Regular Languages and Automata: Problems for Week 1

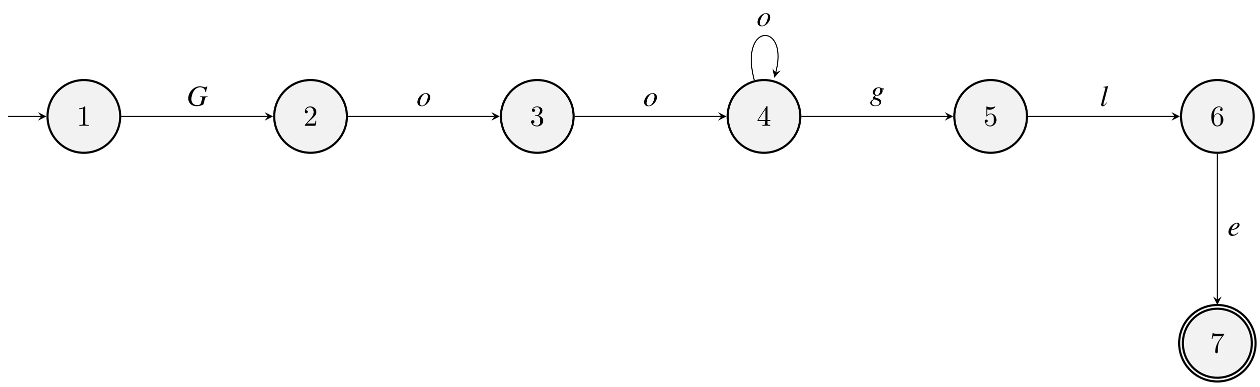
Note: when we ask for a DFA, we are happy for you to supply a partial DFA. Indeed that's usually better, because it's more efficient.

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

Solution 1. $(b|c)^*a(b|c)^*a(b|c)^*$

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

Solution 2.

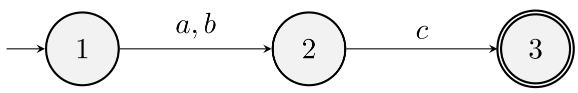


Exercise 3. *Design DFAs for the following regular expressions:*

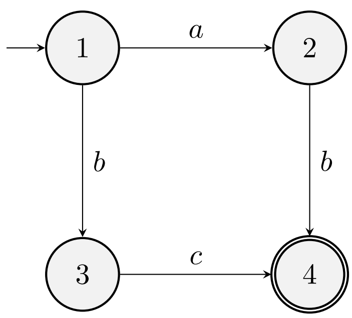
1. `(a|b)c`
2. `ab|bc`
3. `ab|ac` *(Careful! Remember that from any state there must be at most* one *transition labelled with a particular letter.)*
4. `c(a|b)*c`

Solution 3.

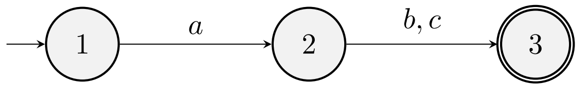
1.



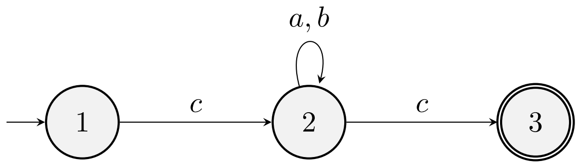
2.



3.



4.



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.*

Solution 4. *One way of solving this problem is to divide it into cases. Clearly there are two cases, passwords beginning with a letter and passwords beginning with a digit. A password beginning with a letter consists of four parts:*

- *a letter*
- *then a (possibly empty) string of letters*
- *then a digit—i.e. the first digit appearing in the password*
- *then a (possibly empty) string of letters and digits.*

The problem is that the second and fourth part can't both be empty because then the password would be only two characters long. So there are two acceptable sub-cases:

- *passwords in which the second part is empty and the fourth part is not:*

```
[a-z][0-9]([a-z]|[0-9])([a-z]|[0-9])*
```

- *passwords in which the second part is not empty and the fourth part has any length:*

```
[a-z][a-z][a-z]*[0-9]([a-z]|[0-9])*
```

Putting together these two sub-cases we obtain an expression for passwords beginning with a letter:

```
[a-z][0-9]([a-z]|[0-9])([a-z]|[0-9])*  |
[a-z][a-z][a-z]*[0-9]([a-z]|[0-9])*
```

