1 Regular Expressions ($\star$)

Give regular expressions for the following languages over the alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

1. $\{w \mid w$ begins with a `1` and ends with a `0`$\}$
2. $\{w \mid w$ contains at least 3 `1`s$\}$
3. $\{w \mid w$ contains the substring `0101`$\}$
4. $\{w \mid w$ has length at least 3 and the third symbol is a `0`$\}$
5. $\{w \mid w$ starts with `0` and has odd length or starts with `1` and has even length $\}$
6. $\{w \mid$ the length of w is at most 5 $\}$
7. $\{w \mid$ contains at least two `0`s and at most one `1` $\}$
8. $\{\varepsilon, \mathtt{0}\}$
9. $\{w \mid$ contains an even number of `0`s or exactly two `1`s $\}$
10. $\emptyset$
11. All strings except the empty string.
12. ($\star\star$) $\{w \mid w$ does not contain the substring `110`$\}$
13. ($\star\star$) $\{w \mid w$ is any string except `11` and `111`$\}$
14. ($\star\star$) $\{w \mid$ every odd position of w is a `1`$\}$

2 Careful with Composition ($\star\star$)

The following construction is supposed to prove that the class of regular languages is closed under the Kleene star operation. However, it is incorrect — give a counterexample of an automaton N for which this construction does not recognise $L(N)^*$.

Let L be a regular language and suppose that $N = (Q, \Sigma, \delta, q_0, F)$ is a NFA that recognises L. Construct a NFA N' as follows:

$$Q' := Q \qquad \Sigma' := \Sigma \qquad q_0' := q_0 \qquad F' := F \cup \{q_0\}$$

$$\delta'(q, a) := \begin{cases} \delta(q, a) & q \notin F \text{ or } a \neq \varepsilon \\ \delta(q, a) \cup \{q_0\} & q \in F \text{ and } a = \varepsilon \end{cases}$$

Hint: try and draw this construction graphically.

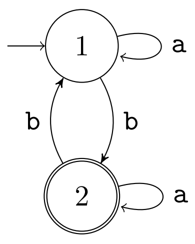
3 Regular Expressions to NFAs ($\star$)

Convert the following regular expressions to NFAs. The alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

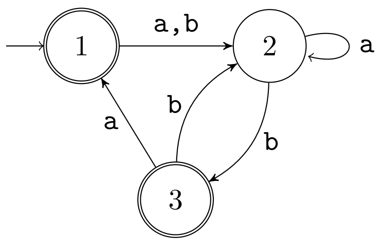
1. $\mathtt{a(abb)}^* \cup \mathtt{b}$
2. $\mathtt{a}^+ \cup \mathtt{(ab)}^+$
3. $\mathtt{(a} \cup \mathtt{b}^+\mathtt{)a}^+\mathtt{b}^+$

4 DFAs to Regular Expressions ($\star$)

Convert the following DFAs to regular expressions.



(1)



(2)

Optional exercises

5 Regular Shuffle ($\star\star\star$)

Let A and B be languages. The *perfect shuffle* of A and B is the following language:

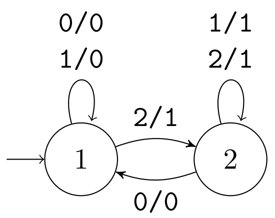
$$\{w \mid w = a_1b_1 \dots a_kb_k \text{ where } a_1 \dots a_k \in A \text{ and } b_1 \dots b_k \in B, \text{ each } a_i, b_i \in \Sigma\}$$

For example, if `abcd` is a word in A and `pqrs` is in B then the word `apbqcrds` is in the perfect shuffle of A and B.

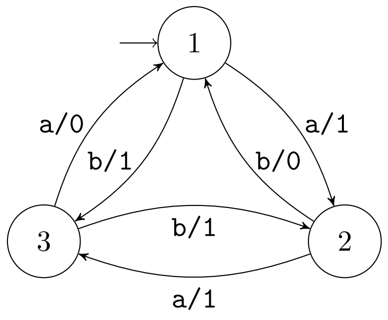
Show that the class of regular languages is closed under the perfect shuffle.

6 Informal FSTs ($\star$)

A Finite State Transducer (FST) is a variation on a DFA that returns a string rather than just accepting or rejecting. Here are two examples:



(1)



(2)

Each transition of an FST is lebeled with two symbols, one designating the input symbol for the transition and the other designating the output symbol. The two symbols are separated by a slash. When an FST computes on an input string w, it takes the input symbols $w_1 \dots w_n$ one by one and, starting at the start state, follows the transitions by matching the input labels with the sequence of symbols $w_1 \dots w_n$. Every time it goes along a transition it outputs the corresponding output symbol.

For example, on input `2212011` the first machine follows the states $1, 2, 2, 2, 2, 1, 1, 1$ and produces output `1111000`.

Give the sequence of states entered and the output produced in each of the following:

1. (1) on input `011`
2. (1) on input `211`
3. (1) on input `121`
4. (1) on input `0202`
5. (2) on input `b`
6. (2) on input `bbab`
7. (2) on input `bbbbbb`
8. (2) on input ε

7 Formal FSTs ($\star\star$)

Given the informal description of FSTs in the last exercise, give a formal definition like for DFAs as a 5-tuple. A FST has two alphabets, an input alphabet Σ and an output alphabet Γ and does not have accept states. Further give a formal definition of the computation of an FST.

Hint: The transition function goes from $Q \times \Sigma$ to $Q \times \Gamma$.