)}, q_{(b,a,0)}, q_{(b,a,1)}, q_{(b,b,0)}, q_{(b,b,1)}\}$$
$$\Sigma = \{a, b\}$$
$$F = \{q_{(a,a,0)}, q_{(a,b,0)}, q_{(b,b,0)}\}$$
start: $q_{(a,b,0)}$

δ	$q_{(a,a,0)}$	$q_{(a,a,1)}$	$q_{(a,b,0)}$	$q_{(a,b,1)}$	$q_{(b,a,0)}$	$q_{(b,a,1)}$	$q_{(b,b,0)}$	$q_{(b,b,1)}$
a	$q_{(a,a,1)}$	$q_{(a,a,0)}$	$q_{(a,a,1)}$	$q_{(a,a,0)}$	$q_{(b,a,1)}$	$q_{(b,a,0)}$	$q_{(b,a,1)}$	$q_{(b,a,0)}$
b	$q_{(b,a,1)}$	$q_{(b,a,0)}$	$q_{(b,b,1)}$	$q_{(b,b,0)}$	$q_{(b,a,1)}$	$q_{(b,a,0)}$	$q_{(b,b,1)}$	$q_{(b,b,0)}$

The corresponding invariant is:

$$\delta^*(q_{(a,b,0)}, s) = \begin{cases} q_{(a,a,0)} & \text{if } s \text{ has no } b\text{, at least one } a\text{, and an even character count.} \\ q_{(a,a,1)} & \text{if } s \text{ has no } b\text{, at least one } a\text{, and an odd character count.} \\ q_{(a,b,0)} & \text{if } s \text{ has no } b\text{, no } a\text{, and an even character count.} \\ q_{(a,b,1)} & \text{if } s \text{ has no } b\text{, no } a\text{, and an odd character count.} \\ q_{(b,a,0)} & \text{if } s \text{ has at least one } b\text{, at least one } a\text{, and an even character count.} \\ q_{(b,a,1)} & \text{if } s \text{ has at least one } b\text{, at least one } a\text{, and an odd character count.} \\ q_{(b,b,0)} & \text{if } s \text{ has at least one } b\text{, no } a\text{, and an even character count.} \\ q_{(b,b,1)} & \text{if } s \text{ has at least one } b\text{, no } a\text{, and an odd character count.} \end{cases}$$

By inspecting the transition function δ I verify that each transition is a triple of transitions from M_a, M_b, and M_2, and each line of the invariant is a conjunction of the corresponding three invariant statements. Since these statements were already proved earlier, the conjunctions follow with no further proof. ■
The invariants lines 1, 3, and 7 say that strings with an even number of characters and either no b or no a (or neither a nor b) are accepted. The contrapositives of the remaining lines say that strings that drive this machine to an accepting state have an even number of characters and do not have both an a and a b. ■

3. Let $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$, $L_0 = \{x \in \Sigma^* \mid x \text{ represents a number equivalent to 0 mod 3 in base 10}\}$, $L_1 = \{x \in \Sigma^* \mid x \text{ represents a number equivalent to 1 mod 3 in base 10}\}$, and $L_2 = \{x \in \Sigma^* \mid x \text{ represents a number equivalent to 2 mod 3 in base 10}\}$. Construct M_0 that accepts $L_0 \cup \{\varepsilon\}$, M_1 that accepts L_1 and M_2 that accepts L_2, being sure that each machine has exactly 3 states. Do **not** draw your machines, specify them with a quintuple.

sample solution: I exhibit the three machines beside their corresponding reverse machines, for the purpose of comparison.

