Regular Languages and Automata: Problems for Week 1

Note: when we ask for a DFA, we are happy for you to supply a partial DFA. Indeed that's usually better, because it's more efficient.

Exercise 1 *Give a regexp over the alphabet $\Sigma = \{a, b, c\}$ for the set of words in which "a" occurs precisely twice.*

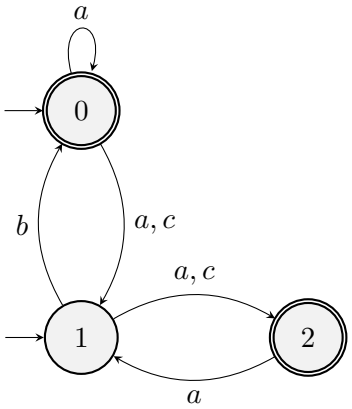
Exercise 2 *Build a DFA that checks whether a string is equal to "`Goo....gle`" with arbitrarily many `o`'s following the initial two.*

Exercise 3 *Design DFAs for the following regular expressions:*

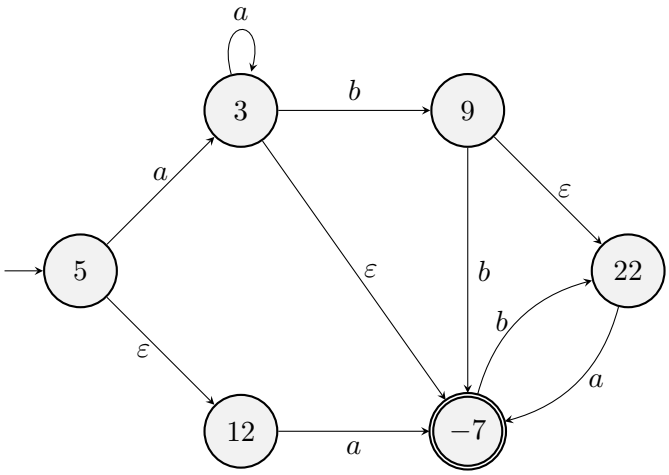
1. `(a|b)c`
2. `ab|bc`
3. `ab|ac` *(Careful! Remember that from any state there must be at most* one *transition labelled with a particular letter.)*
4. `c(a|b)*c`

Exercise 4 *An online shop requires users to provide a password during registration. Every password is a string of lowercase letters and digits. It must contain at least one letter and at least one digit, and it must be at least three characters long. Give a regular expression for passwords. You can use* `[a-z]`*, which matches any lowercase letter, and* `[0-9]`*, which matches any digit.*

Exercise 5 *Determinize the following NFA.*



Exercise 6 *Remove ε-transitions from the following.*



Exercise 7 *Construct a partial DFA that accepts the language described by the regexp*

$$(ab)^*c(a|b)$$

in three steps:

1. *Construct a suitable εNFA, using Kleene's Theorem.*
2. *Transform the εNFA into an NFA.*
3. *Determinize the NFA to obtain a partial DFA.*

Describe each step in detail.

Bonus: turn the partial DFA into a total one, assuming that the alphabet is $\Sigma = \{a, b, c\}$.