HW3 : Nonregular Languages and Pushdown Automata
Sample Solutions

CSE105Sp23

Assigned questions

1. **Regular or not?** (15 points):
   Fix $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, 2\}$. For each of the languages listed below, prove that it is either regular or nonregular. *Note:* You might find it useful to explore the definition of each set and consider alternate (simpler) ways of stating it.

   For each language that is regular, a complete solution will include a precise definition of a DFA, NFA, or regular expression that recognizes or describes it, along with a brief justification of your construction by explaining the role each state plays in the machine or referring back to relevant definitions.

   For each language that is nonregular, a complete solution will use the pumping lemma to prove it, including appropriate justification related to the specific language.

   (a) (*Graded for correctness*) [1] $L_1 = \{0^n x 1^n \mid n \geq 1, x \in \Sigma^*\}$, a language over Σ.

   Solution: Regular.

   Informally: the set description can be simplified by noticing that the only constraint it imposes is that strings in the language both start with 0 and end in 1. Even though n is used at the beginning and end, there doesn't really need to be a balance of 0s and 1s in strings in the langauge because x itself could have some 0s in front or 1s in the back.

   Formally: This language can be described by the regular expression

   $$0\Sigma^*1$$

   as we prove below by subset inclusion in both directions.

[1] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

If we have a string in $L(0\Sigma^*1)$, then it is of the form $0w1$ for some string $w \in \Sigma^*$. Choosing $n = 1$, it is clear that this string is in L_1.

Conversely, if we have a string in L_1, then it is of the form $0^n x 1^n$ for some $n \geq 1$ and $x \in \Sigma^*$. Since n is positive, we can write this string as $00^{n-1}x1^{n-1}1$. Setting w to be $0^{n-1}x1^{n-1}$, we see that this string is of the form $0w1$, and therefore is a string in $L(0\Sigma^*1)$.

(b) (*Graded for correctness*) $L_2 = \{0^n 1 x 0 1^n \mid n \geq 1, x \in \Sigma^*\}$, a language over Σ.

Solution: Nonregular.

Informally: no DFA can have enough memory to check for a balance between the 0s at the front of the string and the 1s at the back.

Formally: Assume, towards a proof by contradiction, that $L_2 = \{0^n 1 x 0 1^n \mid n \geq 1, x \in \Sigma^*\}$ is regular. Let p be the pumping length given by the pumping lemma. Choose s to be the string $0^p 1 0 1^p$ (x is the empty string). Because s is a member of L_2 and $|s| = p + 2 + p = 2p + 2 > p$, the pumping lemma guarantees that s can be split into three sub-strings as follows: $s = xyz$, where y is nonempty and for any $i \geq 0$ the string xy^iz is in L_2. Additionally, the pumping lemma guarantees that $|xy| \leq p$. Thus (since the first p characters of s are all 0s), each of x and y must consist only of 0s. In other words, there are nonnegative integers k, m, and r such that

$$\begin{aligned} x &= 0^k \\ y &= 0^m \\ z &= 0^r 1 0 1^p \end{aligned}$$

where $0 \leq k \leq p$, $1 \leq m \leq p$ (since y is nonempty), and $r = p - k - m$. Calculating:

$$xyyz = 0^k \;\; 0^m \;\; 0^m \;\; 0^r 1 0 1^p = 0^{p+m} 1 0 1^p$$

Since $m > 0$, $p + m > p$ and $xyyz$ is not in L_2 because the number of 0s at the front of the string is not equal to the number of 1s at the back. Thus, $xyyz$ is not a member of L_2, which is a contradiction with the guarantee that all xy^iz are in L_2. Therefore, L_2 is not regular.

(c) (*Graded for correctness*) Recall that for $L \subseteq \Sigma^*$, we define

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

$L_3 = \textsc{Rep}(\{0^n 1^n \mid n \geq 1\})$, a language over Γ.

Solution: Nonregular.

Informally: since the 2s are used to indicate the beginning and end of each balanced string, a DFA would need to be able to detect the balance, which it can't.

Formally: We will prove that $L_3 = \text{REP}(\{0^n1^n \mid n \geq 1\})$ has no pumping length and therefore cannot be regular (by the pumping lemma). Let p be an arbitrary positive integer. We need to show that it is not a pumping length for L_3. Define s to be the string 20^p1^p2. Notice that s is a member of L_3 because the substring between the one pair of successive 2s is an element of $\{0^n1^n \mid n \geq 1\}$. Moreover, $|s| = 2p + 2$, which is greater than p. Consider any choice of strings x, y, z,

- $s = xyz$
- $|y| > 0$
- $|xy| \leq p$

We will show that there is some $i \geq 0$ for which $xy^iz \notin L_3$ (and hence this choice of s serves as the necessary counterexample to p being a pumping length for L_3). Consider $i = 2$; we will show that $xyyz \notin L_3$. Since $|xy| \leq p$ and y is not the empty string, y must either be 2, 0^m, or 20^m for some $0 < m \leq p$:

- Case 1: $y = 2$ (and hence $x = \varepsilon$, $z = 0^p1^p2$). Then, $xyyz = \varepsilon\ 2\ 2\ 0^p1^p2$ which is not in L_3 because the first pair of successive 2s is adjacent, i.e. the string between them is the empty string, but the empty string is not in $\{0^n1^n \mid n \geq 1\}$ (the smallest length of a string in that language is when $n = 1$, the string 01). Hence $xyyz \notin \text{REP}(\{0^n1^n \mid n \geq 1\})$.
- Case 2: $y = 20^m$ for some $0 < m < p$ (and hence $x = \varepsilon$, $z = 0^{p-m}1^p2$). Then, $xyyz = \varepsilon\ 20^m\ 20^m\ 0^{p-m}1^p2$ which is not in L_3 because the first pair of successive 2s has 0^m between them, but 0^m is not in $\{0^n1^n \mid n \geq 1\}$ (all strings in this set have at least one 0 and at least one 1). Hence $xyyz \notin \text{REP}(\{0^n1^n \mid n \geq 1\})$.
- Case 3: $y = 0^m$ for some $0 < m < p$ (and $x = 20^k$ for some $k \geq 0$, $z = 0^{p-m-k}1^p2$). Then, $xyyz = 20^k\ 0^m\ 0^m\ 0^{p-m-k}1^p2$ which is not in L_3 because the string between the successive 2s is $0^{k+2m+p-m-k}1^p = 0^{p+m}1^p$, which is not in $\{0^n1^n \mid n \geq 1\}$ because $m > 0$ so $p + m \neq p$. Therefore, $xyyz$ is not a member of L_3.

Since we have shown that for any positive integer p there is a counterexamples that disprove that p is a pumping length for L_3, L_3 has no pumping length. Therefore, by the pumping lemma, L_3 is not regular.

2. **Properties of nonregular languages** (15 points):
Prove or disprove each of the following statements. In other words, decide whether each statement is true or false and justify your decision. Let $\Sigma = \{0, 1\}$ and let $\Gamma = \{0, 1, 2\}$.

   (a) (*Graded for correctness*) For all languages L, K over Σ, if L is nonregular and K is finite, then $L - K$ is nonregular. Recall: $L - K = \{w \in \Sigma^* \mid w \in L \text{ and } w \notin K\}$.

Solution: True. Assume by contradiction that $L - K$, containing strings that exist in L but *not* K, is regular. $K \cap L$, which contains strings that exist in L *and* K, is regular because it is finite (since K is finite, we know that $|K \cap L| \leq |K|$ because at most all strings in K could also be in L). Therefore, $(L - K) \cup (K \cap L) = L$ is regular since regular languages are closed under union. This contradicts that L is nonregular.

(b) (*Graded for correctness*) Every infinite language over Σ where each string in the language has an equal number of 0's and 1's is nonregular.

Solution: False. Consider the counterexample $L((01)^*)$, which is an infinite language since the sequence (01) can be repeated any number of times, and every string in the language will have an equal number of 0's and 1's. However, this language is regular because it can be described by the regular expression, $(01)^*$.

(c) (*Graded for correctness*) Recall that for language K over Γ,

$$\textsc{Substring}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}.$$

For every nonregular language K over Γ, $\textsc{Substring}(K)$ is nonregular.

Solution: False. Consider the counterexample $K := \{0^n 1^n \mid n \geq 0\}$, which is nonregular. For any $m, p \geq 0$, there exist $a = 0^{n-m}$, $b = 1^{n-p}$, and $w = 0^m 1^p$ such that $awb \in K$. In other words, $\textsc{Substring}(K)$ includes all $w = 0^m 1^p$ where $m, p \geq 0$, which can be written as 0^*1^*. Therefore, The result of $\textsc{Substring}(K)$ can be described by the regular expression 0^*1^*, and thus is regular.

3. **Pumping dilemma** (8 points):
Your friend claims that the Pumping Lemma is useless for proving that an infinite language $K \subseteq \Sigma^*$ is not regular. Their logic goes like this

(Step 1) Suppose that K is regular. It can be recognized by a DFA $M = (Q, \Sigma, \delta, q_0, F)$.

(Step 2) For arbitrary DFA M, the pumping length p is at least $|Q|$.

(Step 3) However, for every integer $n \geq |Q|$, there exists a machine $M' = (Q', \Sigma, \delta', q_0', F')$ such that $L(M') = L(M) = K$ and $|Q'| = n$.

(Step 4) Therefore, the Pumping Lemma cannot be used to pump any string of finite length since its pumping length might be arbitrarily large.

Below, we will examine the steps above in detail. Justify your answer to each part.