M_0 **and** $M_{Rev(0)}$: Notice that the states, alphabet, start, and accepting states are the same, just the transition function is reversed.

$$
\begin{aligned}
Q_0 &= \{q_0, q_1, q_2\} & Q_{Rev(0)} &= \{q_0, q_1, q_2\} \\
\Sigma &= \{0,1,2,3,4,5,6,7,8,9\} & \Sigma &= \{0,1,2,3,4,5,6,7,8,9\} \\
F &= \{q_0\} & F &= \{q_0\} \\
&\text{start: } q_0 & &\text{start: } q_0
\end{aligned}
$$

δ	q_0	q_1	q_2
$0,3,6,9$	q_0	q_1	q_2
$1,4,7$	q_1	q_2	q_0
$2,5,8$	q_2	q_0	q_1

δ	q_0	q_1	q_2
$0,3,6,9$	q_0	q_1	q_2
$1,4,7$	q_2	q_0	q_1
$2,5,8$	q_1	q_2	q_0

Notice that M_0 and $M_{Rev(0)}$ are **isomorphic**, in other words, if we re-name all the q_1 states to q_2, and all the q_2 states to q_1 in the second machine, we have identical transition functions for both machines. This means that M_0 and $M_{Rev(0)}$ have the same behaviour, except strings that end up in q_1 on one machine end up in q_2 in the other, and vice-versa. But the same set of strings end up in q_0, the accepting state. So M_0 accepts multiples of 3, their reverses, and ε. A consequence is that multiples of 3, in base 10, are also multiples of 3 when reversed.

M_1 **and** $M_{Rev(1)}$: Notice that the states and alphabet are the same, the start and accept states are swapped, and the transition functions are reversed.

$$
\begin{aligned}
Q_1 &= \{q_0, q_1, q_2\} & Q_{Rev(1)} &= \{q_0, q_1, q_2\} \\
\Sigma &= \{0,1,2,3,4,5,6,7,8,9\} & \Sigma &= \{0,1,2,3,4,5,6,7,8,9\} \\
F &= \{q_1\} & F &= \{q_0\} \\
&\text{start: } q_0 & &\text{start: } q_1
\end{aligned}
$$

δ	q_0	q_1	q_2
$0,3,6,9$	q_0	q_1	q_2
$1,4,7$	q_1	q_2	q_0
$2,5,8$	q_2	q_0	q_1

δ	q_0	q_1	q_2
$0,3,6,9$	q_0	q_1	q_2
$1,4,7$	q_2	q_0	q_1
$2,5,8$	q_1	q_2	q_0

Notice (again) that M_1 and $M_{Rev(1)}$ are **isomorphic**, in other words, if we re-name all the q_1 states to q_0, and all the q_0 states to q_1 in the second machine, we have identical transition functions. This means that M_1 and $M_{Rev(1)}$ have the same behaviour, except strings that end up in q_1 on one machine end up in q_0 in the other, and vice-versa. But the same set of strings end up in the accepting state in either case. So M_1 accepts strings that have remainder 1 upon division by 3, and their reverses. A consequence is that numbers congruent to 1 mod 3, in base 10, are also congruent to 1 mod 3 when reversed.

M_2 **and** $M_{Rev(2)}$: Notice that the states and alphabet are the same, the start and accept states are swapped, and the transition functions are reversed.

$$\begin{aligned} Q_2 &= \{q_0, q_1, q_2\} & Q_{Rev(2)} &= \{q_0, q_1, q_2\} \\ \Sigma &= \{0,1,2,3,4,5,6,7,8,9\} & \Sigma &= \{0,1,2,3,4,5,6,7,8,9\} \\ F &= \{q_2\} & F &= \{q_0\} \\ &\text{start: } q_0 & &\text{start: } q_2 \end{aligned}$$

