Homework 2

CSC 445-01: Theory of Computation

Matthew Mabrey, Luke Kurlandski

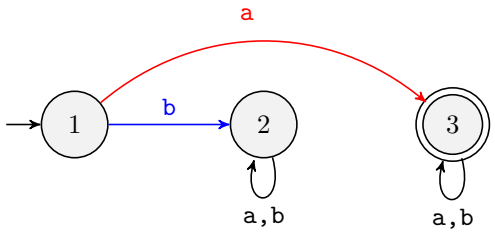
February 22, 2021

1.4

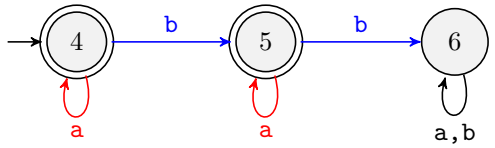
We use blue for 'b' and red for 'a' when space is limited

e

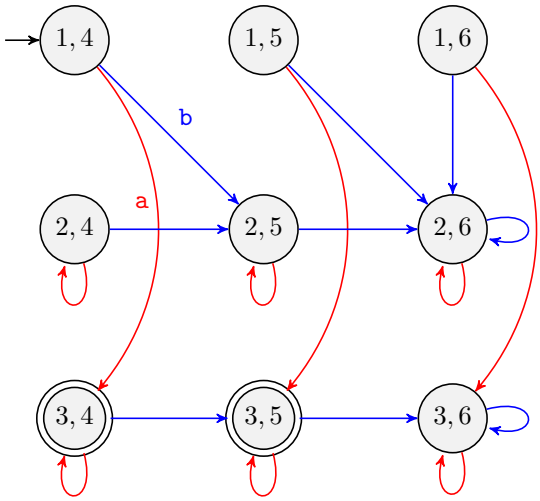
w starts with an 'a'



w has at most one 'b'

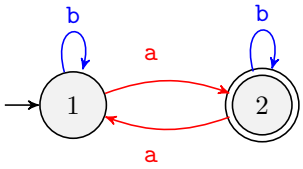


w starts with an 'a' and has at most one 'b'

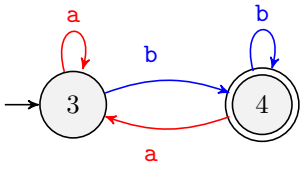


f

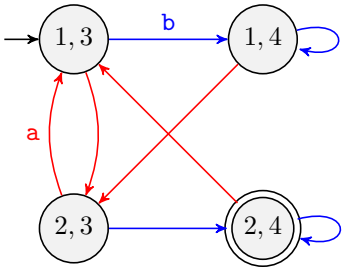
w has an odd number of 'a'



w ends in a 'b'

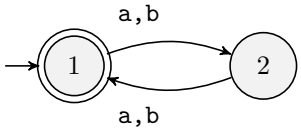


w has an odd number of 'a' and ends in a 'b'

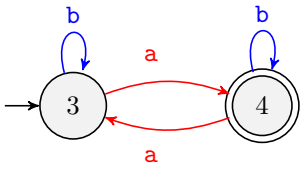


g

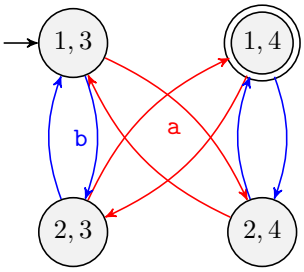
w has an even length



w has an odd number of 'a'



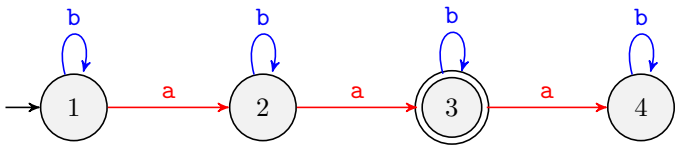
w has an even length and has an odd number of 'a'



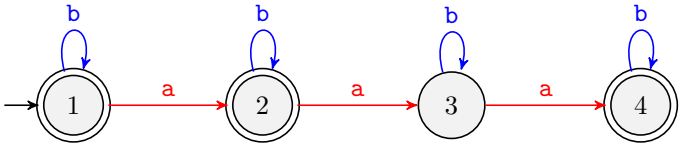
1.5

g

w is any string that contains exactly two 'a'



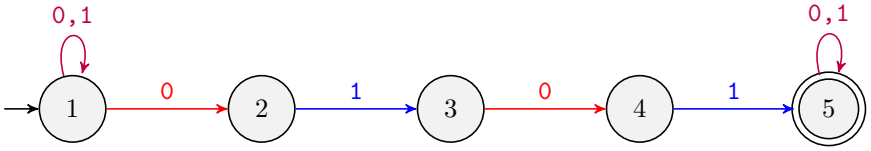
w is any string that does not contain exactly two 'a'