(a) (*Graded for completeness*) [2] (Step 1): Is this statement true? In other words, just because we're assuming that K is regular a regular language, does it mean we can assume there is a DFA that recognizes it?

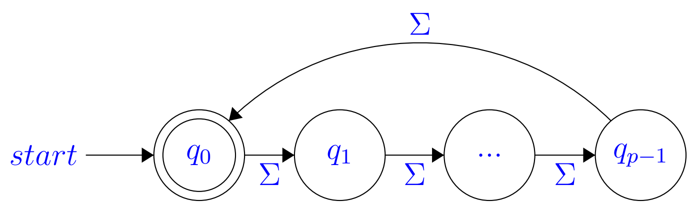
Solution: True, by definition. A language is regular if and only if it is recognized by a DFA, NFA, or a regular expression.

(b) (*Graded for completeness*) (Step 2): In general, it's true that the smallest the pumping length of a language recognized by a DFA with states Q can be is $|Q|$. Prove this by finding a specific infinite language K and a DFA recognizing it that cannot have pumping length smaller than $|Q|$.

Solution: Consider the language of strings that have a length that is divisible by p:

$L = \{w \mid w \in \Sigma^* \text{ and } |w| \mod p = 0\}$

The DFA has p states, with the first state as the only accepting state, as follows:



We cannot pump a string in the language with fewer than p symbols, since the length of string would no longer be a multiple of p and would not end in the final state of the DFA.

(c) (*Graded for completeness*) (Step 3): This step is correct; prove the stated version of this statement: For every integer $n \geq |Q|$, there exists a machine $M' = (Q', \Sigma, \delta', q_0', F')$ such that $L(M') = L(M)$ and $|Q'| = n$.

Solution: It is always possible to create a machine to accept the same language with a greater number of states by adding some unreachable dummy states that loop on themselves. Since these states are unreachable, this would not have any effect on the set of strings accepted by the DFA.

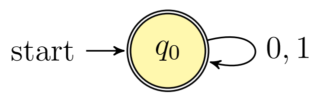
(*Challenge; not graded*): Define a *cycle* to be a sequence of *distinct* states $q_1, q_2, \ldots, q_m$ such that

$$\delta(q_1, \sigma_1) = q_2, \qquad \delta(q_2, \sigma_2) = q_3, \qquad \ldots, \qquad \delta(q_m, \sigma_m) = q_1,$$

[2] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

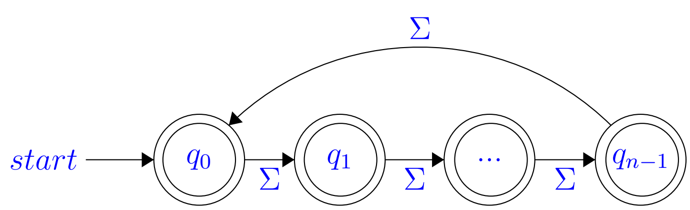
where $\sigma_1, \sigma_2, \ldots, \sigma_m \in \Sigma$ are symbols in the alphabet. An objection to the statement in (Step 3) is that the proof of the Pumping Lemma depends on the length of the cycles in the DFA rather than the number of states. That is, increasing the number of states in your DFA might not increase the pumping length because the length of the smallest cycle stays the same. Nevertheless, a version of your friend's statement is still true whenever you impose this additional cycle constraint: for every integer $n \geq |Q|$, there exists a machine $M' = (Q', \Sigma, \delta', q_0', F')$ such that $L(M') = L(M)$ and the length of the smallest cycle in the M' is at least n.

Your task is to show that even this more general statement is true for the simple language Σ^* recognized by the DFA below:



For all $n \geq 1$, define a DFA for this language where the length of the smallest cycle is n.

Solution: Define a DFA that has n states $Q = \{q_0, \ldots, q_{n-1}\}$, and each only has transitions to the next one in the sequence (i.e., $\delta(q_i, \Sigma) = q_{i \pmod n}$). So, the only cycle in the DFA is of size n. In order for the DFA to accept $\sum^*$, every state should be the accepting states, $F = Q$.



We can also show even more generally that we can *always* increase the length of the smallest cycle regardless of the language: Take the original DFA M and copy it n times to get machines $M_1, M_2, \ldots, M_n$. Modify the transition function so that a transition in M_i actually takes you to the state in $M_{i \pmod n}$. Every cycle must go through all n copies of the machine and so has length at least n.

(d) (*Graded for completeness*) (Step 4): Describe why this statement is true/false/misleading.

Solution: The pumping length is a property of the *language*, not all DFAs that recognize it. As discussed in the challenge question, the DFA could have arbitrary size of the smallest cycle, but the pumping length is not dependent on it. Also in 3(c), we showed that pumping length is independent of number of states in a DFA recognizing the language. Hence, there could be many ways of drawing a DFA for a language with variable number of states, but the pumping length is property of the language rather than DFAs.