δ	q_0	q_1	q_2
$0,3,6,9$	q_0	q_1	q_2
$1,4,7$	q_1	q_2	q_0
$2,5,8$	q_2	q_0	q_1

δ	q_0	q_1	q_2
$0,3,6,9$	q_0	q_1	q_2
$1,4,7$	q_2	q_0	q_1
$2,5,8$	q_1	q_2	q_0

Notice (again) that M_2 and $M_{Rev(2)}$ are **isomorphic**, in other words, if we re-name all the q_2 states to q_0, and all the q_0 states to q_2 in the second machine, we have identical transition functions. This means that M_1 and $M_{Rev(2)}$ have the same behaviour, except strings that end up in q_2 on one machine end up in q_0 in the other, and vice-versa. But the same set of strings end up in the accepting state in either case, since we swapped start and accepting states. So M_2 accepts strings that have remainder 2 upon division by 3, and their reverses. A consequence is that numbers congruent to 2 mod 3, in base 10, are also congruent to 2 mod 3 when reversed.

Now use the structure of your machines to explain why $L_0 = \textbf{Rev}(L_0)$, $L_1 = \textbf{Rev}(L_1)$, and $L_2 = \textbf{Rev}(L_2)$

sample solution: I did this in-line, above.

4. Let $\mathcal{RE}$ be the set of regular expressions over the alphabet $\Sigma = \{0,1\}$. Use structural induction on $\mathcal{RE}$ to prove:

(a)

$$\forall r \in \mathcal{RE}, \exists r' \in \mathcal{RE}, \textbf{Rev}(\mathcal{L}(r)) = \mathcal{L}(r')$$

sample solution: Define $P(r)$: "$\exists r' \in \mathcal{RE}, Rev(L(r)) = L(r')$." I prove this using structural induction.

basis: There are four cases to consider:

- $Rev(L(\emptyset)) = \emptyset = L(\emptyset)$, since there are no strings to reverse in $\emptyset$. So $P(\emptyset)$ follows.
- $Rev(L(\varepsilon)) = Rev(\{\varepsilon\}) = \{\varepsilon^R\} = \{\varepsilon\} = L(\varepsilon)$. So $P(\varepsilon)$ follows.
- $Rev(L(0)) = Rev(\{0\}) = \{0^R\} = \{0\} = L(0)$. So $P(0)$ follows.
- $Rev(L(1)) = Rev(\{1\}) = \{1^R\} = \{1\} = L(1)$. So $P(1)$ follows.

inductive step: Let $r_1, r_2 \in \mathcal{RE}$ and assume $P(r_1) \wedge P(r_2)$. Let $r_1', r_2' \in \mathcal{RE}, Rev(L(r_1)) = L(r_1') \wedge Rev(L(r_2)) = L(r_2')$. There are three cases to consider:

case $(r_1 + r_2)$: Let $s \in \{0,1\}^*$.

Then $s \in Rev(L(r_1 + r_2)) = Rev(L(r_1) \cup L(r_2))$

$\Leftrightarrow\ \exists s' \in L(r_1) \cup L(r_2), s = s'^R$, by definition of Rev.

$\Leftrightarrow\ s \in Rev(L(r_1)) \vee s \in Rev(L(r_2))$, by definition of $\cup$.

$\Leftrightarrow\ s \in L(r_1') \vee s \in L(r_2')$, by IH.

$\Leftrightarrow\ s \in L(r_1') \cup L(r_2') = L(r_1' + r_2')$, by definition of $\cup$

So $Rev(L(r_1 + r_2)) = L(r_1' + r_2')$, and $P(r_1 + r_2)$ follows.

case (r_1r_2): Let $s \in \{0,1\}^*$.

Then $s \in Rev(L(r_1r_2)) = Rev(L(r_1)L(r_2))$

$\Leftrightarrow\ \exists s' \in L(r_1)L(r_2), s = s'^R$, by definition of Rev.

$\Leftrightarrow\ \exists v \in L(r_1), w \in L(r_2), s = w^Rv^R$, by page 185 of course notes.

$\Leftrightarrow\ s \in Rev(L(r_2))Rev(L(r_1)) = L(r_2')L(r_1') = L(r_2'r_1')$, by IH.

