

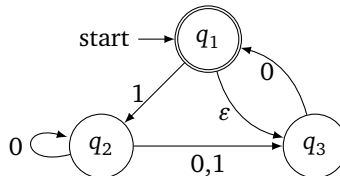
**WEEK 8 WORKSHOP**  
**MATH2301, SEMESTER 2, 2025**

1. WARM-UP: NFA PROBLEMS LEVEL 0

- 1.1. **Problem.** When is an NFA said to *accept* a string? When is an NFA said to *reject* a string?
- 1.2. **Problem.** Let  $M$  be an NFA. Let  $L = L(M)$  and  $L^c = \Sigma^* - L$ . If you take an NFA and switch all the accepting states to non-accepting states and vice-versa, will the language of the resulting machine be  $L^c$ ? Why or why not?
- 1.3. **Problem.** What happens in the calculation of an NFA if from a given state, it is not possible to read the letter we are supposed to read?

2. NFA PROBLEMS LEVEL 1

- 2.1. **Problem.** Consider the NFA shown below.



Find at least one string (ideally more!) that the NFA accepts and at least one that the NFA rejects.

- 2.2. **Problem.** Convert the NFA into an equivalent DFA.

3. NFA PROBLEMS LEVEL 2

The next problems are about constructing machines that do certain things. Carefully convince each other why your solutions work. Some of these problems are hard, and take practice to solve. Don't get discouraged if you can't come up with solutions right away!

- 3.1. **Problem.** Let  $L$  be a language. Consider the following language.

$$L^{del} = \{xz \mid xyz \in L \text{ where } x, z \in \Sigma^*, y \in \Sigma\}.$$

Given an NFA that recognises  $L$ , construct an NFA that recognises  $L^{del}$ .

*Hint: In your NFA, at any point once you've read some portion of the string, you should create the option to ignore one letter and then move on with the calculation.*

- 3.2. **Problem.** Let  $L_1$  and  $L_2$  be languages. Let the *perfect shuffle* of  $L_1$  and  $L_2$  be the language

$$L = \{w \mid w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1, \dots, a_k, b_1, \dots, b_k \in \Sigma \text{ and } a_1 \cdots a_k \in L_1 \text{ and } b_1 \cdots b_k \in L_2\}.$$

(The number  $k$  can be arbitrary). As a warm up to understand the construction:

- (1) Take  $L_1 = 0^*$  and  $L_2 = 1^*$ , and describe the perfect shuffle  $L$ .
- (2) Take  $L_1 = \{0, 00, 000\}$  and  $L_2 = \{1, 11, 111\}$ , describe the perfect shuffle  $L$ .

Now, given automata  $M_1$  and  $M_2$  recognising  $L_1$  and  $L_2$ , respectively, construct an automaton  $M$  to recognise  $L$ . You may assume that  $M_1$  and  $M_2$  are deterministic, and construct a non-deterministic  $M$ .

Here is a series of pointers to help you come to a complete solution of this problem.

- (1) We'll have to somehow combine the DFAs recognising the two languages into a third one.
- (2) We'll need to start at the start state of  $M_1$ , because we expect the first letter of any valid word to be a letter that's valid at the start of any word in  $L_1$ .

- (3) Once we read a letter from  $L_1$ , we have to “pause” and read a letter from  $L_2$  in order to be valid. But after that, we’ll have to “resume” in  $M_1$ , which means we have to remember where we came from. Can you simulate this using a product-type construction?

3.3. **Problem.** (\*, bonus) If you’re finished with the remainder of the worksheet, construct an automaton to recognise the *shuffle* of two regular languages  $L_1$  and  $L_2$ , defined as follows:

$$L = \{w \mid w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1, \dots, a_k, b_1, \dots, b_k \in \Sigma^* \text{ are such that } a_1 \cdots a_k \in L_1 \text{ and } b_1 \cdots b_k \in L_2\}.$$