Computer Science Theory : Homework 1

Question 1

1. Let the DFA described by the state diagram above be called D
The formal definition (5 tuple) of D is $(Q, \Sigma, q_0, F, \delta)$
where $Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b, c\}$
$q_0 = q_0$
$F = \{q_1\}$

δ	a	b	c
q_0	q_1	q_3	q_3
q_1	q_1	q_2	q_3
q_2	q_1	q_2	q_3
q_3	q_3	q_3	q_3

2. → The language recognized by this DFA is all non-empty strings over {a, b, c} that begin with 'a', end with 'a', and contain no 'c'

→ This description is true because from the DFA we may notice that any string that begins with 'b' and 'c' always end in a non-accept state. Therefore, strings that are recognized by this DFA can only begin with 'a'. Additionally, we may notice that if a string begins with 'a', it will only be accepted if it ends with 'a'. If the string has any subsequent 'b', it will still have to end with 'a' for it to be accepted. Once a 'c' is found, we will immediately be sent to a non-accept state where we cannot return to an accept state.

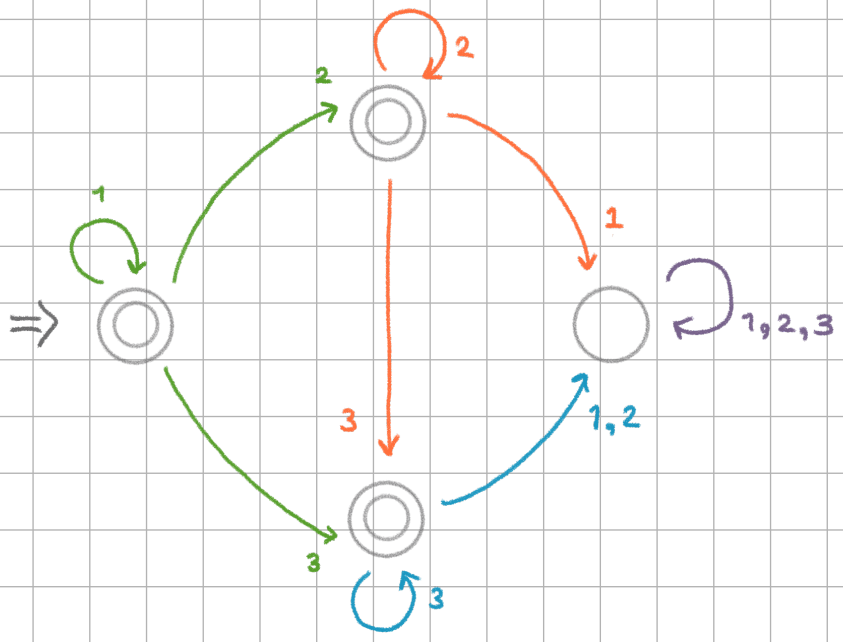
Question 2

1. → The language recognized by this DFA is all non-empty strings over {1, 2} that contain exactly 2 twos and 2 ones and exactly 4 characters in total.

→ This description is true because from the DFA we may notice that there is only one accept state and it is positioned such that we can only reach it after the 4th character is read in and if there is any character that comes after that we will reach a a non-accept state where we cannot return to the only accept state. Additionally, we may also notice that the DFA is arranged such that as long as there's no more than than 2 ones and no more than 2 twos in a given string, we can still reach the accept state. In other words, the string must have exactly 2 ones and 2 twos for it to reach the only accept state. However, once this condition is violated, the string will immediately reach a non-accept state where we cannot get back to the only accept state.

Question 2

2.



→ This DFA recognizes the specified language because for each state if the next character is less than the previous character, then we will be sent to a non-accept state. If the next character is either greater than or equal to the previous character, then we will remain at the current state or get sent to one of the accept states — either way you will end up in an accept state. As such, this DFA recognizes the language over the alphabet $\Sigma = \{1,2,3\}$ consisting of all strings of digits that never decrease.