So $Rev(L(r_1r_2)) = L(r_2'r_1')$, and $P(r_1r_2)$ follows.

case r_1^* Let $s \in \{0,1\}^*$.

Then $s \in Rev(L(r_1^*)) = Rev(L(r_1)^*)$.

$\Leftrightarrow\ \exists s' \in L(r_1)^*, s = s'^R$, by definition of Rev.

$\Leftrightarrow\ s = \varepsilon^R = \varepsilon \vee \exists k \in \mathbb{N}^+, u_1, \ldots, u_k \in L(r_1), s = u_k^R \cdots u_1^R$, by page 185 of course notes and definition of $L(r_1)^*$.

$\Leftrightarrow\ s \in Rev(L(r_1))^* = L(r_1')*$, by IH.

So $Rev(L(r_1^*)) = L(r_1')^*$, and $P(r_1^*)$ follows.

In all cases the claim follows. ■

(b) Using the definition of Prefix(L) in Exercise 11 at the end of Chapter 7:

$$\forall r \in \mathcal{RE}, \exists r' \in \mathcal{RE}, \text{Prefix}(\mathcal{L}(r)) = \mathcal{L}(r')$$

sample solution: Define $P(r) : \exists r' \in \mathcal{RE}, Prefix(L(r)) = L(r')$. I will prove $\forall r \in \mathcal{RE}, P(r)$.

basis: There are four cases to consider:

case $\emptyset$: Since there are no strings in $L(\emptyset) = \emptyset$, there are no concatenations of strings that form a string in $L(\emptyset)$, so $Prefix(L(\emptyset)) = \emptyset = L(\emptyset)$. So $P(\emptyset)$ holds.

case ε: The single string in $L(\varepsilon)$ has length 0, and prefix of it must also have length 0, so $Prefix(L(\varepsilon)) = L(\varepsilon)$. So $P(\varepsilon)$ holds.

case 0: The single string in $L(0)$ has length 1, so any possible prefixes have length 0 or 1. The only possibility of length 0 is ε, and the only possible prefix of length 1 is 0, so $Prefix(L(0)) = \{\varepsilon, 0\} = L(\varepsilon + 0)$. So $P(0)$ holds.

case 1: The single string in $L(1)$ has length 1, so any possible prefixes have length 0 or 1. The only possibility of length 0 is ε, and the only possible prefix of length 1 is 1, so $Prefix(L(1)) = \{\varepsilon, 1\} = L(\varepsilon + 1)$. So $P(0)$ holds.

inductive step: Let $r_1, r_2 \in \mathcal{RE}$ and assume $P(r_1)$ and $P(r_2)$. Let $r_1', r_2' \in \mathcal{RE}, Prefix(L(r_1)) = L(r_1'), Prefix(L(r_2)) = L(r_2')$. There are three cases to consider:

case $(r_1 + r_2)$: Let $s \in \{0,1\}^*$.

Then $s \in Prefix(L(r_1 + r_2)) = Prefix(L(r_1) \cup L(r_2))$

$\Leftrightarrow\ \exists y \in \{0,1\}^*, sy \in L(r_1) \cup L(r_2))$, by definition of $Prefix$.

$\Leftrightarrow\ \exists y \in \{0,1\}^*, sy \in L(r_1) \vee sy \in L(r_2)$, by definition of $\cup$.

$\Leftrightarrow\ s \in Prefix(L(r_1)) \vee s \in Prefix(L(r_2))$, by definition of Prefix.

$\Leftrightarrow\ s \in L(r_1') \vee s \in L(r_2')$, by IH.

$\Leftrightarrow\ s \in L(r_1') \cup L(r_2') = L(r_1' + r_2')$, definition of $\cup$.

So $Prefix(L(r_1 + r_2)) = L(r_1' + r_2')$ and $P(r_1 + r_2)$ follows.

case (r_1r_2): Let $s \in \{0,1\}^*$.

