Math Circles Lecture 1: Finite State Machine Exercises

Troy Vasiga
David R. Cheriton School of Computer Science
University of Waterloo
tmjvasiga@cs.uwaterloo.ca

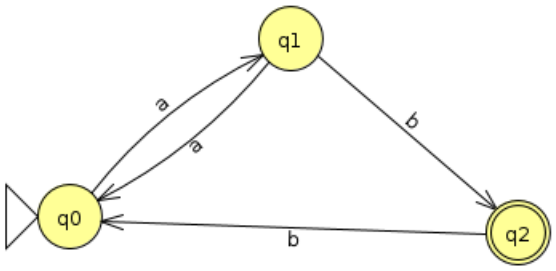
1 Introduction

In lecture, we discussed *Finite State Machines* (also known as *FSMs, finite state automata,* or *FSAs*). We discussed regular expressions, DFAs, NFAs, and λ-NFAs.

2 Exercises

The questions are relatively ranked from easier questions at the beginning to trickier questions at the end.

1. Construct a FSA for the following regular expression: $(ab)^*$
2. Construct a FSA for the following regular expression: a^*b^*
3. Construct a FSA over the alphabet $\Sigma = \{0, 1\}$ which accepts only the words which have exactly three 1's. That is, 01010001 should be accepted, but 1111 and 010100 should be rejected.
4. What is the regular expression given by this machine?



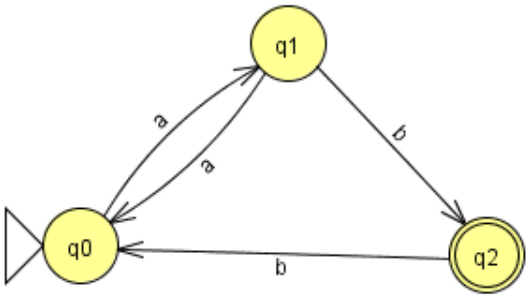
5. Construct a FSA which accepts even base-10 positive integers. For example `1020` and `2` should be accepted, but `1111` should be rejected. (**HINT:** Non-determinism will be your friend. You can do this with 2 states and 2 transition arcs nondeterministically, or with 2 states and 4 transition arcs deterministically.)

6. Repeat the previous question, except accept only odd base-10 positive integers. That is, you should accept `3` and `333433`, but `1110` should be rejected. What has changed from your answer to the previous question?

7. Speaking of threes, construct a FSA which accepts decimal numbers which are divisible by 3. Note that 0, 3, 21, 33960210 are divisible by 3. (**HINT:** You should need exactly 4 states to do this, and think about what it means to be divisible by 3). Try it on some numbers which are divisible by 3 (like 3 and 27) and other numbers which are not divisible by 3 (like 4 and 83).

8. Again, since we are on the topic of threes, construct a DFA which accepts all words have the subword `abba` within them. You should check to make sure that `abbba` and `ababa` are rejected and that `abbbabba` is accepted.

9. What is the regular expression which is equivalent to the language accepted by this machine?



10. Construct a FSA over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ which accepts words which contain an even number of `a`'s. There are no restrictions on the number of `b`'s or `c`'s. You should verify that `aabaacc`, `b` and `abacababc` are accepted and that `aaacacac` and `abb` are rejected. Hint: You should use two states and keep track of the "parity" (even or odd) of the number of `a`'s you have read in.

11. Find a regular expression for the FSA in Question 10.

12. Construct a FSA over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ which accepts words which contain an even number of `a`'s and an odd number of `b`'s. There are no restrictions on the number of `c`'s. You should verify that `aabaacc`, `b` and `abacababc` are accepted and that `aaaacacac` and `bb` are rejected. Hint: You need 4 states, and you should keep track of the parity of both `a`'s and `b`'s.

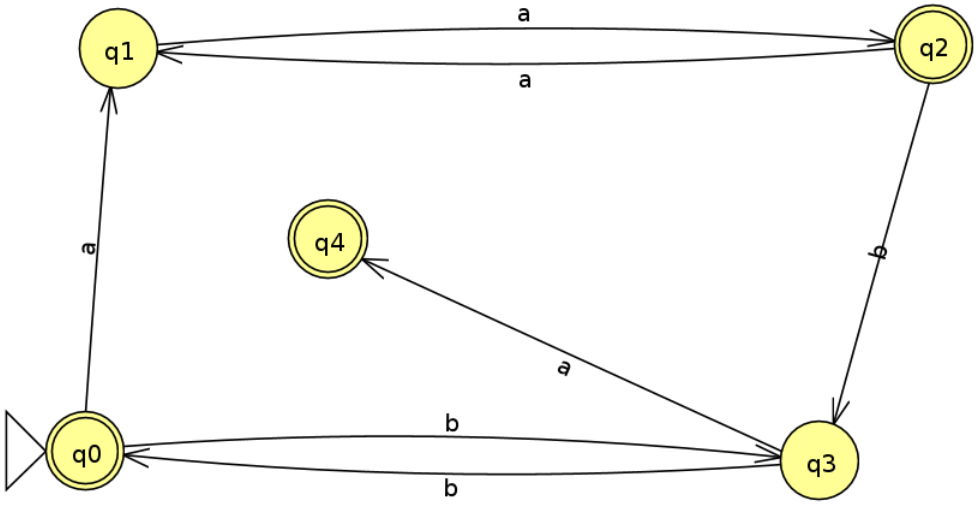
13. Construct the FSA for the following regular expression:

$$\texttt{a}^*\texttt{bc}^*|\texttt{a}(\texttt{bc})^*|(\texttt{abc})^*$$

(Hint: you should think about creating three distinct FSMs, for each part of the regular expression. Then, connect them using a λ-transition from the start state.)

14. Find a regular expression for the FSA in Question 12. Hint: this is really hard to do by hand.

15. Consider the following DFA:



Can you write a smaller (i.e., less states) DFA which is equivalent to the above DFA? How can you prove that your new DFA is minimal?

16. Consider $L(M)$, where M is the DFA shown in Question9. Create a DFA M' such that $L(M') = \Sigma^* - L(M)$: that is, all the words that M accepts should be rejected by M', and all the words that M rejects should be accepted by M'.

3 Acknowledgments

You should try out some of your answers using a free tool called JFLAP. The latest version of JFLAP can be found at: `http://www.jflap.org`