

ASSIGNMENT 8 (DUE ON 3 OCTOBER 2025)

MATH2301, SEMESTER 2, 2025

1. GENERAL REMARKS

- (1) The assignment is due on gradescope.
- (2) Please read the academic integrity policy for assignments. Remember that if you want an extension, you must ask at least 24 hours ahead of the deadline.
- (3) The words “show” and “prove” are synonyms. You may not be used to writing formal mathematical proofs, which is OK. Write a justification in plain language that would convince the reader.
- (4) If you are having trouble with any of the points, come and discuss with me in office hours. It is part of my job to help you understand this stuff, so please use my time!

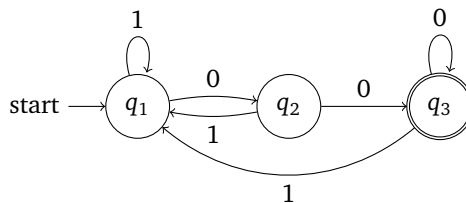
2. PROBLEMS

2.1. **Problem.** Construct an NFA recognising the following languages. Justifications not required.

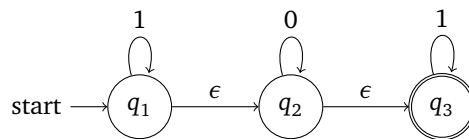
- (1) $L = \{w \mid w \text{ ends with } 00\}$.
- (2) The language $L = L(1^*0^*1^*)$.

2.1.1. *Solution.*

- (1) $L = \{w \mid w \text{ ends with } 00\}$.



- (a) The language $L = L(1^*0^*1^*)$.

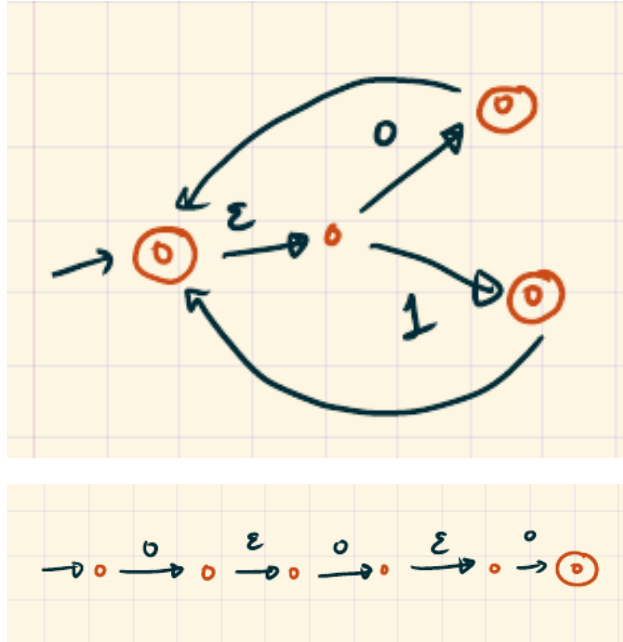


2.2. **Problem.** Convert the following regular expressions to equivalent NFAs. (In each case, break down the given regex into manageable pieces such that you can directly construct a DFA/NFA for each “piece”. Then combine the pieces using the procedures we discussed in class.)

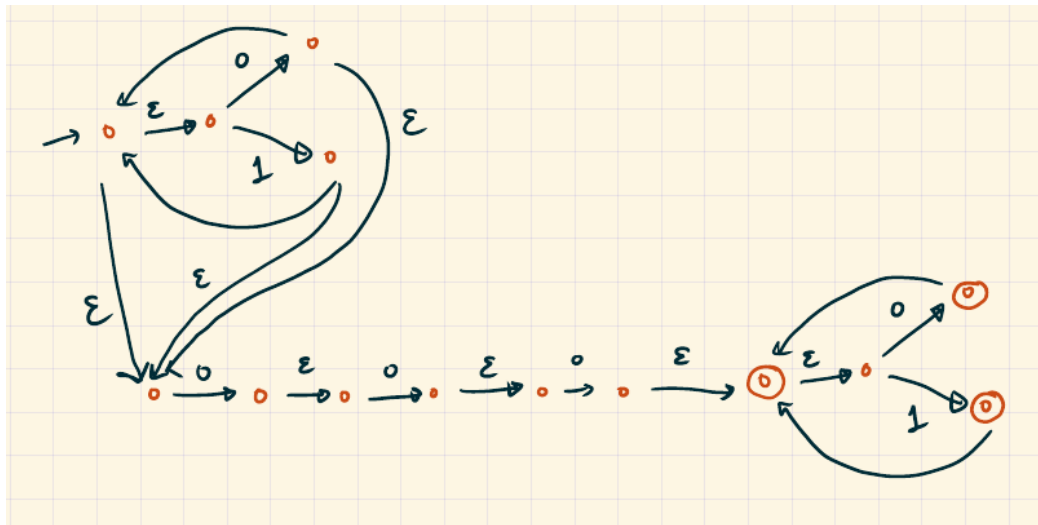
- (1) $r = (0|1)^*000(0|1)^*$
- (2) $r = (((00)^*(11))|01)^*$

2.2.1. *Solution.*

- (1) $r = (0|1)^*000(0|1)^*$ First construct two NFAs: one that recognises $(0|1)^*$ and another that recognises 000.