Then $s \in Prefix(L(r_1r_2)) = Prefix(L(r_1)L(r_2))$

$\Leftrightarrow\ \exists y \in \{0,1\}^*, sy \in L(r_1)L(r_2)$, by definition of $Prefix$

$\Leftrightarrow\ \exists y \in \{0,1\}^*, u \in L(r_1), v \in L(r_2), sy = uv$, by definition of concatenation.

$\Leftrightarrow\ \exists y, s', u' \in \{0,1\}^*, u \in L(r_1), v \in L(r_2), sy = uv \wedge (u = ss' \vee s = uu')$, since either $|u| \leq |s|$ or $|s| \leq |u|$.

$\Leftrightarrow\ s \in Prefix(L(r_1)) \vee s \in L(r_1)Prefix(L(r_2))$, definition of $Prefix$ and $u'y \in L(r_2)$.

$\Leftrightarrow\ s \in Prefix(L(r_1)) \cup L(r_1)Prefix(L(r_2)) = L(r_1' + r_1 r_2')$, by IH

So $Prefix(L(r_1 r_2)) = L(r_1' + r_1 r_2')$ and $P(r_1 r_2)$ follows.

case r_1^*: Let $s \in Prefix(L(r_1^*)) = Prefix(L(r_1)^*)$.

$\Rightarrow\ \exists k \in \mathbb{N}, s \in Prefix(L(r_1)^k)$, by definition of $L(r_1)^*$. Let $k \in \mathbb{N}, s \in Prefix(L(r_1)^k) \wedge \forall k' \in \mathbb{N}, s \in Prefix(L(r_1)^{k'}) \Rightarrow k \leq k'$.

$\Rightarrow\ \exists y \in \Sigma^*, sy \in L(r_1)^k$, by definition of $Prefix$. Let $y \in \Sigma^*$ be such a string.

$\Rightarrow\ sy \in L(r_1)^0 \vee sy \in L(r_1)^{k-1}L(r_1)$, since natural number k is either 0 or positive.

$\Rightarrow\ s = y = sy = \varepsilon \vee s \in L(r_1)^{k-1}Prefix(L(r_1))$, by minimality of k, since all of sy contributed by $L(r_1)^{k-1}$ must be a substring of s, or a smaller k' could have been chosen above.

$\Rightarrow\ s = \varepsilon \vee s \in L(r_1)^*Prefix(L(r_1))$, that is, $s \in L(r_1^*(\varepsilon + r_1'))$, by IH.

$\Rightarrow\ Prefix(L(r_1^*)) \subseteq L(r_1^*(\varepsilon + r_1'))$.

Let $s \in L(r_1^*(\varepsilon + r_1')) = L(r_1)^* \cup L(r_1)^*Prefix(L(r_1)))$, by IH.

It is clear that $L(r_1)^* \subseteq Prefix(L(r_1)^*)$, since every string is a prefix of itself. So now consider the other case, where $s \in L(r_1)^*Prefix(L(r_1))$, and assume $Prefix(L(r_1))$ is non-empty (since otherwise the concatenation yields the empty set, which is a subset of any set).

$\Rightarrow\ \exists k \in \mathbb{N}, s \in L(r_1)^k Prefix(L(r_1))$. Let $k \in \mathbb{N}, s \in L(r_1)^k Prefix(L(r_1))$, by definition of $L(r_1)^*$.

$\Rightarrow\ \exists u \in L(r_1)^k, v \in Prefix(L(r_1)), s = uv$, Let u, v be such strings.

$\Rightarrow\ \exists y \in \Sigma^*, vy \in L(r_1)$, definition of $Prefix(L(r_1))$. Let y be such a string.

$\Rightarrow\ sy = uvy \in L(r_1)^{k+1} \subseteq L(r_1)^*$.

$\Rightarrow\ s \in Prefix(L(r_1)^*)$, by definition of $Prefix(L(r_1)^*) = Prefix(L(r_1^*))$.

