

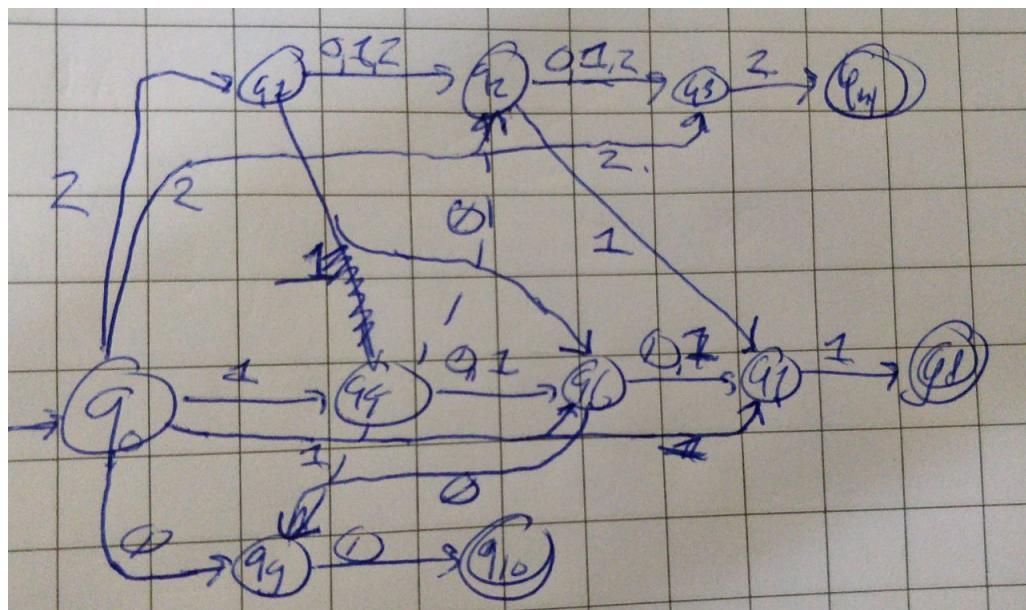
# Languages and Automata Assignment 3

Dibran Dokter s1047390

February 21, 2020

1

a)



This automaton uses the non-determinism to allow for a shorter word, so we can accept 00 and so forth. For the rest we use non-determinism to allow a change between a word with 2's and 1's or 0's. This can be done by taking a path from the top row of states downwards. The automaton does not allow higher numbers in between since there are no transitions the other way around.

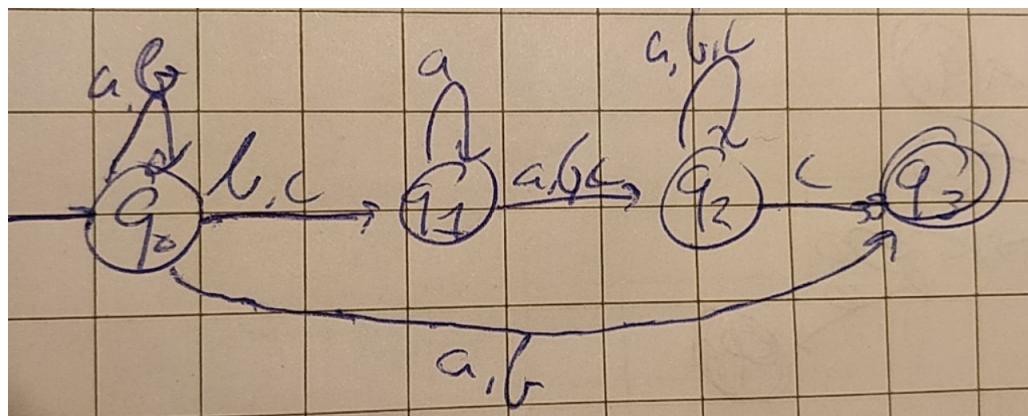
b)

In the automaton of 1a 2101 is accepted if we take the path  $q_0 \rightarrow q_2 \rightarrow q_6 \rightarrow q_7 \rightarrow q_8$  where  $q_8$  is a final state. With the parsing of this word we can see the change in the layers of states in action when going from  $q_2$  to  $q_6$ .

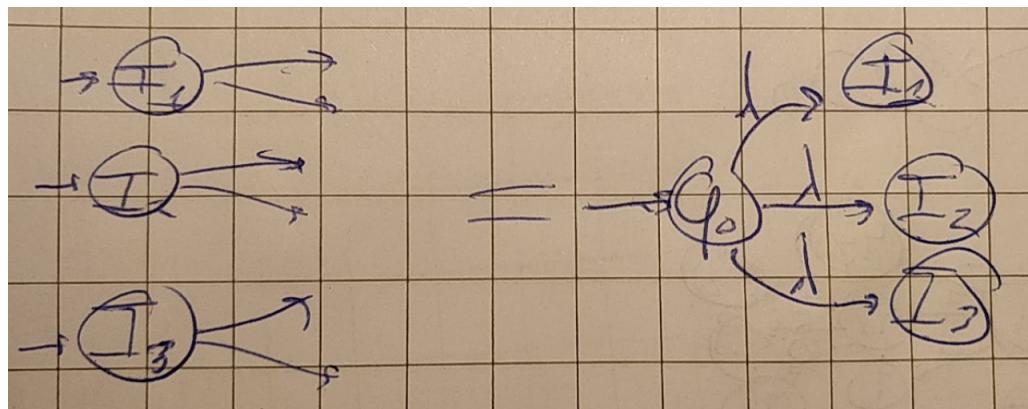
The word 121 is not accepted since we take the following path:  $q_0 \rightarrow q_4$  and then we are unable to parse 2 since there is no transition for it.