In the same way, we obtain an expression for passwords beginning with a digit:

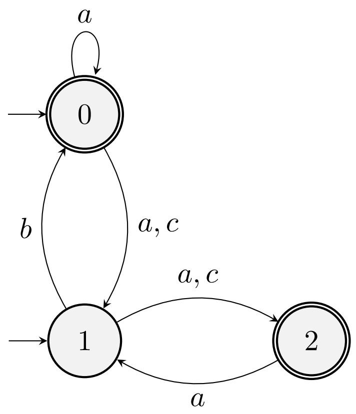
```
[0-9][a-z]([a-z]|[0-9])([a-z]|[0-9])*  |
[0-9][0-9][0-9]*[a-z]([a-z]|[0-9])*
```

Putting together the two kinds of passwords, we obtain:

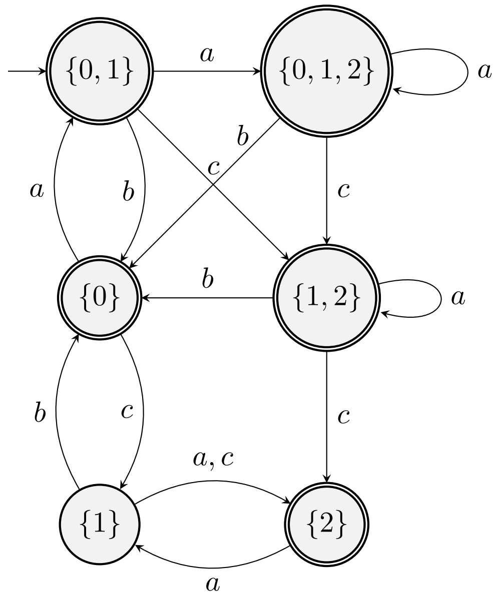
```
[a-z][0-9]([a-z]|[0-9])([a-z]|[0-9])*  |
[a-z][a-z][a-z]*[0-9]([a-z]|[0-9])*    |
[0-9][a-z]([a-z]|[0-9])([a-z]|[0-9])*  |
[0-9][0-9][0-9]*[a-z]([a-z]|[0-9])*
```

Of course this is just one solution to the question; there are many others.

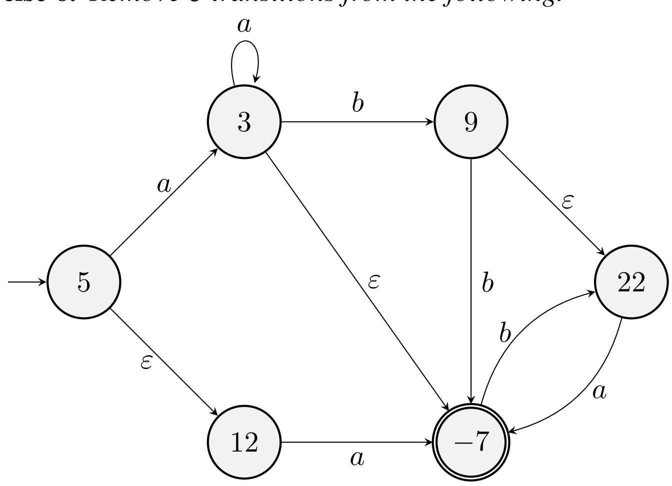
Exercise 5. *Determinize the following NFA.*

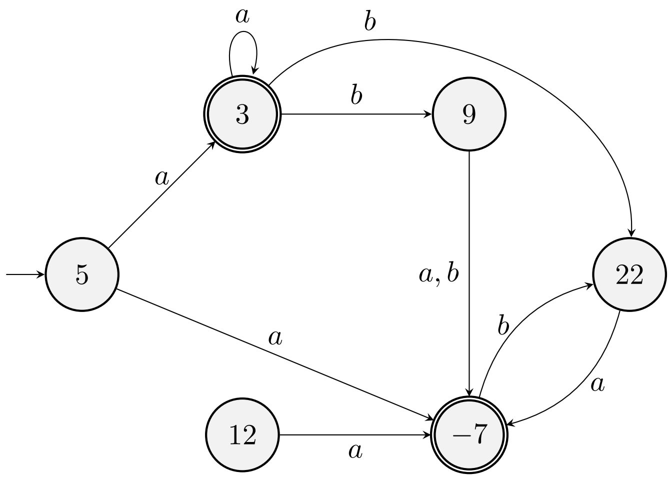


Solution 5.