$\Rightarrow\ L(r_1^*(\varepsilon + r_1')) \subseteq Prefix(L(r_1^*))$

So $L(r_1^*(\varepsilon + r_1')) = Prefix(L(r_1^*))$. ■

(c) If $r \in \mathcal{RE}$ does not contain the Kleene star, then $|L(r)|$ is finite.

sample solution: I'll define the predicate: $P(r)$: "if r does not contain the Kleene star, then $|L(r)|$ is finite." I will prove $\forall r \in \mathcal{RE}, P(r)$ using structural induction.

basis: There are four cases to consider:

- $L(\emptyset) = \emptyset$, and so $|L(\emptyset)| = 0$, which is quite finite.
- $L(\varepsilon) = \{\varepsilon\}$, and so $|L(\varepsilon)| = 1$, also finite.
- $L(0) = \{0\}$, and so $|\{0\}| = 1$, also finite.
- $L(1) = \{1\}$, and so $|\{1\}| = 1$, also finite.

So $P(r)$ is true when r is in the basis

inductive step: Assume $r_1, r_2 \in \mathcal{RE}$ and also assume $P(r_1) \wedge P(r_2)$. There are three cases to consider:

case $(r_1 + r_2)$: $|L(r_1 + r_2)| = |L(r_1) \cup L(r_2)| \leq |L(r_1)| + |L(r_2)|$, which is finite, since the IH says that $L(r_1)$ and $L(r_2)$ are finite. So $P(r_1 + r_2)$ follows

case $(r_1 r_2)$: $|L(r_1 r_2)| = |L(r_1)L(r_2)| \leq |L(r_1)| \times |L(r_2)|$ (the number of concatenations of a string from $L(r_1)$ with a string from $L(r_2)$), which is finite, since the IH says that $L(r_1)$ and $L(r_2)$ are finite. So $P(r_1 r_2)$ follows

case r_1^*: $P(r_1^*)$ is vacuously true, since the antecedent is false. So $P(r_1^*)$ follows.

In all possible cases, the claim follows from the inductive hypothesis. ■

5. Let $\Sigma = \{a, b, c\}$ and $L_{R4} = \{x \in \Sigma^* \mid |x| = 4 \wedge x = x^R\}$. Prove that any DFSA that accepts L_{R4} has at least nine states, not including dead states. **Hint**: consider what happens if 2 distinct length-2 prefixes of strings in L_{R4} drive the machine to the same state. Generalize your result: what can you say about a DFSA that accepts $L_R = \{x \in \Sigma^* \mid x = x^R\}$?

sample solution: Assume, for the sake of contradiction, that DFSA M with $|Q| < 9$ states, and starting state q_0, accepts L_{R4}. Since there are 9 distinct strings of length 2 over Σ, by the pigeonhole principle, there must be 2 distinct 2-character strings that drive M to the same state. Let y, z be two distinct 2-character strings such that $\delta^*(q_0, y) = \delta^*(q_0, z)$. Then $\delta^*(q_0, yy^R) = \delta^*(q_0, zy^R)$, but $yy^R = (yy^R)^R$ is in L_{R4} but $zy^R \neq yz^R = (zy^R)^R$, so zy^R is not in L_{R4}, so the state these strings drive M to is both accepting and non-accepting $\rightarrow\leftarrow$ Contradiction. Since assuming that DFSA M that accepts L_{R4} with fewer than 9 states exists led to a contradiction, this assumption is false. ■

Now consider $L_R = \{x \in \Sigma^* \mid x = x^R\}$. If L_R is regular, then there is a machine M_R with $|Q_R| = n$ states that accepts it. However, we can always choose $k \in \mathbb{N}$ so that $3^k > n$, and show that M_R cannot properly distinguish accepted strings of length $2k$ from those of length $2k$ that should be rejected. So L_R is not regular.