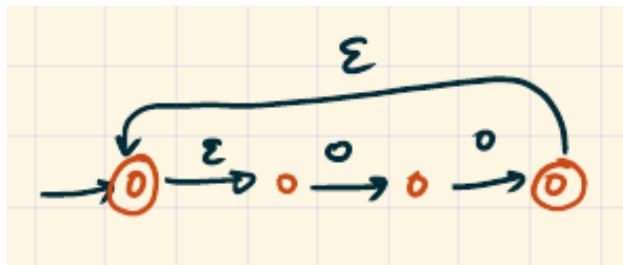


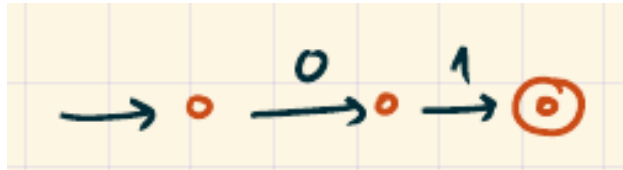
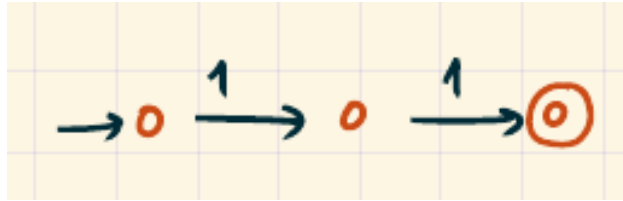
Next put them together as shown in the figure below.



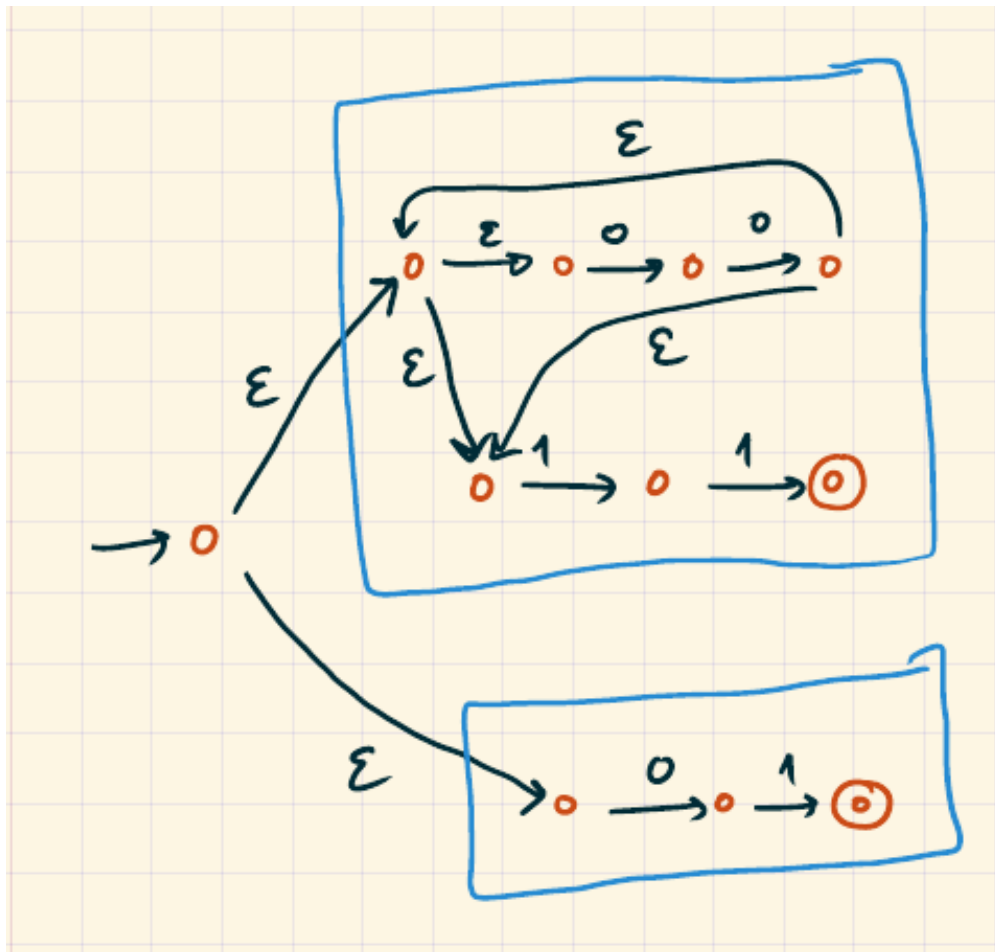
(2) $r = (((00)^*(11))|01)^*$ First construct NFAS corresponding to each individual piece:

- (a) $(00)^*$,
- (b) 11 ,
- (c) 01 .

