Assignment 2
DFA and NFA

Josefin Ulfenborg
940806-5960
yunalescca@gmail.com

2017-04-09

1 Task 1

My first thought was to express the language as something that contains the words on the form "x0y", as seen in the lecture notes. The result I got from this was that the language accepts words on the form "x1y0z", and where $x, y, z \in \Sigma^*$. However, this suggests that y could be any sequence of symbols, when it really can only be 2 or nothing. The same thing goes for z. This means you'd have to restrict y and z to only be a certain set of words, which can get tedious, and increases the risk for error making. Hence, I choose not do describe the language like this.

A more descriptive and better way to describe this is to use more words and context rather than just the above notation. My first note on the words are that the words accepted by the automaton starts with either a 0, a 1 or a 2 (trivial).

A thing that can be said about the form of the words are existing sub words. All words that are accepted by the automaton will contain either "10" or "120" as sub word(s). But one has to be careful here not to mistakenly believe that just because a word contains one of the sub words, it is accepted. This is not the case since the word "110" contains "10" as a sub word but is not accepted by the automaton. Nonetheless, all words that will be accepted will contain either "10" or "120", but not all words that contain those will be accepted (i.e. not equivalent).

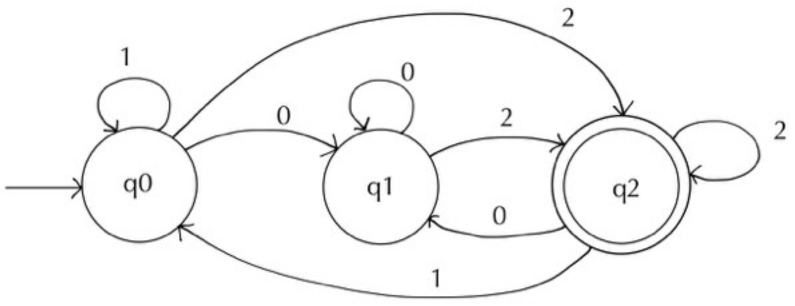
What else can be said about the language is that it accepts words that has at least one 1 **and** one 0, as well as the words should always have an odd number of 1's. However, the same rule for equivalence goes here. Words that just contain a 0 or a 1 will not necessarily be accepted.

Finally, a thing that can be said about the accepted words is that they will either end with a 0 or a 2. And again, the same goes here: just because a word ends with a 0 does not mean it will be accepted by the automaton, but all words that will be accepted will end with a 0.

2 Task 2

This task is about defining a DFA that has three non-dead states that accepts the language over the alphabet 0,1,2 where words end with a 2, and have no 1 following directly after 0. This means that *all* words that follow this form have to be accepted, for instance 2, 02, 12, 22, 10002 and so on.

The result can be seen in the image below, where q_0 has been marked as the starting state ($\rightarrow$) and q_2 as the final state ($\star$).

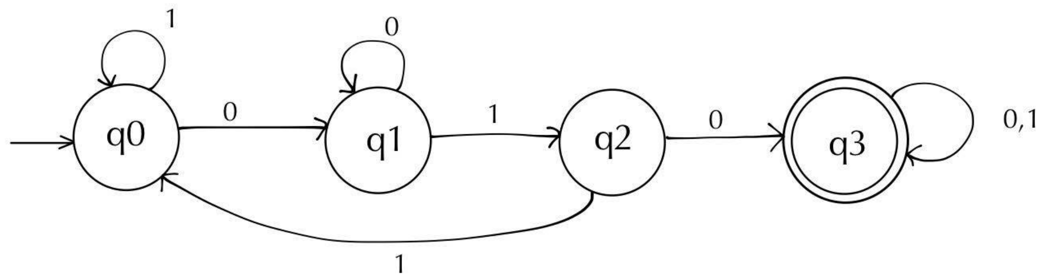


As can be seen in my DFA, I have a fourth dead state that is not drawn in the picture for conventional reasons. No 1 can ever follow directly after a 0, so if I am to get a 1 from state q_1, that means I go to a dead state where I will stay. From state q_2, all I can read in order to stay here is reading more 2's, since the words always have to end with a 2. That means, if I read a 1 or a 0, I have to go back to another state.