Note that for this DFA there are 3 accept states, each representing 1, 2, and 3. Whenever a condition is violated (decreasing digits), we get sent to the only non-accept state.

Question 3

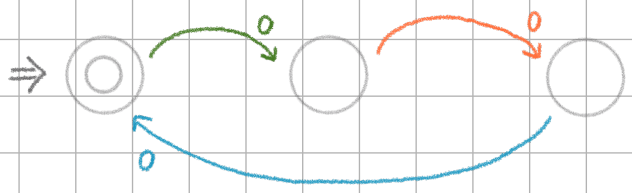
1. To draw a state diagram for an NFA on the alphabet $\Sigma = \{0\}$ that recognizes the language:

$L = \{ w \mid |w| \text{ is divisible by 3 or by 5 (or by both 3 and 5)} \}$

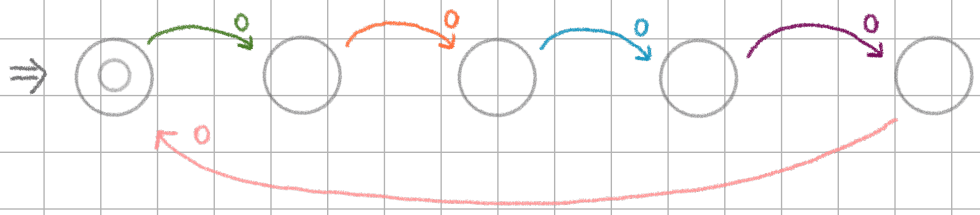
One technique that we know is breaking down the problem into 2 parts (or 2 DFAs) and attach them together to form an NFA with ε transitions

From the given problem, we can create the following DFAs:

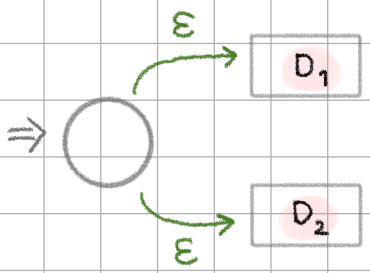
D_1 (for the string whose length is divisible by 3):



D_2 (for the string whose length is divisible by 5):



Now, we can build an NFA from D_1 and D_2:

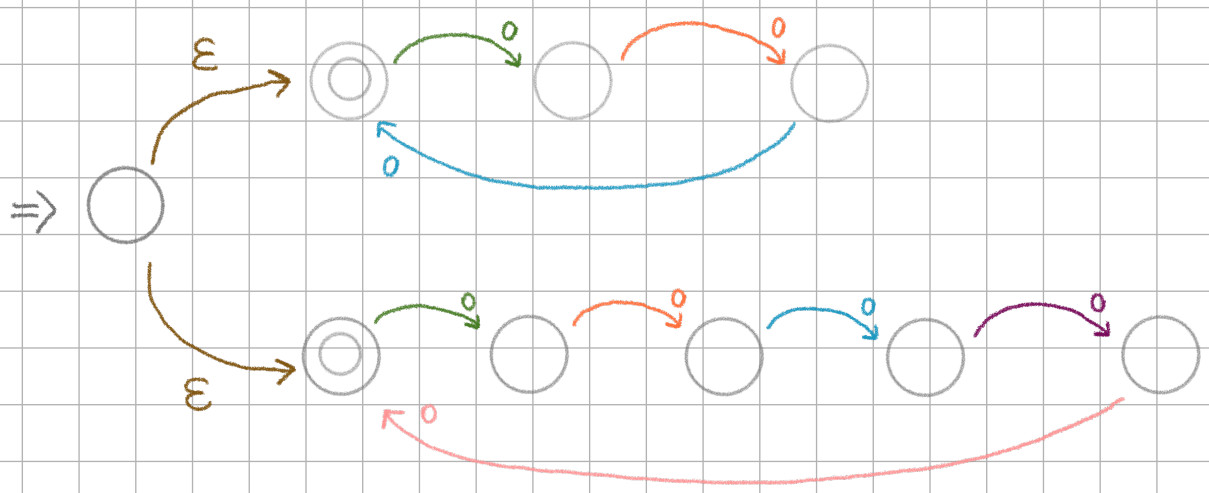


(Continue on the next page...)

