CSCI 1010 - Fall'26 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.

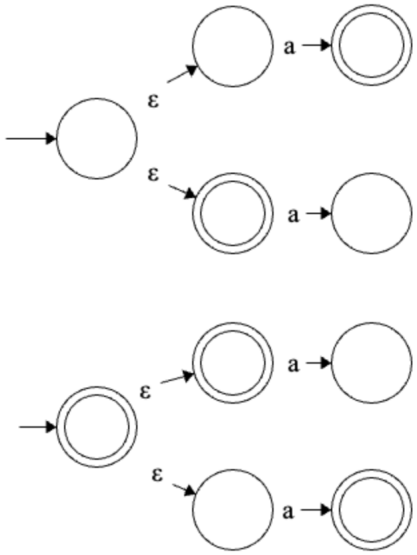
Problem 1

(a) Show by giving an example that if M is an NFA that recognizes language C, swapping the accept and non-accept states in M doesn't necessarily yield a new NFA that recognizes the complement of C.

(b) Is the class of languages recognized by NFAs closed under complement? Explain your answer.

Solution:

1. Consider the following NFA M and the corresponding NFA M' with accept and non-accept states swapped:



M accepts the string 'a', as does M'. Therefore swapping the accept and non-accept states of an NFA does not necessarily yield a new NFA which recognizes the complement of the language recognized by the original.

2. As shown in class, the class of languages recognized by NFAs is the same as the class of languages recognized by DFAs, which is the class of regular languages. We have shown that regular languages are closed under complement, so therefore the class of languages recognized by NFAs is also closed under complement.

Problem 2

Let Σ and Δ be finite alphabets of symbols. A *homomorphism* is a function $h : \Sigma^* \to \Delta^*$ defined as follows:

(a) $h(\epsilon) = \epsilon$;

(b) For any $a \in \Sigma$, $h(a) \in \Delta^*$;

(c) For $a \in \Sigma^*$ such that $a = a_1 a_2 \dots a_n$ with $n \geq 2$,

$$h(a) = h(a_1)h(a_2)\dots h(a_n).$$

That is, a homomorphism is a function from strings to strings that "respects" concatenation: for any $x, y \in \Sigma^*$, $h(xy) = h(x)h(y)$.

Example: $h : \{0,1\} \to \{a,b\}^*$ where $h(0) = ab$ and $h(1) = ba$. Then $h(0011) = ababbaba$.

Given a homomorphism $h : \Sigma^* \to \Delta^*$ and a language $L \subseteq \Sigma^*$, we define the homomorphism of a language L as $h(L) = \{h(w) \mid w \in L\} \subseteq \Delta^*$.

Prove that the set of regular languages is *closed under homomorphism*, that is, for any given regular language L, $h(L)$ is a regular language.

Solution: We will use the representation of regular languages in terms of regular expressions to argue this. Here is the plan of the proof:

1. Define homomorphism as an operation on regular expressions
2. Show that $L(h(R)) = h(L(R))$
3. Let R be such that $L = L(R)$. Let $R' = h(R)$. Then $h(L) = L(R')$.

For a regular expression R, let $h(R)$ be the regular expression obtained by replacing each occurrence of $a \in \Sigma$ in R by the string $h(a)$. Formally:

- $h(\epsilon) = \epsilon$
- $h(a)$
- $h(\emptyset)$
- $h(R_1 R_2) = h(R_1)h(R_2)$
- $h(R_1 \cup R_2) = h(R_1) \cup h(R_2)$
- $h(R^*) = h(R)^*$

Claim: For any regular expression R, $L(h(R)) = h(L(R))$.

Proof. By induction on the number of operations in R

- Base Cases: For $R = \epsilon$ or $R = \emptyset$, $h(R) = R$ and, trivially, $h(L(R)) = L(R)$. For $R = a$, $L(R) = a$ and $h(L(R)) = h(a) = L(h(a)) = L(h(R))$.
- Induction Step: For $R = R_1 \cup R_2$, observe that $h(R) = h(R_1) \cup h(R_2)$ and $h(L(R)) = h(L(R_1) \cup L(R_2)) = h(L(R_1)) \cup h(L(R_2))$. By inductive hypothesis, $h(L(R_i)) = L(h(R_i))$ and so $h(L(R)) = L(h(R_1) \cup h(R_2))$. The other cases $R = R_1 R_2$ and $R = R_1$ follow from similar reasoning.

Problem 3

We have already seen that the language

$$L = \left\{ w \;\middle|\; \begin{array}{c} w \text{ contains an equal number of} \\ \text{occurrences of the substrings 01 and 10} \end{array} \right\}.$$

is regular. Now let

$$L' = \left\{ w \;\middle|\; \begin{array}{c} w \text{ contains an equal number of} \\ \text{occurrences of the substrings 010 and 101} \end{array} \right\}.$$

Show that L' is not regular.

Hint: if you're struggling to find a good string, consider one whose first segment contains only one of the substrings 010 or 101.

