CSCI 1010 - Fall'25 Theory of Computation Lorenzo De Stefani

HW2

Due: September 25, 2025

Reminder: Submit your assignment on Gradescope by the due date. Submissions must be typeset. Each page should include work for only one problem (i.e., make a new page/new pages for each problem). See the course syllabus for the late policy.

While collaboration is encouraged, please remember not to take away notes from any labs or collaboration sessions. Your work should be your own. Use of other third-party resources is strictly forbidden.

Please monitor Ed discussion, as we will post clarifications of questions there.

Problem 1

Recall that a **pumping length** for a language A is a positive integer p such that all strings $s \in A$ with $|s| \geq p$ can be written in the form xyz, where:

1. $|xy| \leq p$,
2. $|y| \geq 1$,
3. and $xy^iz \in A$ for all $i \geq 0$.

The **pumping lemma** states that every regular language has a pumping length. The **minimum pumping length** of a language A, $p_{\min}$, is the smallest pumping length for A. Note that this implies that every integer $p \geq p_{\min}$ is also a valid pumping length for A.

For each of the following languages, give the minimum pumping length $p_{\min}$ and prove your answer.

1. bb^*
2. $(ab)^*$
3. $L_c = \{w \mid w \in \{a, b\}^* \text{ and } w \text{ ends in } ab\}$
4. $L_d = \{aabb, bbaa, abab, a\}$
5. $L_e = \{w \mid w \in \{a, b\}^* \text{ and } w \text{ does not end in } aa\}$

Solution:

Solution

1. $p_{min} = 2$. For any string s in bb^* such that $|s| \geq 2$, we can write $s = xyz$, where $x = b$, $y = b$ and z is the rest of the string. By pumping y we just change the number of b's in the string (adding as many as needed, or removing at most one), which still keeps it in the language. Suppose there was a pumping length $p = 1$. Then $x = \epsilon$, $y = b$, $z =$ the rest. But for $s = b$, we get $x = \epsilon$, $y = b$, $z = \epsilon$, so $xy^0z = xz = \epsilon$ should also belong to the language, which is false.

2. $p_{min} = 2$. The pumping length cannot be 1 since we know $|y| \geq 1$ and $|xy| \leq p$; if $p = 1$ then the previous conditions imply that $y = a$. But if we pump y then we increase the number of a's in front of b's which would no longer be in the language. $p_{min} = 2$ works though since we can set $y = ab$ and pumping just corresponds to performing the star operation.

3. $p_{min} = 3$. Take $x = \epsilon$, $y =$ the first character of the string, $z =$ the rest of the string. By pumping y, we never change the string's last two characters, so it remains in the language. The pumping length cannot be two. If $p = 2$ then ab, which belongs to the language, can be written as $ab = xyz$ such that $|y| = a$ and $z = b$. If we set $i = 0$ in xy^iz, then our new string becomes b, which is not in the language.

4. $p_{min} = 5$. Since this language is finite, no string in it is pumpable. If there were a pumpable string in it, pumping that string would generate an infinite number of different elements of the language. So any pumping length p satisfies $p > |w|, \forall w$ in the language. Thus the minimum pumping length is five.

5. $p_{min} = 2$. Again, we can take $x = \epsilon$, $y =$ the first character of the string, $z =$ the rest. Firstly, if $|s| > 2$, then by pumping y, we never change the string's last two characters. Second, if $|s| = 2$, then xy^kz has the same last two characters for $k \geq 1$, and xz has only one character and therefore does not end in aa for $k = 0$. But $|s|$ cannot be 1, because then if $s = a$, the only valid way to pump s is to assign $s = \epsilon, y = a, z = \epsilon$, meaning $xy^2z = aa \notin L$.

Problem 2

Argue whether each of the following languages is regular or not. If you believe a language to be regular, prove that it is so by providing a correct regular expression for it with a proof of correctness. If you believe a language is not regular, prove it using the pumping lemma for regular languages.

1. $L_1 = \{a^k | \text{k is a prime number}\}$

2. Let $B = \{1^k w | w \in \{0,1\}^*$ and w contains at least k 1s, for $k \geq 1\}$. Hint: Try out some strings to see what does and doesn't belong to B, in order to find another simpler way of thinking about B.

Solution:

1. This language is not regular.

   Proof : Assume towards contradiction that L_1 is regular. This means the pumping lemma must hold for L_1. Now, consider the string $s = a^p$, where p is a prime number. s is therefore in L_1. Next, when considering the pumping lemma, from the two conditions that $|y| > 0$ and $|xy| \leq$ p, we know the substring y has somewhere in the range of 1 to p a's. Define k as the number of a's in our substring y. When we pump the string xy^iz, our resulting string is then $a^{(p+(k*(i-1)))}$ (Note: It is i-1 since i represents the occurrences of the string and we are using i here to track how many a's we have added to the string). Therefore, if we choose i to be p + 1, the number of a's will then be a multiple of p and our string will not be a prime number of a's (and therefore be out of the language). This then yields a contradiction to our assumption that L_1 is regular since we have found a string of length p where there is no way to split it such that the pumping lemma holds. L_1 must not be a regular language.

2. This language is regular.

   Regular Expression :

   $1(1^*0^*)^*1(1^*0^*)^*$

   Brief Justification : Since the only restriction on the number of 1's in the respective parts of the string are that k is greater than or equal to one and that w must have at least that many 1's, then any string

with at least two 1's and that starts with a 1 could be in our language. (Note that w can start at any point of seeing 1's after the first 1 at the beginning since there is no restriction on what w must start with).