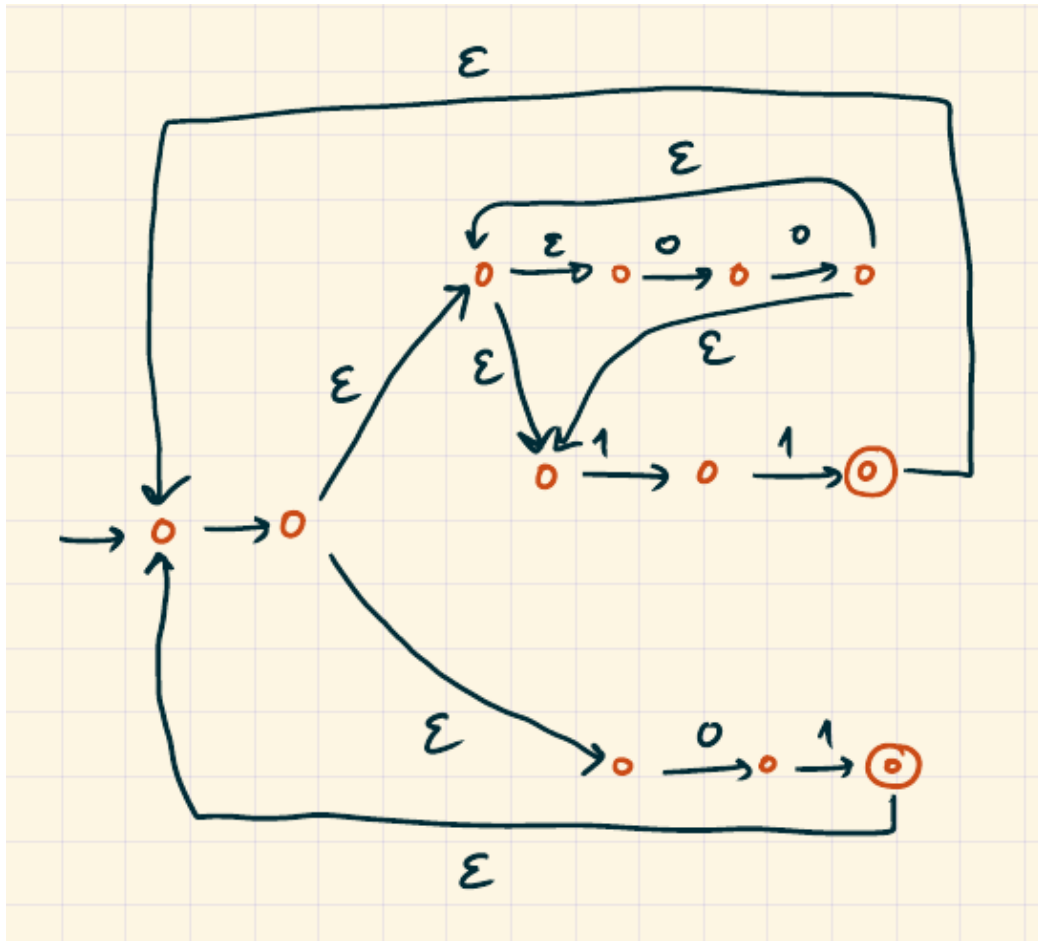




Then we concatenate with 11 to get an NFA for $(00)^*11$, and we use the or operation to combine with 01. The blue boxes show the breakdown into the individual pieces, so to speak.

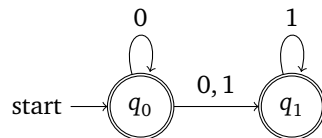


Finally, use star on everything.