Question 3

1. (continue...)

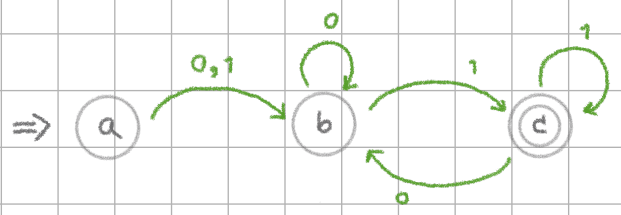
More specifically, we will get this NFA:



→ This NFA recognizes the specified language, L, because the ε transitions allow us to do two computations simultaneously. As such, for each string, the NFA will check if the length is divisible by 3 or 5. The top ε branch will determine if the length is divisible by 3, whereas the bottom ε branch will determine if the length is divisible by 5. The NFA will accept as long as we end in at least one of the accept states, meaning that if the length is divisible by 3 OR 5 or 3 AND 5 the NFA will accept the string.

Question 3

2. Below is a DFA that recognizes the same language as the NFA from the question:

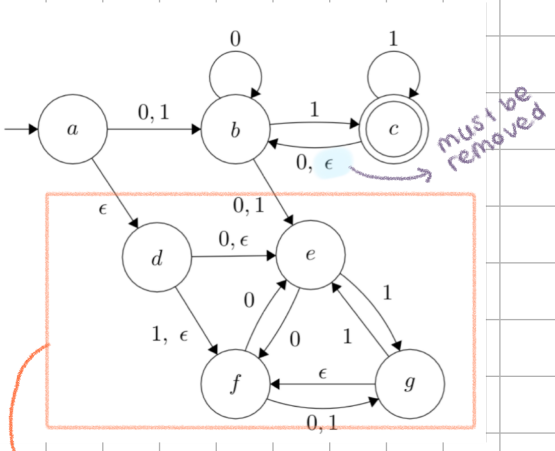


This DFA captures the same language as the NFA because the NFA seems complex but can be very much simplified. If we take a look at the NFA, we may notice that the ε branch from state a must be taken immediately before the first character is read in, yet if we look closely, we may observe that anything that branches out from this ε edge will not lead to an accept state. In fact, there is only one accept state which can only be reached if we take the branch after reading the first symbol as either 0 or 1. In other words, we can omit the bottom half of the NFA when constructing the DFA (see diagram).

Additionally, we may also notice that there's an extra ε from state c to state b. When constructing a DFA, this ε should also be removed because it would lead to a redundant state.

Alternatively, we can approach this problem by deciphering the strings this NFA accepts and building a DFA based on that. By testing some string inputs, we found that this NFA accepts any string of the length of at least 2 that ends with the symbol 1. As such, we build a DFA that accepts this language.

NFA:



This part will never reach the accept state and can thus be omitted

Question 4

1. From the question, we have regular languages $L(D_1)$ and $L(D_2)$
We want to show that $L(D_1) \odot L(D_2)$ is also regular

We also know that D_1 and D_2 recognizes $L(D_1)$ and $L(D_2)$, respectively

To prove that $L(D_1) \odot L(D_2)$ is also regular, we have to demonstrate that a new DFA D_3 recognizes $L(D_1) \odot L(D_2)$

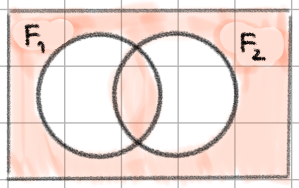
Proof by Construction

We have to construct D_3 from D_1 and D_2
Let D_1 recognizes $L(D_1)$ where $D_1 = (Q_1, \Sigma, \delta_1, q_{f1}, F_1)$
Let D_2 recognizes $L(D_2)$ where $D_2 = (Q_2, \Sigma, \delta_2, q_{f2}, F_2)$