3 Task 3

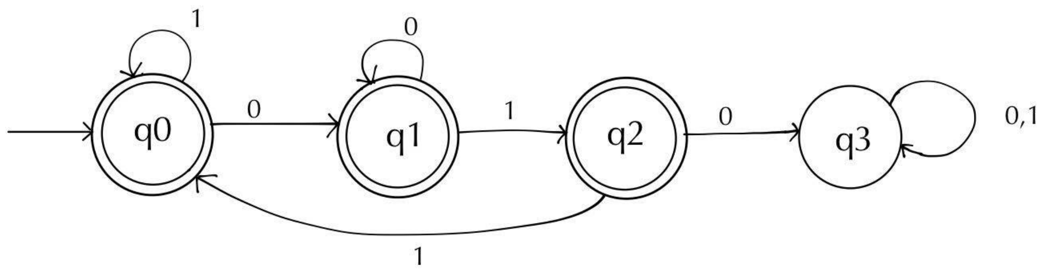
For this assignment I have the standing alphabet $\Sigma = \{0,1\}$.

a) In order to define a DFA that only accepts the words over the given alphabet which *do not* contain 010, I find it easier to first think of a DFA that *only* accepts words containing 010.

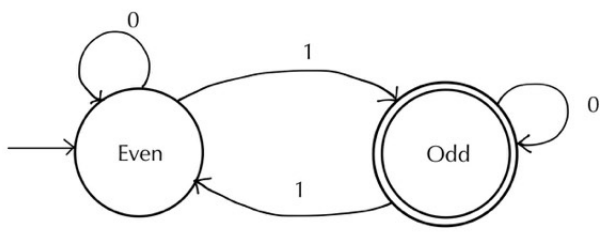


Now, with this DFA, I'd just have to think of the exact opposite, and the opposite in this case is the complement of the DFA. Getting the desirable DFA from the above is very easy, since all I have to do is change what my final state will be. The new DFA would accept any word that does not contain 010, which means it would accept 0,1, 01, 10, and so on. But as soon as I'd read 010, I have lost the chance of the DFA accepting my word.

Since this complement is the exact opposite of the DFA above, that means that it would have the opposite states and the accepting states. That is, all three states q_0, q_1, q_2 would be accepting, whereas q_3 is not an accepting state anymore. The result can be seen below.



b) For the b part I am now supposed to construct a DFA that accepts the words over the given alphabet with an odd number of 1's. This DFA will only need two states, one starting state for when we have words with an even number of 1's, and one accepting state for when we have an odd number of 1's. Between these two states we can have any number of 0's. The DFA will look like this:



c) Finally, now I need to construct the product of these two DFA's. Is the first DFA from a) is called D_1, and the one from b) D_2, then the new DFA D will be the product of D_1 and D_2, that is

$$D = D_1 \times D_2$$

Now, the definition of the product automaton is this:

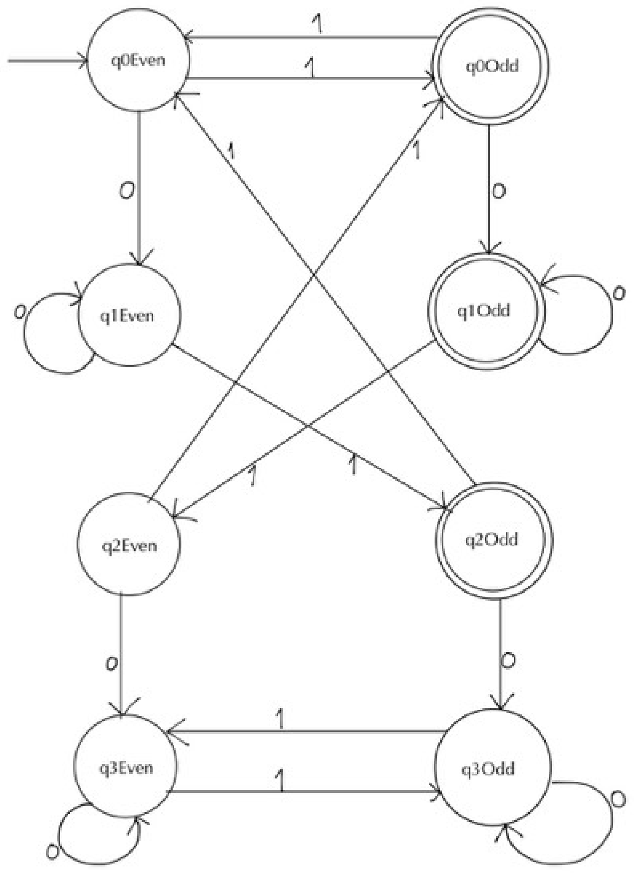
- Q $= Q_1 \times Q_2 = \{q_0even, q_0odd, q_1even, q_1odd, q_2even, q_2odd\}$ (all the possible combinations).

- $\delta((r_1, r_2, a) = (\delta_1(r_1, a), \delta_2(r_2, a))$, that is, if I read a symbol a from state q_0even, I'll have to know where I'd go if I'd only go from q_0, and where I'd go if I'd only go from *even*. Combining these two, the function will return one of the states in Q, and this is the definition of the transition function.

