Assignment 3 by Lucas Karlsson

1. Construct an NFA (without ϵ-transitions) that recognises the same language over $\{0, 1\}$ as the rgular expression $(0 + 01^*)^*(\epsilon + 1)1(\epsilon + 0 + 1)^*$.

Solution 1:
To construct this NFA I'd first like to rewrite or simplify the regular expression, there is no easy way of doing this have I come to find the hard way. But I'm going to give it a shot!

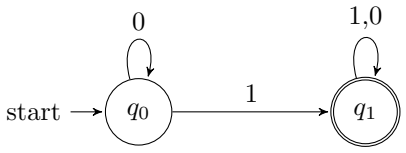
$(0+01^*)^* = (01^*)^*$ Because it has to start with either nothing or a zero followed by zero or more ones and doing all of this zero or more times.

We now have $(01^*)^*(\epsilon + 1)1(\epsilon + 0 + 1)^*$ and $(\epsilon + 1)1 = (1 + 11)$ because of distributative laws, basically we either have one 1 or two 1's.

We can also simplify $(\epsilon + 0 + 1)^*$ which is the same thing as $(0 + 1)^*$ because $0\epsilon1$ is the same thing as 01

Our regular expression will now be $(01^*)^*(1+11)(0+1)^*$ we can immediatelly see that $(1 + 11)$ is can be simplified further because our 11 case is already covered by the $(0+1)^*$ so we can simply rewrite the expression again to $(01^*)^*1(0+1)^*$. In words we can explain this as the paragraph below and we can easily create a simple NFA as seen below.

"Starts with either zero or more 0's followed by atleast one 1 then any of 0 or 1 zero or more times"

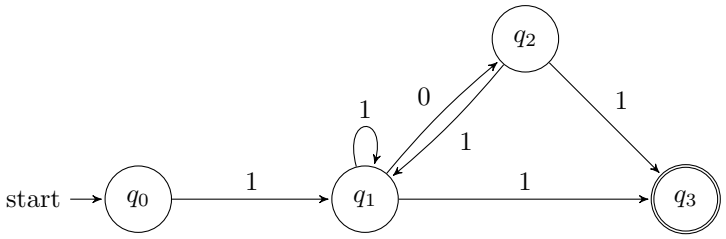


This NFA satisfies the regular expression in question and recongnises the same language over $\{0, 1\}$ You can also find this NFA in the file **ass3point1.jff**.

2. Construct a regular expression for the language over $\{0, 1\}$ that consists of every string $w \in \{0, 1\}^*$ that satisfies conditions in the question.

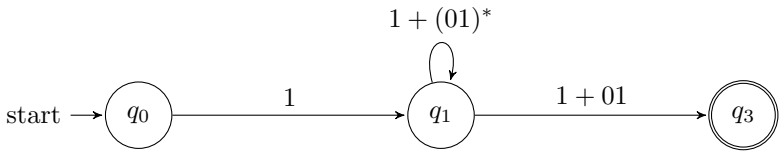
Solution 2:
We need to satisfy the properities $|w| \geqslant 2$, the first and last symbol in w must be 1 and the string 00 must not occur in w.

What I did is first create a NFA satisfying the properties, which looks as below:

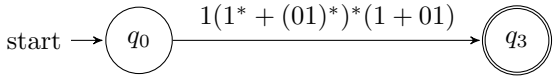


Now we can start converting this NFA to a regular expression this can be done using a method that eliminates states replacing them with regular expressions.

Step 1, we can now prepare to remove states. This is easy because we have no states coming in to our initial states and no states coming out of our final state, that basically translates into we don't need to do something before we start removing states. We start by removing the q_2 state. Our automata will look like this:



The only thing we now have left to remove is the q_1 state, this would look as following:



We are done here but we can simplify the regular expression further because $(1^* + (01)^*)^* = (1 + 01)^*$ and our final regular expression will then be:

$$1(1 + 01)^*(1 + 01)$$

3. Convert the ϵ-NFA in the question to an equivalent DFA.

	State	ϵ	**a**	**b**
$\rightarrow$	s_0	$\emptyset$	$\{s_1\}$	$\{s_0, s_2\}$
	s_1	$\{s_2\}$	$\{s_4\}$	$\{s_3\}$
	s_2	$\emptyset$	$\{s_1, s_4\}$	$\{s_3\}$
	s_3	$\{s_5\}$	$\{s_4, s_5\}$	$\emptyset$
	s_4	$\{s_3\}$	$\emptyset$	$\{s_5\}$
*	s_5	$\emptyset$	$\{s_5\}$	$\{s_5\}$

Solution 3:
I'm going to start by converting the ϵ-NFA to a equivalent NFA this can be done by everytime we find a ϵ for every occurence of the state we add the states in the ϵ transition. You can also express this with $\delta(q_0, a) = \delta_{hat}(q_0, a)$. Our new table will look this:

	State	**a**	**b**
$\rightarrow$	s_0	s_1, s_2	s_0, s_2
	s_1	s_3, s_4, s_5	s_3, s_5
	s_2	s_1, s_2, s_3, s_4, s_5	s_3, s_5
	s_3	s_3, s_4, s_5	$\emptyset$
	s_4	$\emptyset$	s_5
*	s_5	s_5	s_5

Now we can create a DFA using this NFA by starting at s_0 and doing $\delta(s_0, a)$ and $\delta(s_0, b)$ and using the states that we get from the calculation doing the same δ function for a and b for them. I will do the calculations here in latex and filling in my table as I go along.

$\delta(s_0, a) = \delta_n(s_0, a) = \{s_1, s_2\}$
$\delta(s_0, b) = \delta_n(s_0, b) = \{s_0, s_2\}$

$\delta(s_1 s_2, a) = \delta_n(s_1, a) \cup \delta_n(s_2, a) = \{s_1, s_2, s_3, s_4, s_5\}$
$\delta(s_1 s_2, b) = \delta_n(s_1, b) \cup \delta_n(s_2, b) = \{s_3, s_5\}$
$\delta(s_0 s_2, a) = \delta_n(s_0, a) \cup \delta_n(s_2, a) = \{s_1, s_2, s_3, s_4, s_5\}$
$\delta(s_0 s_2, b) = \delta_n(s_0, b) \cup \delta_n(s_2, b) = \{s_0, s_2, s_3, s_5\}$

$\delta(s_1 s_2 s_3 s_4 s_5, a) = \delta_n(s_1, a) \cup \delta_n(s_2, a) \cup \delta_n(s_3, a) \cup \delta_n(s_4, a) \cup \delta_n(s_5, a) = \{s_1, s_2, s_3, s_4, s_5\}$
$\delta(s_1 s_2 s_3 s_4 s_5, b) = \delta_n(s_1, b) \cup \delta_n(s_2, b) \cup \delta_n(s_3, b) \cup \delta_n(s_4, b) \cup \delta_n(s_5, b) = \{s_3, s_5\}$
$\delta(s_3 s_5, a) = \delta_n(s_3, a) \cup \delta_n(s_5, a) = \{s_3, s_4, s_5\}$
$\delta(s_3 s_5, b) = \delta_n(s_3, b) \cup \delta_n(s_5, b) = \{s_5\}$
$\delta(s_0 s_2 s_3 s_5, a) = \delta_n(s_0, a) \cup \delta_n(s_2, a) \cup \delta_n(s_3, a) \cup \delta_n(s_5, a) = \{s_1, s_2, s_3, s_4, s_5\}$
$\delta(s_0 s_2 s_3 s_5, b) = \delta_n(s_0, b) \cup \delta_n(s_2, b) \cup \delta_n(s_3, b) \cup \delta_n(s_5, b) = \{s_0, s_2, s_3, s_5\}$

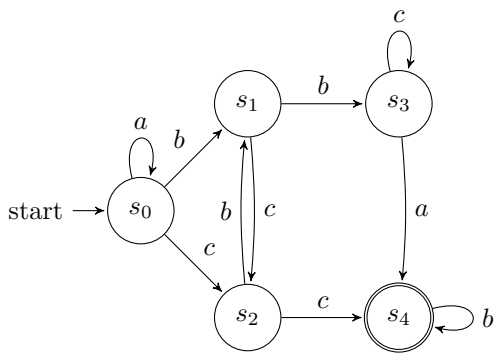
$\delta(s_3 s_4 s_5, a) = \delta_n(s_3, a) \cup \delta_n(s_4, a) \cup \delta_n(s_5, a) = \{s_3, s_4, s_5\}$
$\delta(s_3 s_4 s_5, b) = \delta_n(s_3, b) \cup \delta_n(s_4, b) \cup \delta_n(s_5, b) = \{s_5\}$

$\delta(s_5, a) = \delta_n(s_5, a) = \{s_5\}$
$\delta(s_5, b) = \delta_n(s_5, b) = \{s_5\}$

	State	**a**	**b**
$\rightarrow (q_0)$	s_0	$s_1 s_2$	$s_0 s_2$
(q_1)	$s_1 s_2$	$s_1 s_2 s_3 s_4 s_5$	$s_3 s_5$
(q_2)	$s_0 s_2$	$s_1 s_2 s_3 s_4 s_5$	$s_0 s_2 s_3 s_5$
$* (q_3)$	$s_1 s_2 s_3 s_4 s_5$	$s_1 s_2 s_3 s_4 s_5$	$s_3 s_5$
$* (q_4)$	$s_3 s_5$	$s_3 s_4 s_5$	s_5
$* (q_5)$	$s_0 s_2 s_3 s_5$	$s_1 s_2 s_3 s_4 s_5$	$s_0 s_2 s_3 s_5$
$* (q_6)$	$s_3 s_4 s_5$	$s_3 s_4 s_5$	s_5
$* (q_7)$	s_5	s_5	s_5

