HW1 : Regular Expressions and Deterministic Finite Automata **Sample Solutions**

CSE105Sp23

Assigned questions

1. **Functions over sets of strings:**
   For this question, fix the alphabets $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, 2\}$.

   Whenever K is a set of strings over Γ and L is a set of strings over Σ, we can use the following rules to define associated sets of strings:

   $$\begin{aligned} \textsc{Substring}(K) &:= \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\} \\ \textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2s in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\} \end{aligned}$$

   Note: Formally, Substring and Rep are functions whose domains and codomains are specified as

   $$\textsc{Substring} : \mathcal{P}(\Gamma^*) \to \mathcal{P}(\Gamma^*)$$

   and

   $$\textsc{Rep} : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Gamma^*)$$

   In other words, Substring maps sets of strings with characters $\{0, 1, 2\}$ to associated sets of strings with characters $\{0, 1, 2\}$; and Rep maps sets of strings with characters in $\{0, 1\}$ to associated sets of strings with characters in $\{0, 1, 2\}$.

   (a) (*Graded for correctness*) Consider $w = 0120$ (which is a string in Γ^*). List every element of the set $\textsc{Substring}(\{w\})$. In other words, fill in the blank

   $$\textsc{Substring}(\{w\}) = \{\_\_\_\_\_\}$$

   Briefly justify your answer by referring back to the relevant definitions.
   Not graded, but good to think about: Why do we need the curly braces—"{" and "}"—around w for the input to Substring*?*

Solution: $\textsc{Substring}(\{w\}) = \{\varepsilon, 0, 1, 2, 01, 12, 20, 012, 120, 0120\}$

Justification: Using the definitions of SUBSTRING and w, we have

$$\begin{aligned}\textsc{Substring}(\{0120\}) := \{&v \in \Gamma^* \mid \text{there exist} \\ &a, b \in \Gamma^* \text{ such that } avb \in \{0120\}\} \\ = \{&v \in \Gamma^* \mid \text{there exist} \\ &a, b \in \Gamma^* \text{ such that } avb = 0120\}\end{aligned}$$

because 0120 is the only element of the set $\{0120\}$. In other words, the result will be the set of all strings v in Γ^* for which there are two strings over Γ, a and b, such that the three strings, avb, concatenated together is 0120.

To determine which values of v satisfy this criterion, we notice that w itself, as well as each possible sub-string of w, are in Γ^*, since all symbols in the string w are in the alphabet Γ. So, as the strings a and b surrounding some v could be any sub-string of w, including the empty string, then v can be any possible sub-string of w. Also, v cannot be any string that is not a sub-string of w, because then it would not be possible to be concatenated with other strings and be equal to w. Therefore, $\textsc{Substring}(\{w\})$ is the set of all possible sub-strings of w.

To enumerate all of them, we consider each possible length of a substring (from 0 to the length of w, 4) and consider all substrings of w of that length. There is exactly one string of length 0: the empty string. There are three possible strings over Γ of length 1, and each of them appears as a substring of 0120. There are nine possible strings over Γ of length 2 but only three of them are substrings of 0120 (the three substrings that start at the leftmost, second from leftmost, and second from rightmost characters of 0120). There are twenty-seven possible strings over Γ of length 3 but only two of them are substrings of 0120 (the substring that starts at the leftmost character and the substring that ends at the rightmost character). The only substring of 0120 of length 4 is the string itself.

(b) (*Graded for correctness*) Specify an example language A over Γ such that $A \neq \Gamma^*$ and yet $\textsc{Substring}(A) = \Gamma^*$, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language A and a precise and clear description of the result of computing $\textsc{Substring}(A)$ using relevant definitions to justify this description and to justify the set equality with Γ^*, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Solution: There are any possible example languages that work, e.g., let the language A be the set of all strings that begin with a 0: the language described by the regular expression $0\Gamma^*$ (in other words, adding a 0 before every element of

Γ^*). In this example, the language A is not equal to Γ^*, as it excludes all strings in Γ^* that do not begin with a 0 (e.g., ε, 1, 10, ...). However, $\textsc{Substring}(A) = \Gamma^*$ because for each string $x \in \Gamma^*$, $0x \in A$ so x is a substring of some string in A and hence, by definition of $\textsc{Substring}(A)$ (using $a = 0$ and $b = \varepsilon$), $x \in \textsc{Substring}(A)$.