- $q_0 = (q_1, q_2)$, the starting state is the starting state from both DFA's. In my case the starting state will be q_0even.

- $F = F_1 \times F_2$. I'll get the accepting states by combining the accepting states from both the DFA's. For me that means the set $\{q_0odd, q_1odd, q_2odd\}$

On the next page one can see the resulting automaton:

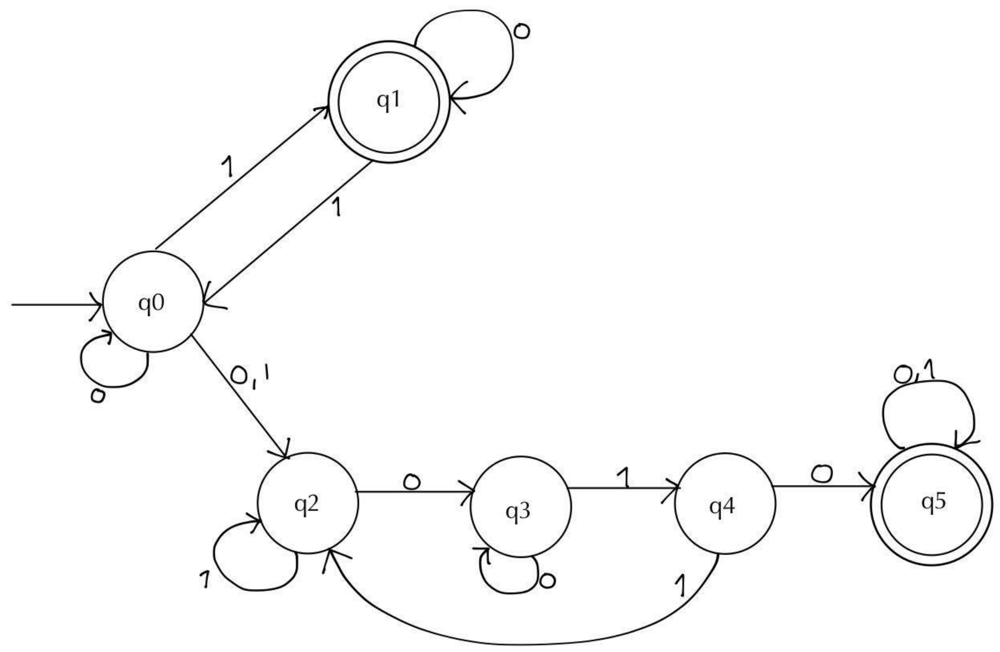

q0Even
q0Odd
q1Even
q1Odd
q2Even
q2Odd
q3Even
q3Odd
1
0

4 Task 4

a)For the last task I am to construct a NFA over the language $\Sigma = \{0, 1\}$ with the following properties:

- it should accept strings that contain 010 or
- it should accept strings that have an odd number of 1's.

My approach to this was to think of the problem as two separate DFAs, where the first one would have the first property and the second one with the other property. After this step I put them together into one NFA, where you can take several paths after reading one symbol. My resulting NFA looks like this:



Here, when you read a 1, you could either go to one of the accepting states since you at this time have an odd number of 1s. Or, you could go to q_2, and continue reading the input data and see if you get the sub word 010. In this NFA I have q_0 as a starting state and to final states, q_1 and q_5.

b) In b I am now supposed to perform the subset construction algorithm to my NFA, in order to get the corresponding DFA. Rather than trying to draw it right away (which would get tedious and I'd risk doing a lot of errors), I'm going to use the same idea as we did in the exercise, where we first made the transition table and then drew the transition diagram.

Here's how the transition table looks for the NFA (based on my drawing above):

	0	1
$\rightarrow q_0$	$\{q_0, q_2\}$	$\{q_1, q_2\}$
$\star q_1$	$\{q_1\}$	$\{q_0\}$
q_2	$\{q_3\}$	$\{q_2\}$
q_3	$\{q_3\}$	$\{q_4\}$
q_4	$\{q_5\}$	$\{q_2\}$
$\star q_5$	$\{q_5\}$	$\{q_5\}$

And now, from this I do the subset construction. If the definition for a NFA is

$$N = (Q_N, \Sigma, \delta_N, q_0, F_N)$$

, then the definition for a DFA D constructed from a NFA, is this:

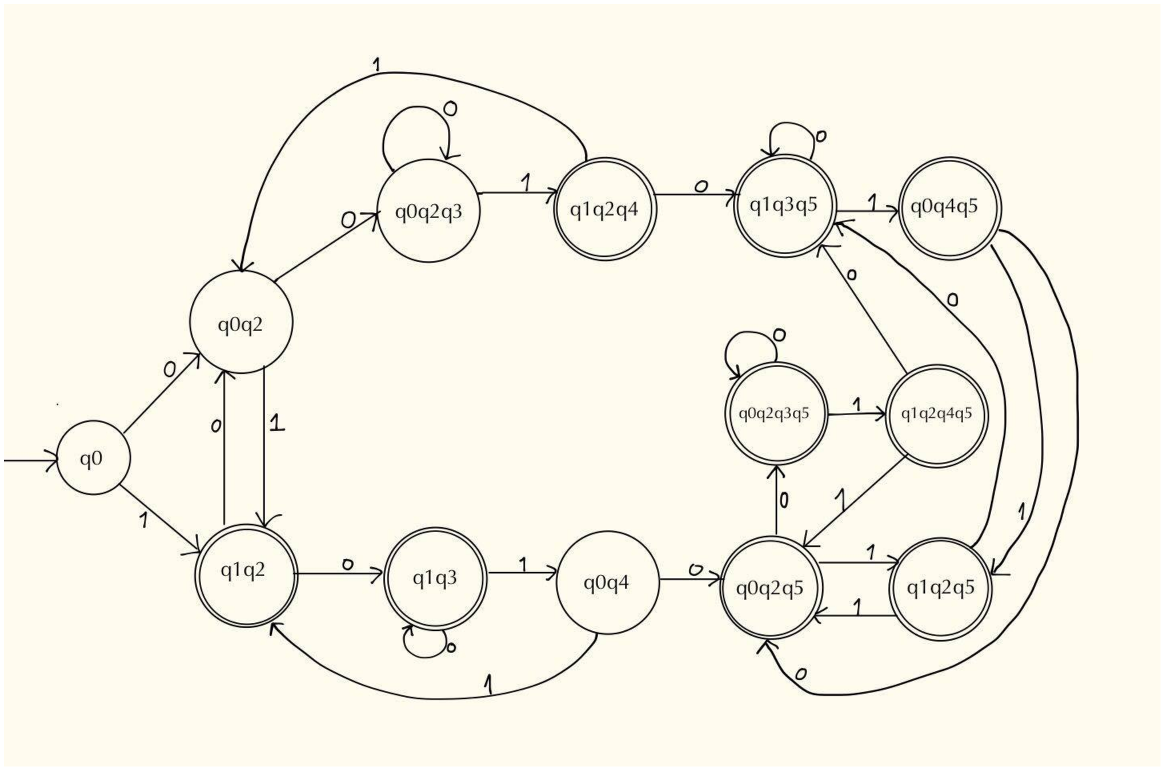
$$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

Where $Q_D = \rho ow(Q_N)$, i.e. Q_D will be the set containing all subsets of Q_N. This means that, in worst case scenario, Q_D will contain $2^6 = 64$ elements (which gives me reasons to believe there is an alternative solution with fewer states).

Now what I'll do is I'll start with the same first line in my new transition table as in the old one. From thereon I will simply add a new line to the transition table as long as I get a new combination of states.

	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_2\}$	$\{q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_1, q_2\}$
$\star\{q_1, q_2\}$	$\{q_1, q_3\}$	$\{q_0, q_2\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_1, q_2, q_4\}$
$\star\{q_1, q_3\}$	$\{q_1, q_3\}$	$\{q_0, q_4\}$
$\star\{q_1, q_2, q_4\}$	$\{q_1, q_3, q_5\}$	$\{q_0, q_2\}$
$\{q_0, q_4\}$	$\{q_0, q_2, q_5\}$	$\{q_1, q_2\}$
$\star\{q_1, q_3, q_5\}$	$\{q_1, q_3, q_5\}$	$\{q_0, q_4, q_5\}$
$\star\{q_0, q_2, q_5\}$	$\{q_0, q_2, q_3, q_5\}$	$\{q_1, q_2, q_5\}$
$\star\{q_0, q_4, q_5\}$	$\{q_0, q_2, q_5\}$	$\{q_1, q_2, q_5\}$
$\star\{q_0, q_2, q_3, q_5\}$	$\{q_0, q_2, q_3, q_5\}$	$\{q_1, q_2, q_4, q_5\}$
$\star\{q_1, q_2, q_5\}$	$\{q_1, q_3, q_5\}$	$\{q_0, q_2, q_5\}$
$\star\{q_1, q_2, q_4, q_5\}$	$\{q_1, q_3, q_5\}$	$\{q_0, q_2, q_5\}$

The resulting transition diagram will look like this:


q0
q0q2
q1q2
q0q2q3
q1q2q4
q1q3
q0q4
q1q3q5
q0q4q5
q0q2q3q5
q1q2q4q5
q0q2q5
q1q2q5
0
1