Problem 3

Consider the language over the alphabet $\Sigma = \{x, y, z\}$:

$L = \{xy^n z^n \ : \ n \geq 0\} \cup \{x^k w \ : \ k \neq 1 \ and \ w \in \Sigma^* \ and \ the \ first \ symbol \ of \ w \ is \ not \ an \ `x'\}$

1. Prove that L satisfies the pumping lemma property.
2. Is L regular? Provide a proof for your answer.

Solution:

Let $L_1 = \{xy^n z^n \ : \ n \geq 0\}$ and $L_2 = \{x^k w \ : \ k \neq 1 \ and \ w \in \Sigma^* \ and \ the \ first \ symbol \ of \ w \ is \ not \ an \ `x'\}$ so that $L = L_1 \cup L_2$.

1. We claim that L satisfies the pumping lemma with pumping length 3. Let $s \in L$ such that $|s| \geq 3$. Then $s \in L_1$ or $s \in L_2$. We will show that in both cases, we can pump a substring in the first three characters.

   If s in L_1, it can be written as $xy^i z^i$ where $i \geq 1$. We can divide s into substrings a, b, c such that $a = \epsilon$, $b = x$ and $c = y^i z^i$. For any $n \in \mathbb{N}$, we have that $ab^n c = x^n y^i z^i$. If $n = 0$, $ab^0 c = y^i z^i \in L_2$. If $n = 1$, $abc = s \in L_1$. If $n > 1$, $ab^n c = x^n y^i z^i \in L_2$.

   If s in L_2, it can be written as $x^k w$ where w starts with a y or a z and $k \neq 1$. If $k = 0$, then $s = w$, and pumping the first character of w will yield a string that is still in L_2 since it begins with 0 xs. If $k = 2$, then we can divide s into $a = \epsilon$, $b = xx$, and $c = w$. For any $n \in \mathbb{N}$, we have that $ab^n c = (xx)^n w$. If $n = 0$, $ab^0 c = w \in L_2$. If $n > 0$, $ab^n c = (xx)^n w \in L_2$. Finally, if $k > 2$, we can divide s into $a = \epsilon$, $b = x$, and $c = w$. For any $n \in \mathbb{N}$, we have that $ab^n c = (x)^n x^{k-1} w$. If $n = 0$, $ab^0 c = x^{k-1} w \in L_2$. If $n > 0$, $ab^n c = x^n x^{k-1} w \in L_2$.

   Therefore, $\forall s \in L$, s can be pumped and stay in L!

2. We will show that L is not regular, despite satisfying the pumping lemma. The general strategy will be: assume for the sake of contradiction that L is regular. Then, $L_1 = L \cap \overline{L_2}$. $\overline{L_2}$ is regular, and

regular languages are closed under intersection so L_1 must be regular too. However, L_1 is not regular.

Therefore, we want to show that : $L_1 = L \cap \overline{L_2}$, L_1 is not regular and $\overline{L_2}$ is regular.

First, though, we will need to show that regular languages are closed under both intersection and complement.

Complement: We will proceed with a proof by construction. Let M be a DFA that recognizes the language A. We will construct a DFA M' using M to accept $\overline{A}$. Let the language that M' recognizes be A'; we will show that $A' = \overline{A}$. We can do this simply by making M' exactly equal to M, except we change all the accepting states of M to non-accepting states and change all the non-accepting states of M to accepting states. Consider an arbitrary string $w \in A$ (which means $w \notin \overline{A}$). Since M accepts w, this means that if w is provided as an input to M, M halts in an accepting state. Denote this state by q_i. Then, since M' has the same starting state and transition function as M, this means that when given w, M' also halts in state q_i. But, by construction, q_i is not accepting in M', so M' does not accept w. So, we have $w \notin A$ and $w \in \overline{A}$. We can similarly consider an arbitrary string $w \notin A$ (which means $w \in \overline{A}$). When M is given w as input, it halts in a non-accepting state. Denote this state by q_j. When M' is given w as input, it also halts in state q_j. By construction, q_j is accepting in M', so M' accepts w. So, we have $w \in \overline{A}$ and $w \notin A$. This means that $A' = \{w \in \Sigma^* | w \notin A\} = \overline{A}$.

Since we have constructed a DFA that recognizes $\overline{A}$ given a DFA that recognizes A, we have shown that $\overline{A}$ is regular if A is regular, as desired, using the fact that a language is regular if and only if there is a DFA that recognizes it (note that if given an NFA we can construct an equivalent DFA).

Intersection: Since regular languages are closed under complement, we have that $\overline{L_1}$ and $\overline{L_2}$ are both regular, since L_1 and L_2 are regular. Then, since regular languages are closed under union, we have that $\overline{L_1} \cup \overline{L_2}$ is regular. Then, by DeMorgan's laws, we know that $\overline{L_1} \cup \overline{L_2} = \overline{(L_1 \cap L_2)}$. So, $\overline{(L_1 \cap L_2)}$ is regular. Then, we use the fact that regular languages are closed under complement again to get that $\overline{\left(\overline{(L_1 \cap L_2)}\right)} = L_1 \cap L_2 = L$ is regular, since L is the intersection of L_1 and L_2, by construction.

Alternatively, you can prove closure under intersection without using DeMorgan's laws. Consider the cartesian product of two DFAs' states, and define the accepting states to be the cartesian product of the individual DFAs' accepting states (see the proof for union on page 46 of Sipser, and intersection follows very similarly).

The following is just a sketch of the remaining proof:

- First you show that L_1 and L_2 are disjoint, which justifies the claim that $L_1 = L \cap \overline{L_2}$.
- Then you show that L_2 is regular. You can provide a regex or a DFA as you wish.
- Then you show that L_1 is not regular by using the pumping lemma and considering the problematic string xy^pz^p.