(c) (*Graded for completeness*) Define the language B to be the language over Σ described by the regular expression

$$\Sigma^* 1 \Sigma^*$$

In plain English, we might explain that B is the set of all strings of 0s and 1s that contain a 1. Give a plain English explanation for the set of strings $\textsc{Rep}(B)$.

Solution: A plain English description of the set of strings $\textsc{Rep}(B)$ is the set of strings such that between every pair of 2's there is at least one 1.

Explanation: The strings in $\textsc{Rep}(B)$ can be thought of the concatenation of a prefix of 0s and 1s, followed by a some number of chunks where a 2 is followed by a string in B followed by a 2, and then a trailing string of 0s and 1s. Since each string in B is a string of 0s and 1s that contains at least 1, we get that each "chunk" in a string in $\textsc{Rep}(B)$ has at least one 1 between successive 2s.

(d) (*Graded for correctness*) Prove/disprove: For every finite language L over Σ, $\textsc{Rep}(L)$ is also a finite set of strings. A complete answer will either give a general argument starting with an arbitrary finite language and proving that the result of applying Rep is also finite, or will give a counterexample (which is a specific example of a finite language L for which applying Rep gives an infinite language, with justification referring back to the relevant definitions).

Note: A finite language is a set of finitely many strings. This includes the possibility that L is the empty set!

Solution: This statement is false. To disprove it, we find a counterexample. There are many possible counterexamples; for this solution, we consider the language $L = \emptyset$, which is a finite language by definition. Substituting $L = \emptyset$ in the definition of $\textsc{Rep}(L)$, the constraint becomes that w is in $\textsc{Rep}(L)$ whenever "between every pair of successive 2s in w is a string in $\emptyset$". Since there's no string in the empty set, this means that there can be no successive 2s in strings in $\textsc{Rep}(L)$. All strings without at least two 2s do not have any substring between successive 2s, thus these strings make the universal statement in the definition of Rep vacuously true. In particular, each string w in Σ^* has only characters 0 and 1 so satisfies the condition of not having two 2s. Thus, $\Sigma^* \subseteq \textsc{Rep}(\emptyset)$. Since Σ^* is infinite (there's no upper bound to the length of strings and hence there are infinitely many strings over Σ), we've found an infinite subset of $\textsc{Rep}(\emptyset)$ and

hence $\textsc{Rep}(\emptyset)$ is infinite.

(e) (*Graded for completeness*) Write a template for a regular expression that describes $\textsc{Rep}(L)$ when L is described by a regular expression R. You may use union, concatenation, Kleene star, and Σ, Γ, and R. (We're using the shorthand for regular expressions describing alphabets from page 64.)

Solution: $\Sigma^* \cup \Sigma^*(2R)^*2\Sigma^*$

Explanation: Any string over $\{0,1\}$ (which can be written as an element of Σ^*) is and element of $\textsc{Rep}(L)$ since there are no 2's so no successive pairs of 2s so no opportunities to fail to satisfy the membership criterion by having a string between successive pairs of 2s that is not in L. In addition, any number of repetitions of a 2 followed by a string in $L(R)$, with another 2 at the end to complete the final pair of 2's with R in between, is the pattern described by $(2R)^*2$. Concatenating Σ^* before and after the regular expression $(2R)^*2$ allows prefixes and suffixes of 0s and 1s (that can be any string because they're not "captured" by successive pairs of 2s).

2. **Deciphering regular expressions:**
For this question, let's fix the regular expression

$$R = 0^*(1 \cup 10)^*$$

For each choice of strings of length 3, $a, b, c \in \{0,1\}^3$ we can define the regular expression:

$$X_{a,b,c} = 0(a \cup b \cup c)^*$$

(a) (*Graded for completeness*) Give a plain English explanation for the language defined by regular expression R. This continues a theme from Problem 1—before trying to prove formal statements about a specific regular expression, it's often good to try to translate it into a form that is more easy to reason about. Typically speaking, the shorter and more concise your plain English description is, the more useful it will be in reasoning about the language.

Solution: $L(R)$ is the language of all strings such that after the first 1, there are no consecutive 0's.