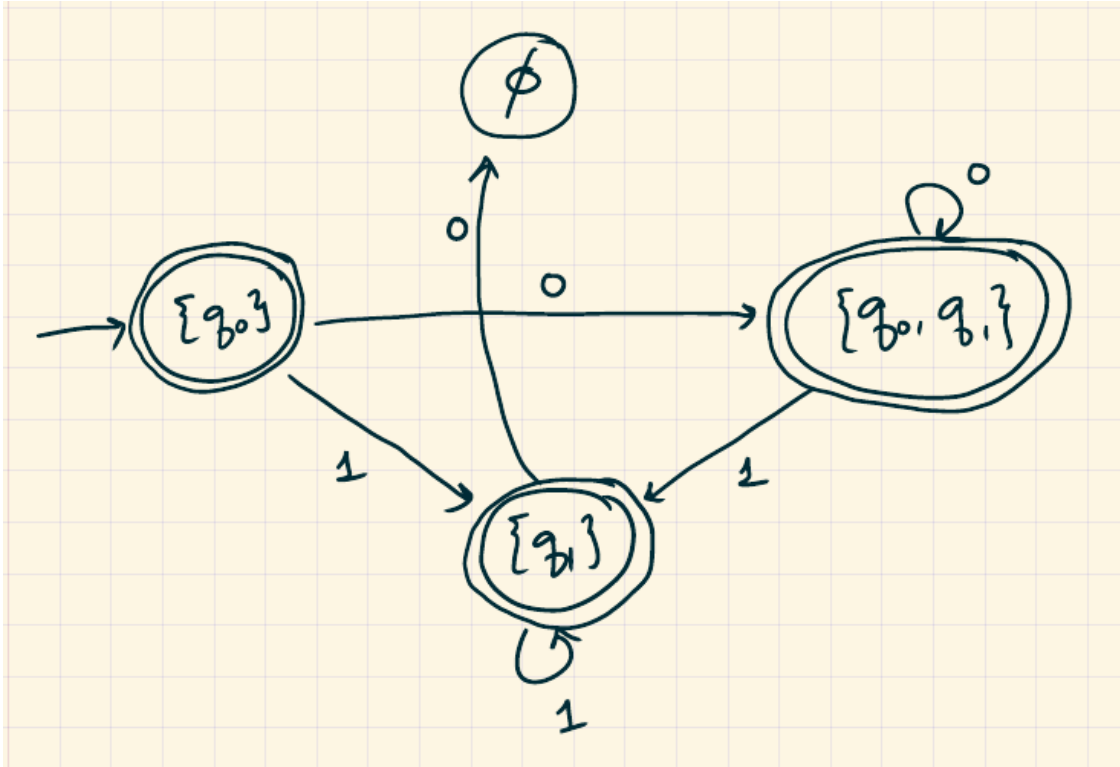
2.3. Problem.

(1) Convert the following NFA into an equivalent DFA.



(2) Let L be the language of the DFA above. How many equivalence classes does \sim_L have? Justify your answer.

2.3.1. Solution. (Edit: the following picture is missing a self-loop from the \emptyset state to itself, labelled by both 0 and 1.)



Since there are 4 states in our DFA, there are at most 4 equivalence classes. Note that the strings ϵ , 0, 1, and 10 end at 4 distinct states. So any other string is equivalent to one of these 4.

Note that $\epsilon \cdot 0$ and $0 \cdot 0$ end at the same state and $\epsilon \cdot 1$ and $0 \cdot 1$ end at the same state. So for any non-empty string z , the strings $\epsilon \cdot z$ and $0 \cdot z$ will end at the same state. In particular, they are both accepted or both rejected. Even for $z = \epsilon$, both ϵ and 0 are accepted. So ϵ and 0 are equivalent.

Note that

- $z = 0$ distinguishes ϵ and 1,
- $z = \epsilon$ distinguishes ϵ and 10,
- $z = \epsilon$ distinguishes 1 and 10.

So there are 3 equivalence classes, represented by ϵ , 1, and 10.

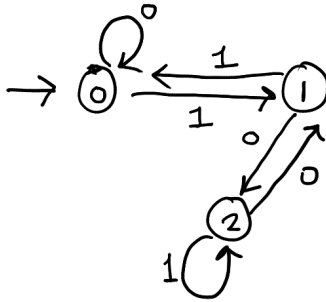
2.4. Problem. Let L be the language

$$L = \{w \mid \text{Read in binary, the number } w \text{ is divisible by } 3\}.$$

Is L recognised by an automaton? If yes, draw an DFA/NFA for L . Otherwise, justify why an automaton does not exist.

We make the convention that leading 0s do not affect the number. So the number 00011 is the same as the number 11, which is the number three.

2.4.1. Solution. Suppose we have read the first k letters and the resulting number is n . If the $k + 1$ -th letter is 0, then the resulting number is $2n$. If the $k + 1$ -th letter is 1, then the resulting number is $2n + 1$. We keep track of these numbers modulo 3 to build our automaton:



The states represent “the number read so far modulo 3”.