Construct D_3 to recognize $L(D_1) \odot L(D_2)$ where
$D_3 = (Q, \Sigma, \delta, q_{f0}, F)$

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. Σ is the same as in D_1 and D_2
3. δ : For each $(r_1, r_2) \in Q$ and $a \in \Sigma$
$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
4. $q_{f0} = (q_{f1}, q_{f2})$
5. $F = \{(r_1, r_2) \mid (r_1 \in F_1 \text{ and } r_2 \in F_2) \text{ or } (r_1 \in \overline{F_1} \text{ and } r_2 \in \overline{F_2})\}$

XNOR :



$\Rightarrow$ Translated from :
$A \odot B := \{x \mid x \in (A \cap B) \text{ or } x \in (\overline{A} \cap \overline{B})\}$

* Note : I used this venn diagram to construct the F of D_3

This concludes the construction of the new DFA D_3 that recognizes $L(D_1) \odot L(D_2)$

$\therefore$ We have just shown that $L(D_1) \odot L(D_2)$ is regular, thereby proving that the class of regular languages is closed under the $\odot$ operation $\square$

Question 4

2. **Goal:** If A and B are regular, $A \odot B$ is also regular
(and thus the class of regular languages is closed under $\odot$)

Proof: Suppose that A and B are regular

① A, B regular $\Rightarrow$ $A \cap B$ is regular
(regular languages are closed under intersection)

② A, B regular $\Rightarrow$ $\bar{A}$, $\bar{B}$ are regular
(regular languages are closed under complement)

③ $\bar{A}$, $\bar{B}$ regular $\Rightarrow$ $\bar{A} \cap \bar{B}$ are regular
(regular languages are closed under intersection)

④ $(A \cap B)$, $(\bar{A} \cap \bar{B})$ are regular $\Rightarrow$ $(A \cap B) \cup (\bar{A} \cap \bar{B})$ is regular
(regular languages are closed under union)

Since $A \odot B := \{x \mid x \in (A \cap B) \text{ or } x \in (\bar{A} \cap \bar{B})\}$
$:= \{x \mid x \in (A \cap B) \cup (\bar{A} \cap \bar{B})\}$
and $(A \cap B) \cup (\bar{A} \cap \bar{B})$ is regular,

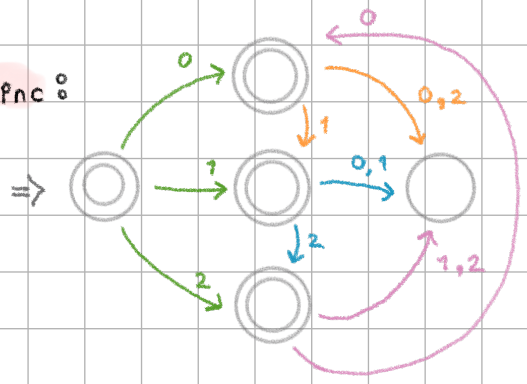
Therefore, $A \odot B$ is regular, thereby proving that the class of regular languages is closed under the $\odot$ operation. □

Question 5

a) Show that the following languages are regular:
Recall that a language is called regular if some finite automaton (DFAs/NFAs) recognizes it.

i) L_{inc} $\Sigma = \{0, 1, 2\}$

D_{inc}:

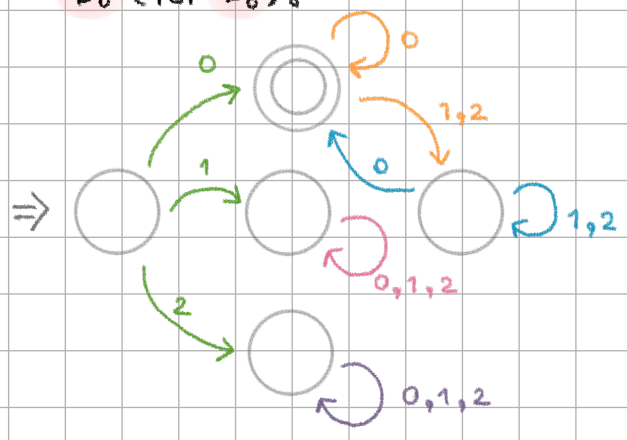


Because a DFA D_{inc} recognizes the language L_{inc}, therefore L_{inc} is regular