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





Solution 6.

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.

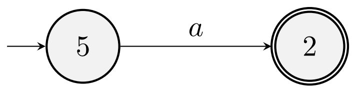
Bonus: turn the partial DFA into a total one, assuming that the alphabet is $\Sigma = \{a, b, c\}$.

Solution 7.

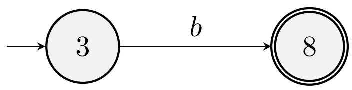
1. *We create the automaton piecewise:*

   (a) *An εNFA for $(ab)^*$ is obtained as follows:*

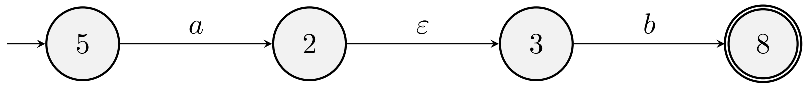
   For just "a":

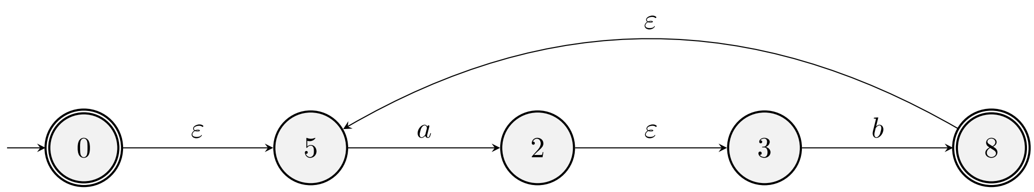
   For just "b":

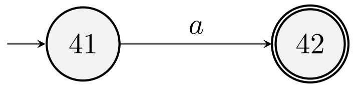
   For "ab" we thus obtain:

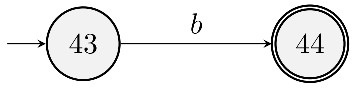
   

   An automaton recognizing $(ab)^*$ thus is

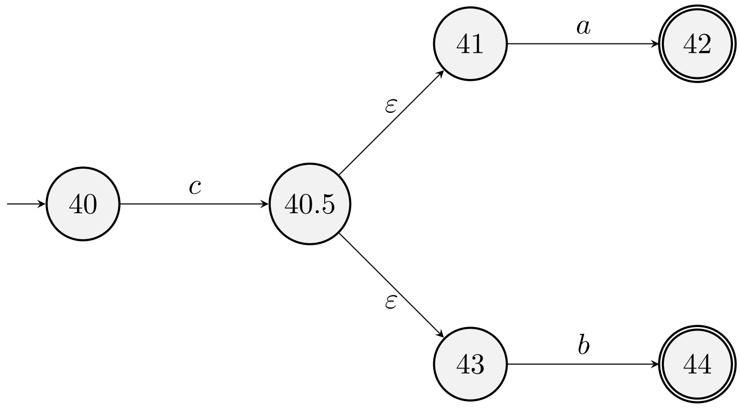
   

   (b) *An automaton for "$a|b$" is given by*





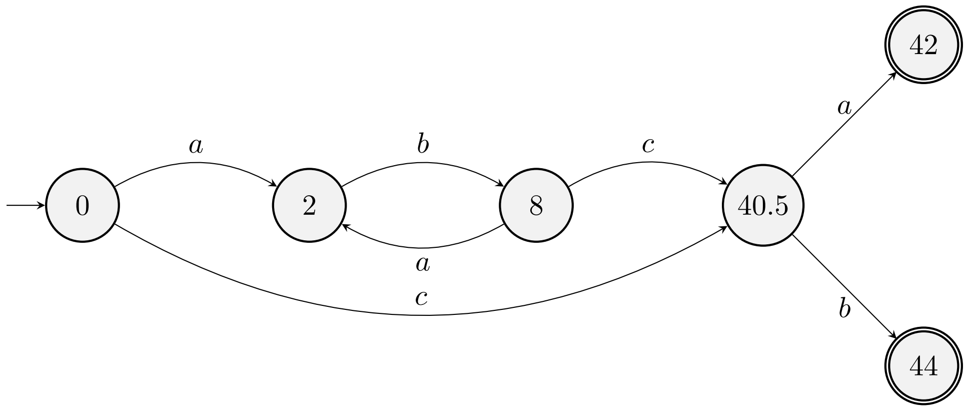
Consequently, an automaton accepting exactly $c(a|b)$ *is*



