Creating Finite State Machines using JFLAP

Carmen Bruni (cbruni@uwaterloo.ca) with thanks to Troy Vasiga

1 Exercises

Try the following. See how far you get. Write down your answers (and save them to a file, if you wish). The questions are relatively ranked from easier questions at the beginning to trickier questions at the end.

1. Construct a FSM for the following regular expression:

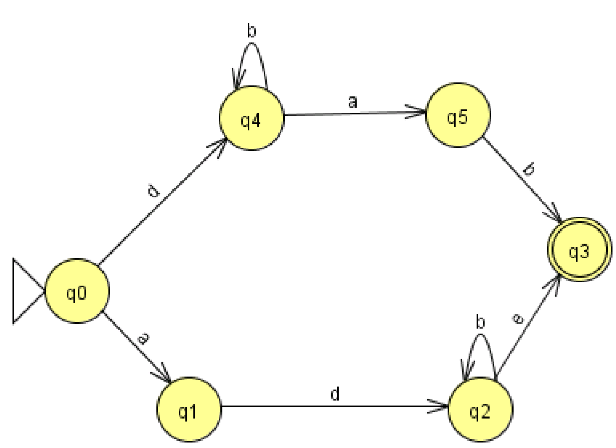
$$(ab)^*$$

2. Construct a FSM for the following regular expression:

$$a^*b^*$$

3. Construct a FSM 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. Give a FSM for your postal code. For example, the postal code for the University of Waterloo is `N2L 3G1`. However, people write this a whole bunch of different ways. That is:

   - n2l3g1
   - N2L-3G1
   - N2L 3g1
   - N2L 3G1

   are all valid. Write a postal code recognizer which accepts these variations on your postal code (**HINT**: you may want to use λ-transitions.)

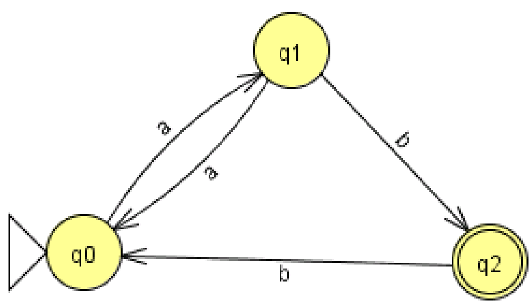
6. Construct a FSM which accepts even base-10 positive integers. For example `1020` and `2` should be accepted, but `1111` should be rejected. (**HINT:** Non-determinism will be your friend. You can do this with 2 states and 2 transition arcs nondeterministically, or with 2 states and 4 transition arcs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

7. 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?

8. Speaking of threes, construct a FSM 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).

9. Construct a DFA which accepts all words having the subword `abba` within them. You should check to make sure that `abbba` and `ababa` are rejected and that `abbbabba` is accepted.

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



11. Construct a FSM over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ 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.

12. Find a regular expression for the FSM in Question 11.

13. Construct a FSM over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ 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.

14. Find a regular expression for the FSM in Question 13. Hint: this is really hard to do by hand. Use JFLAP to do this work for you, which should convince you how hard it is.

15. Construct the FSM for the following regular expression:

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

(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.)