1.7

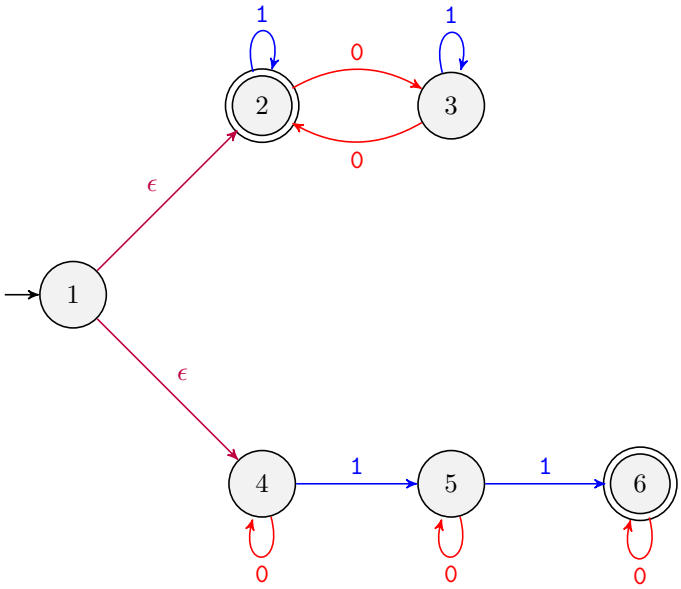
b

A is an NFA that accepts any string w containing the substring 0101 with 5 states



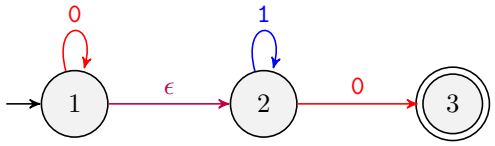
c

A is an NFA that accepts any string w containing an even number of 0s, or contains exactly two 1s with 6 states



e

A is an NFA that accepts any string w containing 0*1*0+ with three states



1.31

If A is regular then there is an FSA that accepts A with a finite number of states which we will reverse the edges between in the transition function δ. We will then move the initial states $q_0, q_1, ..., q_n$ into the set of final states F and make all of the states previously in F initial states. Now this FSA which originally accepted A will accept the reversed strings w^R in the language A^R and thus if A is regular then A^R is regular.

1.33

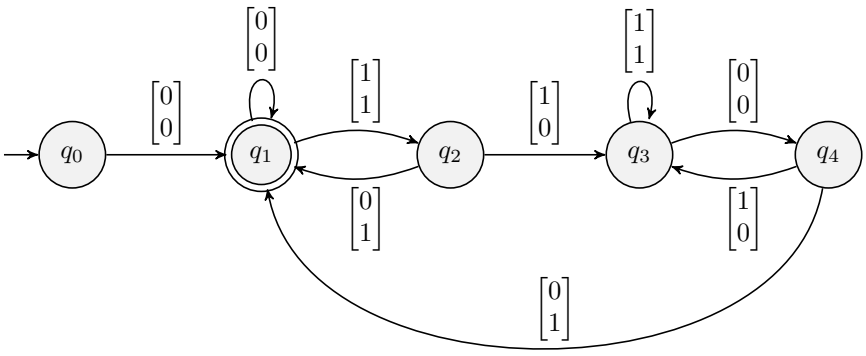
From problem 1.31, we know we are allowed to "wind the tape backwards" i.e., we will demonstrate that C^R is a regular language, thus prove that C itself is regular. To prove that C^R is regular, we will construct a nondeterministic finite automata that recognizes C^R.

For each symbol, we will read its top bit and judge whether or not the bottom bit is correct such that $3\text{row}_1 = \text{row}_2$. To do so, we track two conditions: 1) whether or not 3row_1 results in a carry-out in base two and 2) what the previous top symbol is. With these two pieces of information, given a top bit, we can check to see if the bottom bit is correct.

These two pieces of information result in an automata with 4 states to keep track of, but we had an additional start state.

- q_0 : the start state, to prevent the automata from accepting the empty string
- q_1 : the state where there is no carry-in from previous step and the previous top symbol is a 0
- q_2 : the state where there is a carry-in and the previous top value is a 1
- q_3 : the state where there is no carry-in and the previous top value is a 1
- q_4 : the state where there is a carry-in from previous step and the previous top value is a 0

Our transition function merely Recall that we read the string backwards, so the binary number is read from the 2^0th place first.



1.34

This NFA is sufficient to accept any string w composed of the 2 row binary matrices where the top row is larger then the bottom row when read from left to right as a binary number. The larger binary number is determined by the first matrix where one row value is different from the other row value. This is because the most-significant unique binary digit is greater than all less-significant binary digits summed together. So even if the other row of binary digits had all 1's it still would not be greater than the other row which received the first unique 1.