Solution: Suppose for contradiction that L' is regular with pumping length p. Consider the string $w = (010)^p(101)^p \in L$ with $(p+1)$ occurrences of each of the two substrings. Then by the pumping lemma, we can divide w into substrings $w = xyz$ satisfying the pumping properties; namely, $|xy| < p$, $y \neq \epsilon$, and $xy^k z \in L'$ for $k \geq 0$. Since $|xy| < p$, the substring y must be entirely contained within the half $w_1 = (010)^p$ containing no copies of 101. We will proceed by cases on y.

- **Case 1 ($1 \in y$).** Since there can be no more occurrences of 010 in w_1 than there are distinct 1's, if y held a 1, then the string xy^0z would have at most p occurrences of 010 and at least $p + 1$ occurrences of 101. Hence, y cannot contain 1 as a substring, so y must be composed only of 0's.
- **Case 2 ($y = 0$).** We consider two subcases. If y is the first 0 of w, then the string $xy^0z = 10(010)^{p-1}(101)^p$ has p copies of 010 and

$(p+1)$ of 101, so it is not contained in L'. Otherwise, xy^0z is of the form $(010)^i01(010)^j(101)^p$, where $j > 0$ and $i + j = p$. This string has $(p+1)$ copies of 010 and $(p+2)$ of 101, so it again cannot be contained in L'.

- **Case 3 ($y = 00$).** If $y = 00$, then the string xy^0z is of the form $(010)^i11(010)^j(101)^p$, where $i + j = p - 1$. This has p copies of 010 and $(p+1)$ of 101, so again, it cannot be contained in L'.

As we have seen, no choice of y allows w to be pumped, so L' is not regular.

Problem 4

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

(a) Let

$$L_1 = \left\{ x^k w \;\middle|\; \begin{array}{c} k \neq 1 \text{ and } w \in \Sigma^* \text{ and the} \\ \text{first symbol of } w \text{ is not an '}x\text{'} \end{array} \right\}.$$

Prove that L_1 is regular. Showing a correct DFA/NFA/RegEx is sufficient.

(b) Let

$$L_2 = \{xy^n z^n \mid n \geq 0\}.$$

Prove that L_2 is not regular.

(c) Prove that $L = L_1 \cup L_2$ satisfies the pumping lemma property.

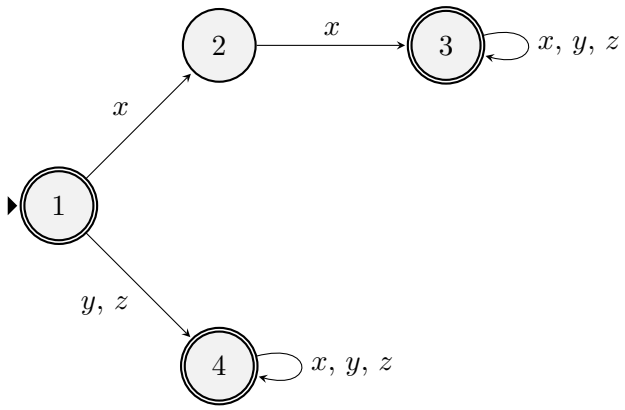
(d) Is L regular? Provide a proof for your answer.

Hint: it might help you to show that regular languages are closed under complement and/or intersection...

Solution:

(a) *Possible RegEx:* $\epsilon \cup y(xyz)^* \cup z(xyz)^* \cup xx(xyz)^*$

Possible DFA:



(b) Suppose for contradiction that L_1 is regular with pumping length p, and consider the string $xy^pz^p \in L_1$. Since L_1 is regular, there exists some choice of substrings $abc = xy^pz^p$ satisfying the pumping property. First, notice that a cannot be empty, since this would give us $b = xy^k$ for some $0 \leq k \leq p-1$, and hence $ab^0c = y^{p-k}z^p \notin L_1$. Therefore, the initial x is contained in the substring a, and $b = y^k$ for some $1 \leq k \leq p-1$. But then $ab^0c = xy^{p-k}z^p$, and since $p - k \neq p$, the string is again not in L_1. Therefore, L_1 does not satisfy the pumping property, so it is not regular.

(c) 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^iz^i where $i \geq 1$. We can divide s into substrings a, b, c such that $a = \epsilon$, $b = x$ and $c = y^iz^i$. For any $n \in \mathbb{N}$, we have that $ab^nc = x^ny^iz^i$. If $n = 0$, $ab^0c = y^iz^i \in L_2$. If $n = 1$, $abc = s \in L_1$. If $n > 1$, $ab^nc = x^ny^iz^i \in L_2$.

If s in L_2, it can be written as x^kw 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 $ab^nc = (xx)^nw$. If $n = 0$, $ab^0c = w \in L_2$. If $n > 0$, $ab^nc = (xx)^nw \in L_2$. 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 $ab^nc = (x)^nx^{k-1}w$. If $n = 0$, $ab^0c = x^{k-1}w \in L_2$. If $n > 0$, $ab^nc = x^nx^{k-1}w \in L_2$.

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

(d) 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).

But $L_1 = L \cap \overline{L_2}$ is not regular, as we have shown; thus, L cannot have been regular.