After this tutorial you should be able to:

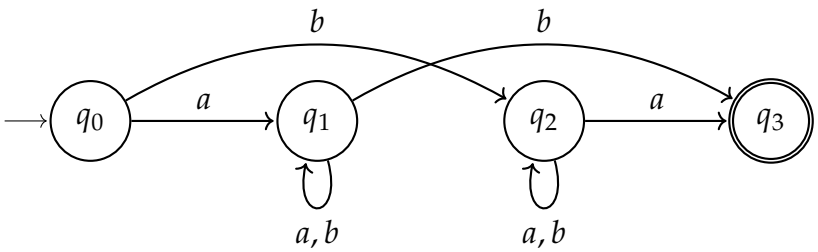
1. Remove ϵ-transitions from NFAs
2. Convert NFAs to DFAs
3. Convert DFAs to REs
4. Devise algorithms for solving certain decision problems about DFAs/NFAs

---

Problem 1. Write pseudo-code for EC, i.e., a procedure that takes an NFA N and a pair (q, r) of states as input, and returns whether or not there is a path from q to r labeled by zero or more ϵ-transitions.

Problem 2.[Exam 2020] Using the methods from the course, convert the regular expression $a^* \mid b$ into an NFA, and then remove its epsilon transitions. Show all the steps.

Problem 3.[Exam 2022] Describe in words the language recognised by the following NFA (no additional justification is needed), and then give the DFA that results from applying the subset construction to the NFA:



You may draw or write the DFA, and no additional justification is needed. Note that you should only include the states in the DFA that are reachable from the initial state.

Problem 4.[Exam 2021] Consider the DFA M with states $Q = \{0, 1, 2\}$, $\Sigma = \{a, b\}$, $q_0 = 0$, $F = \{1\}$, and δ is as follows:

	a	b
0	0	1
1	2	0
2	2	2

Give a short English description of $L(M)$ and a regular expression for $L(M)$. No additional justifications are needed. (In the exam the question did not specify how this should be done; in the tutorial, you should use the methods from lectures).

Algorithms

Problem 5. Consider the decision problem which takes as input a regular expression R and string w, and outputs 1 if $w \in L(R)$, and 0 otherwise. This problem is called the *regular-expression membership problem*. In a previous tutorial we saw a recursive algorithm for solving this problem (called Match(R, w)). We gave the following algorithm in class:

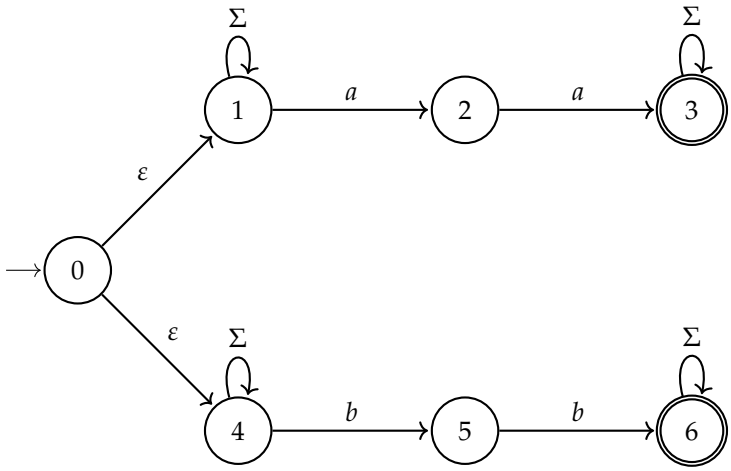
1. Convert R into a NFA N such that $L(R) = L(N)$.
2. Check if $w \in L(N)$.

We say how to do step 1 in class. Write pseudocode for step 2.

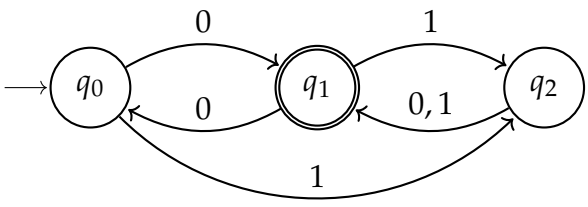
Problem 6. Consider the decision problem which takes as input regular expressions R_1, R_2, and outputs 1 if $L(R_1) = L(R_2)$, and 0 otherwise. This problem is called the *regular-expression equivalence problem*. Give an algorithm that solves this decision problem. (Hint: use automata).

Extra

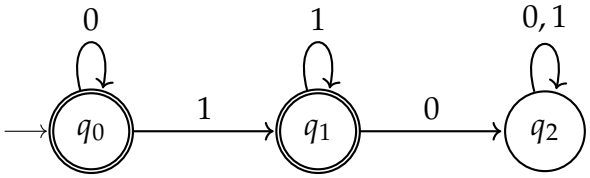
Problem 7. Here is an NFA over $\Sigma = \{a, b\}$. What language does it describe? Transform the NFA into an equivalent DFA, using the method shown in lectures.



Problem 8. Convert the DFA to a regular expression.



Problem 9. Convert the DFA to a regular expression.



Problem 10. Construct a DFA which accepts strings of 0's and 1's, where the number of 1's modulo 3 equals the number of 0's modulo 3.

Give the corresponding regular expression

Problem 11. Construct a DFA which accepts strings of 0's and 1's, where the number of 1's modulo 3 does not equal the number of 0's modulo 3.

Give the corresponding regular expression.

Problem 12.

1. Construct a DFA which accepts binary numbers that are multiples of 3, scanning the most significant bit (i.e. leftmost) first. Hint: consider the value of the remainder so far, as we scan across the number.
2. Construct an automaton which accepts binary numbers which are NOT multiples of 3.

Problem 13.

1. Construct a DFA which recognises binary numbers which are multiples of 2.
2. Construct a DFA which recognises binary numbers which are multiples of 3 (previous exercise)
3. Using these two DFA, construct an NFA which accepts binary numbers which are either not a multiple of 2, or not a multiple of 3, or both. (i.e. 2 is accepted, 3 is accepted, but 6 is not accepted)
4. Convert it to a DFA
5. Swap the accept and non-accept states. What is the language accepted by this new DFA?