Explanation: The first part of the regular expression is 0^*, which describes the language of all strings of 0s. The second part of the regular expression is $(1 \cup 10)^*$ which describes the set of strings that have some number of repetitions of 1 and 10. When 1 is followed by 1, or 10 followed by 1, or 1 followed by 10, or 10 followed by 10, there's no repeated 0. Any 0 in this part of the string must be preceded by a 1. Some sample strings in $L(R)$ are ε, 0, 1, 00, 01, 10, 010, 001, 0010.

(b) (*Graded for correctness*) Suppose $a = 000$, $b = 001$, $c = 011$ so

$$X_{a,b,c} = 0(000 \cup 001 \cup 011)^*$$

Show that $L(R) \not\subseteq L(X_{a,b,c})$ by giving some string in $L(R)$ which is not in $L(X_{a,b,c})$, and justifying this choice referring back to relevant definitions.

Solution: For reference, the first few elements of $L(X_{a,b,c})$ in string order are: 0, 0000, 0001, 0011, 0000000, 0000001, 0000011.

Let's say we take the string 0110.

This string is in $L(R)$ because the string can be decomposed into $0 \circ 1 \circ 10$ where $0 \in L(0^*)$ since it has only 0s and $1 \in L(1 \cup 10)$ since it matches the left part of the regular expression and $10 \in L(1 \cup 10)$ since it matches the right part of the regular expression.

However, $0110 \notin L(X_{a,b,c})$: by definition of concatenation, in order to be in $L(X_{a,b,c})$, a string needs to start with a 0 and then be followed by a string that is in $L((000 \cup 001 \cup 011)^*)$. By definition of Kleene star, each string in $L((000 \cup 001 \cup 011)^*)$ can be decomposed into some (nonnegative integer) number of 3-character substrings each of which starts with a 0. Since 110 does not start with a 0, 0110 is not in $L(X_{a,b,c})$.

(c) (*Graded for correctness*) More generally, prove that

$$L(R) \not\subseteq L(X_{a,b,c})$$

for *all* possible strings $a, b, c \in \{0,1\}^3$. Hint: What are the possible lengths of strings in $L(R_1)$ (and why does this help)?

Solution: Let's say we take the string 01.

This string is in $L(R)$ because the string can be decomposed into $0 \circ 1$ where $0 \in L(0^*)$ since it has only 0s and $1 \in L(1 \cup 10)$ since it matches the left part of the regular expression and $10 \in L(1 \cup 10)$.

However, we will show that 01 is not in $L(X_{a,b,c})$. By the definition of concatenation and Kleene Star (Sipser Book Page 44 - Definition 1.23), each string in $L(X_{a,b,c})$ starts with 0 and then can be decomposed into length 3 substrings, each being 000 or 001 or 011. In particular, the length of each string in $L(X_{a,b,c})$ is $3k+1$ where k is a nonnegative integer (representing the number of "slots" in the string). Since the length of 01 is 2, which has remainder 2 upon division by 3, it cannot be written as $3k+1$ for any nonnegative integer k (since each of those numbers has remainder 1 upon division by 3). Thus, 01 doesn't have the right length to be an element of $L(X_{a,b,c})$ and so is not an element of this set.

(d) (*Graded for correctness*) Give a specific example of three distinct strings $a, b, c \in \{0,1,2\}^3$ such that

$$L(X_{a,b,c}) \subseteq L(R)$$

Briefly justify your answer by explaining how an arbitrary element of $L(X_{a,b,c})$ is guaranteed to be an element of $L(R)$.

Solution: One answer: $a = 101$, $b = 110$, $c = 111$. The trick is to make sure that a, b, c are defined in such a way that you can never have consecutive 0's. Then, we claim that $L(X_{a,b,c}) \subseteq L(R)$.

Proof: Let w be an arbitrary string in $L(X_{a,b,c})$ with a, b, c as defined above. By definition of Kleene star, w can be decomposed into consecutive substrings of length 3 where each substring is 101, 110, or 111. Notice that $101 = 10 \circ 1$, $110 = 1 \circ 10$, and $111 = 1 \circ 1 \circ 1$ so each of these substrings can be thought of as taking up two or three "slots" in a string that's in the language described by the regular expression $(1 \cup 10)^*$, which the is the right-hand-side piece of R. Moreover, $w = \varepsilon \circ w$ and since $\varepsilon \in L(0^*)$, by definition of concatenation and since we already showed that $w \in L((1 \cup 10)^*)$, we conclude $w \in L(R)$. Since w was an arbitrary element of $L(X_{a,b,c})$, the proof of subset inclusion is complete.