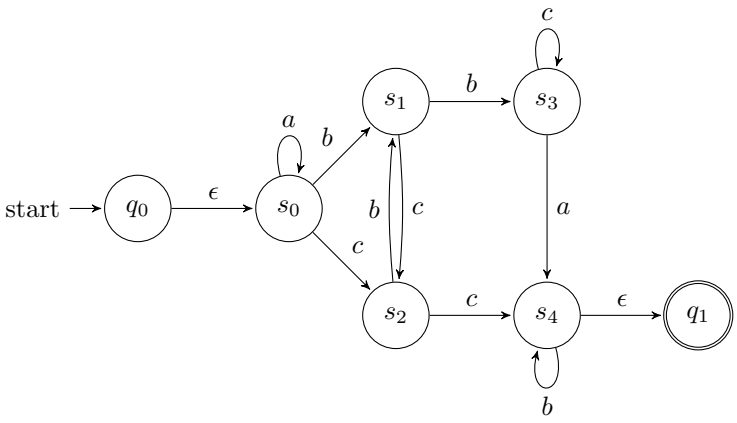
This will be our final construction, we have now created a DFA from the given ϵ-NFA. This DFA can also be found in the file **ass3point3.jff**.

4. Convert the NFA to an equivalent regular expression

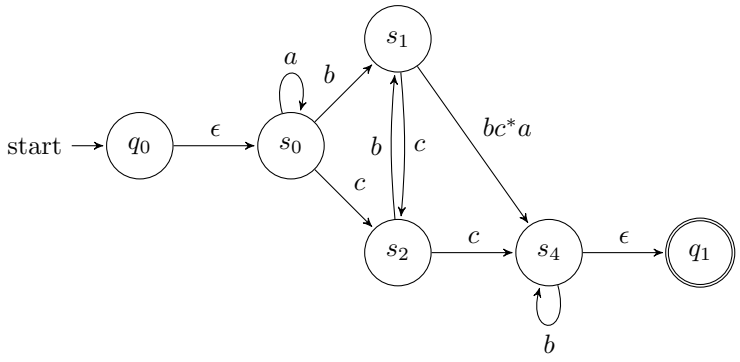


Solution 4:
To start off converting this NFA I'm going to establish the technique I'm going to use. It is indeed quite similar to the one I used in **Solution 2** but we need to do some extra steps. I can immediatelly see that because I have **in-going** edges on my initial state which needs to be fixed. I also have **outgoing** edges on my accepting state, this will also need correcection.

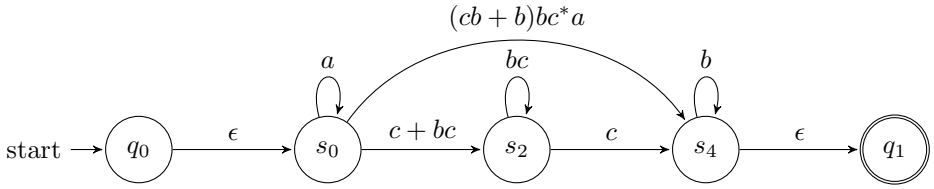
Step 1 Add incoming ϵ-transistion and new accepting state.



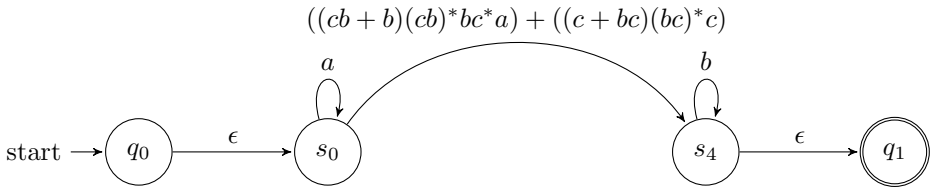
Step 2 We can now start removing states and replacing them with regular expressions. I'd like to remove s_3 first. This can easily be done by adding the bc^*a edge.



Step 3 Next I'd like to remove the s_1 state, this is a bit harder but can be done by adding bc loop on s_2 and changing the s_0s_2 edge to $c + bc$ and adding s_0s_4 edge $(cb + b)bc^*a$.



Step 4 Now we are almost done and can simply remove the s_2 state and join our two regular expression with $(bc)^*$ in the middle.



Step 5 Only thing left now is removing s_0 and s_4, easy peasy lemon squeezy. Our final regular expression and our answer to the question will be the one inbetween q_0 and q_1