(c) *We thus obtain an automaton accepting exactly* $(ab)^*c(a|b)$ *as indicated on page 6*

2. *We replace the transitions by* slow *transitions, obtaining the automaton on page 7.*

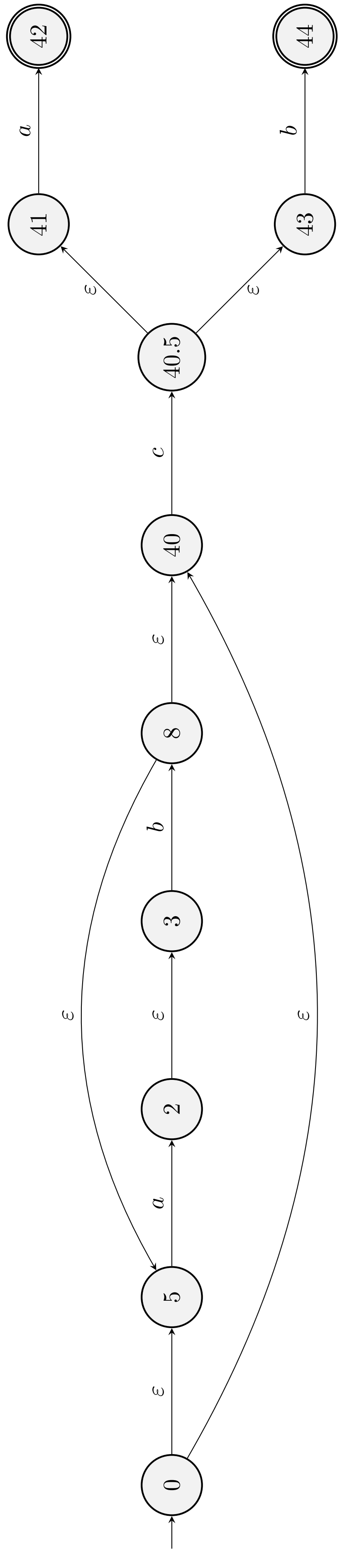
   After "cleaning up" by removing the unreachable states, we obtain the automaton

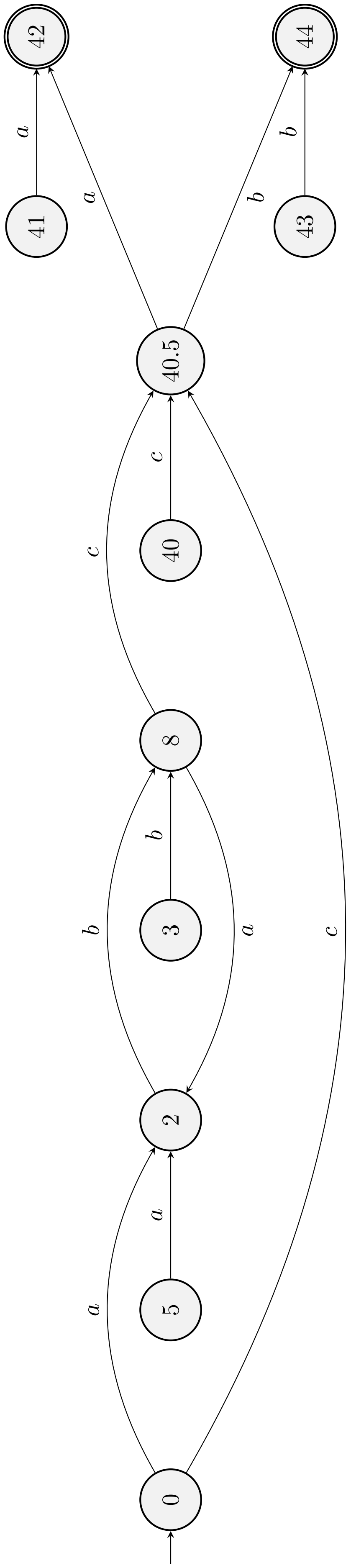


3. *The automaton is already deterministic, since*

   (a) *there is exactly one initial state, and*

   (b) *there is at most one* X*-transition out of any state, for any* $X \in \{a, b, c\}$


0
ε
5
a
2
ε
3
b
8
ε
40
c
40.5
ε
41
a
42
ε
43
b
44
ε
ε


0
5
2
3
8
40
40.5
41
42
43
44
a
b
c