(e) (*Graded for correctness*) Give a specific example of three distinct strings $a, b, c \in \{0,1,2\}^3$ such that

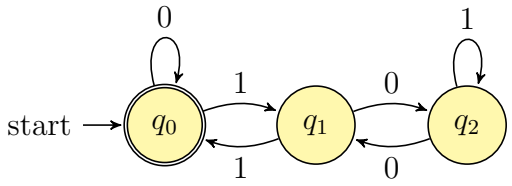
$$L(X_{a,b,c}) \not\subseteq L(R)$$

Briefly justify your answer by giving a counterexample string that is in $L(X_{a,b,c})$ and is not in $L(R)$ (and explaining why using relevant definitions).

Solution: One answer: $a = 000$, $b = 001$ and $c = 011$ (in particular, notice that b and c contain at least one 1). To prove $L(X_{a,b,c}) \not\subseteq L(R)$, consider the counterexample 001000. It is in $L(X_{000,001,011})$ because it can be written $001 \circ 000$ (using two "slots" from the Kleene star). However, it is not in $L(R)$, because (as we saw in part (a), $L(R)$ is the set of all strings such that after the first 1, there are no consecutive 0 and in 001000, after the first 1 there is 000.

Another answer: $a = 210$, $b = 120$, $c = 021$ (so that at least one of them has a 2). A counterexample that proves $L(X_{a,b,c}) \not\subseteq L(R)$ is 210, because $210 \in (210 \cup 120 \cup 021)^*$ (using one "slot" from the Kleene star) but $210 \notin L(R)$ because R only uses 0s and 1s.

3. **The right transition function can make or break a DFA:**
Consider the finite automaton $(Q, \Sigma, \delta, q_0, F)$ depicted below



where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, and $F = \{q_0\}$.

(a) (*Graded for completeness*) Find and fix the mistake in the following symbolic description of the transition function $\delta\colon Q \times \Sigma \to Q$: for each $j \in \{0, 1\}$

$$\delta(q_0, j) = q_j \qquad \delta(q_1, j) = q_{1-j} \qquad \delta(q_2, j) = q_{1+j}$$

Solution: The claimed description gives $\delta(q_1, 0) = q_{1-0} = q_1$, but the actual transition function in the diagram has $\delta(q_1, 0) = q_2$. Fix it by putting $\delta(q_1, j) = q_{2-2j}$ and keeping the rest the same.

Explanation: Currently, the transition function pertaining to q_1 is incorrect for $\delta(q_1, 0)$. While the visual depiction of the finite automaton shows that from q_1, a 0 should transition to q_2, the symbolic representation says that the transition function from q_1 is $\delta(q_1, j) = q_{1-j}$. This is incorrect when j is 0, since q_{1-0} is q_1, not q_2. Instead, we need the transition function to do the same thing that's depicted in the visual representation: from q_1, a 0 should transition to q_2, and a 1 should transition to q_0. We can fix this by changing the transition function for q_1 to $\delta(q_1, j) = q_{2-2j}$, since q_{2-0} is q_2, and q_{2-2} is q_0.

(b) (*Graded for correctness*) Keeping the same set of states Q, alphabet Σ, starting state q_0, and set of accepting states $\{q_0\}$, change the transition function δ so that the resulting finite automaton recognizes the language described by the regular expression

$$0^* \cup \Sigma^* 1000^*$$

Briefly justify why the resulting finite automaton works by describing the role of each state with your new transition function and relating it to a plain English description of the language described by the regular expression.

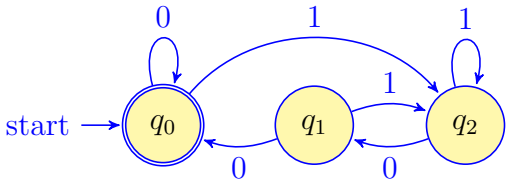
Note: with regular expressions $*$ binds more tightly than concatenation so $1000^* = (100)(0^*)$.

Solution: Transition function is the same as shown except swap the transition on q_1, i.e., $\delta(q_1, j) = q_{2j}$ and change the transition on q_0 to $\delta(q_0, 0) = q_0$ and $\delta(q_0, 1) = q_2$.