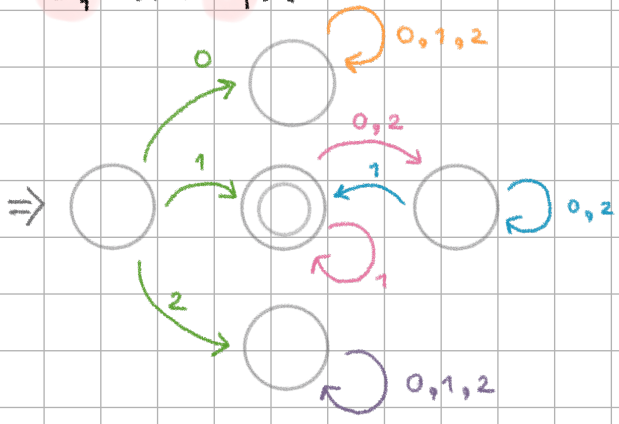
* Note: ε is accepted because it doesn't violate any condition

ii) L_0, L_1, L_2 $\Sigma = \{0, 1, 2\}$

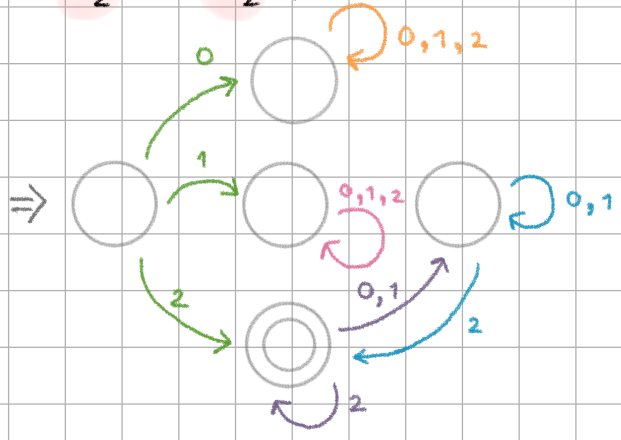
D_0 (for L_0):



D_1 (for L_1):



D_2 (for L_2):



Because DFAs D_0, D_1, D_2 recognize the languages L_0, L_1, L_2, respectively, therefore L_0, L_1, L_2 are regular

* Note: ε is not accepted because it violates the condition that both begin and end digits must be the same

Question 5

b) **Goal:** If L_{prc} and L_0, L_1, L_2 are regular, then L is regular

Proof: Suppose that L_{prc} and L_0, L_1, L_2 are regular

① L_0, L_1, L_2 regular $\Rightarrow$ $L_0 \cup L_1 \cup L_2$ is regular
(regular languages are closed under union)

② $L_{prc}, (L_0 \cup L_1 \cup L_2)$ are regular $\Rightarrow$ $L_{prc} \cap (L_0 \cup L_1 \cup L_2)$ is regular
(regular languages are closed under intersection)

Since $L = L_{prc} \cap (L_0 \cup L_1 \cup L_2)$
(given from the question)

and $L_{prc} \cap (L_0 \cup L_1 \cup L_2)$ is regular,

Therefore, L is regular $\square$

Question 6

1. There are some parts that were more straight-forward such as the DFA/NFA construction. However, some other parts such as proofs were rather confusing. I'm also somewhat unclear about how to translate what's given in a problem into states and edges in automata. Regular operations were also still a little confusing.

2. I love the office hours & the companion videos! ❤

3. I think it would be wonderful if the classes are recorded. If the professor could pause a little bit on iPad before scrolling down and start a new topic, that would be great! Sometimes the screen moved a little too fast and I could no longer see what the professor just wrote.