2

a)

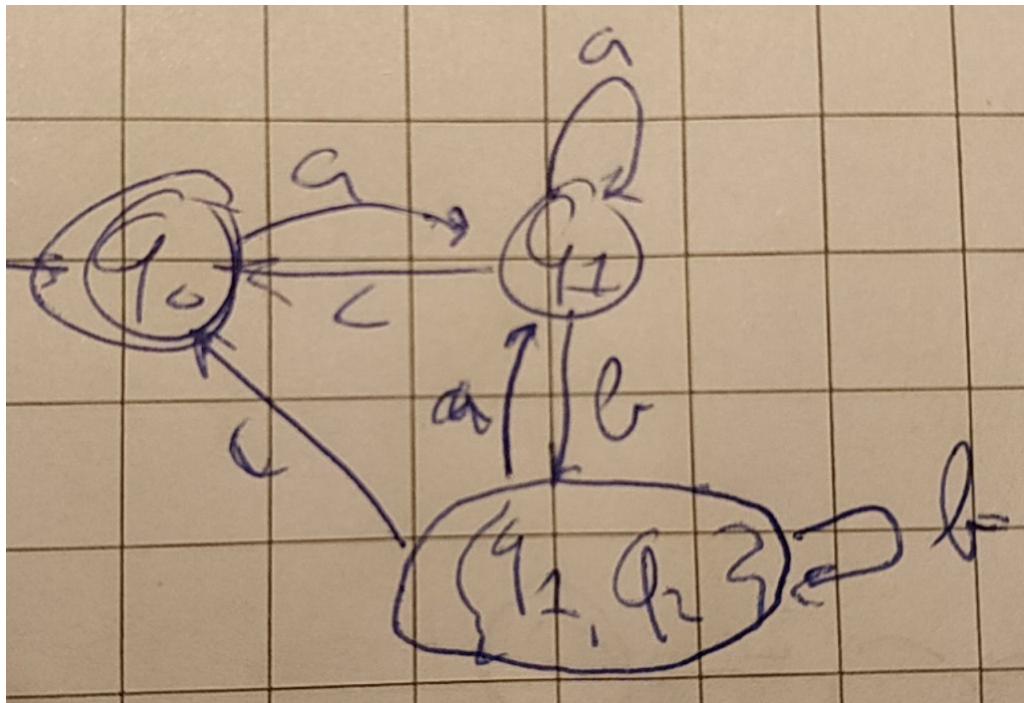


b)



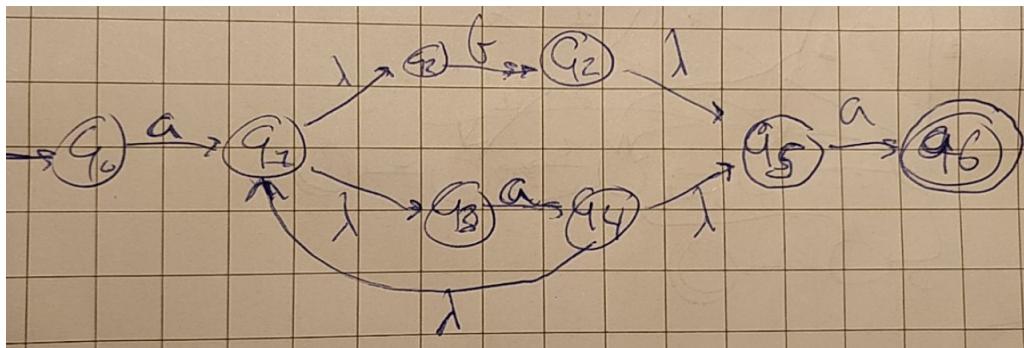
When we have multiple initial states (denoted by  $I_n$ ) we can replace those states by the right side as shown. We place a single initial state before the other initial states and add  $\lambda$ -transitions between the initial state and the  $I_n$  states.

3



All the states from  $M$  relate to the same state in  $M'$  except for  $q_1$ . This state is now split between the original state  $q_1$  and the new set state  $\{q_1, q_2\}$ . We do this to make sure that we only get one  $b$  transition, to the set state. To make sure that the original automata is reserved we need the  $a$  transition back to  $q_1$ .

4



To create this automata we can use all the separate steps to create a NFA- $\lambda$  from a regular expression. We first use the  $a$  (for  $a \in A$ ) step to create the part  $a$ . Then we take the part  $e_1 \cup e_2$  to take the  $b \cup a^*$  part. We then take the  $a^*$  part to make the  $a^*$  part. In the end we take the last part and connect all the parts using the  $e_1 e_2$  part.