Explanation: To recognize the language described by the regular expression, the finite automaton should accept all and only strings that are all 0s or that end with a 1 followed by at least two 0s. To represent this with a finite automaton with

three states, similar to the one above, we need the start state to be in the set of accepting states, q_0, when no characters have been read, or at least two 0's have occurred in a row (since if the string ended there, it would be in the language described by the regular expression). To do this, we can use q_2 to represent the state when two more 0's are needed to be in the accept state (i.e., any time that a 1 occurs). Similarly, q_1 represents the state when one more 0 is needed to be in an accept state.

The diagram of this new finite automaton is



Informally, the state $q0$ encodes that either only $0s$ have been seen or the last two characters read were 00; the state $q1$ encodes that exactly one consecutive 0 has been most recently read; and the state $q2$ encodes that the most recent symbol read was a 1.

(Challenge question, not graded) There is a beautiful plain English description of the language recognized by the finite automaton with the state diagram depicted at the start of Question 3. What is it?

Solution: If the input is considered to be the binary expansion of a number, then the automaton accepts exactly those numbers that are divisible by 3.

Idea: Let $a \in \{0,1\}^n$ be the input to the DFA. Normally, we might write that as $a = a_0 a_1 \dots a_{n-1}$ where each symbol $a_i \in \{0,1\}$. Instead, let's label the symbols in reverse so that $a = a_{n-1} a_{n-2} \dots a_0$. Therefore, if we consider the input to be a binary number, then that number can be written as

$$A := a_{n-1} 2^{n-1} + a_{n-1} 2^{n-2} + \dots + a_1 2^1 + a_0 2^0$$

Suppose for the moment that we also had the space in the DFA to write down arbitrary numbers. Notice that we can build this number as we see each new symbol of the input: on receiving input a_i, we multiply our previous number by 2 and add a_i to it. We get a sequence like

$$0 \to a_{n-1} \to 2a_{n-1} + a_{n-2} \to 4a_{n-1} + 2a_{n-2} + a_{n-3} \to \dots \to A$$

The numbers that we're trying to accept are exactly those for which $A = 0 \pmod 3$. But (using a common trick in modular arithmetic), we don't have to wait until the end of the computation to take remainder. We can do this in steps, just like we built up

the number before

$$a_{n-1} \to 2a_{n-1}+a_{n-2} \pmod 3 \to 4a_{n-1}+2a_{n-2}+a_{n-3} \pmod 3 \to \ldots \to A \pmod 3$$

However, there are only three possible remainders mod 3, so we actually only need to be able to count up to 3 (and correspondingly, the states of the DFA are q_0, q_1, q_2). Let's take one example: suppose the DFA is current in state q_1, meaning that the number it had constructed is 1 (mod 3). When we see the next bit of the input, we apply the procedure to build up the number—namely, we multiply by 2 and add the next bit. In other words, the next modulus is $2 * 1 + a_i \pmod 3$. So if $a_i = 0$, then we go to state q_2 and if $a_i = 1$, we go to state q_0. Notice how both of these transitions are in the DFA. The rest of the proof just goes by considering the other possibilities.

4. **Being precise with terminology**:
For each of the following statements, determine if it is true, false, or if the question doesn't even make sense (because the statement isn't well formed or doesn't use terms in ways consistent with definitions from class).

   (a) (*Graded for completeness*) The empty string is in every language.

   Solution: False.

   Explanation: A counterexample is the empty set, which is a language and does not have the empty string as an element. Another counterexample is $\{0\}$ which is the language whose only element is 0.

   (b) (*Graded for completeness*) Σ^* is a language.

   Solution: True.

   Explanation: The definition of a language is a set of strings. Since Σ^* is the set of all strings, it itself is a set of strings, hence a language.

   (c) (*Graded for completeness*) Every language is a regular expression.

   Solution: The question doesn't make much sense.

   Explanation: A related statement that would make sense is "Every language is described by a regular expression". However, this statement is false; we will prove this in the next week or so.

   (d) (*Graded for completeness*) Alphabets are infinite.

Solution: False.

Explanation: The definition of an alphabet is a nonempty *finite* set of characters.

(e) (*Graded for completeness*) There is a finite number $k \in \mathcal{N}$ such that every DFA has fewer than k states.

Solution: False.

Explanation: While each DFA has a finite number of states, there's no upper bound on the specific number of states of any one DFA.