CS46, Swarthmore College, Spring 2014
Homework 2 (due Thursday 6 February)

This assignment consists of two parts: a written homework and a programming portion.

Part 1: Written homework

- Please read Sipser Chapter 1-1.2 before starting this homework.

1. Draw state diagrams for DFAs recognizing the following languages:

   (a) Sipser 1.4a: $\{w|w$ has at least three a's and at least two b's $\}$

   (b) Sipser 1.4e: $\{w|w$ starts with an a's and has at most one b $\}$

   (c) Sipser 1.5g: $\{w|w$ does not contain exactly two a's $\}$

   (d) Sipser 1.6i: $\{w|w$ every odd position of w is an $a\}$

   For this problem you may ignore the instructions for Exercise 1.4 and just give the state diagram, but you might find the instructions helpful. Assume the alphabet is $\Sigma = \{a, b\}$ even though 1.6 uses $\{0, 1\}$

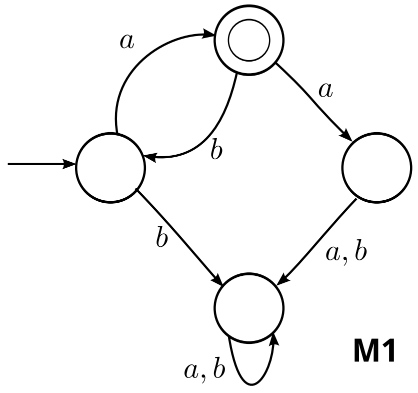
2. Write concise descriptions of the languages recognized by the four DFAs defined on the next page.

3. Let our alphabet $\Sigma = \{0, 1\}$ and let

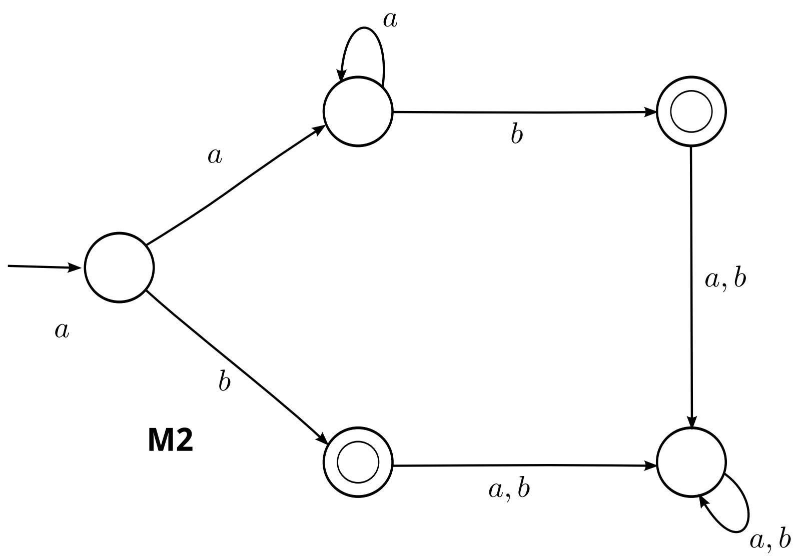
$$L = \{w : w \text{ contains the substring } 0ab0 \text{ or } 1ab1 \text{ for some symbols } a, b \in \Sigma\}$$

   (a) Draw a state diagram for a DFA that accepts L. Please take extra care to be neat; be sure to group related states together, and (where possible) give your states meaningful names or descriptions. I recommend that you use a single piece of paper for just this part of this problem.

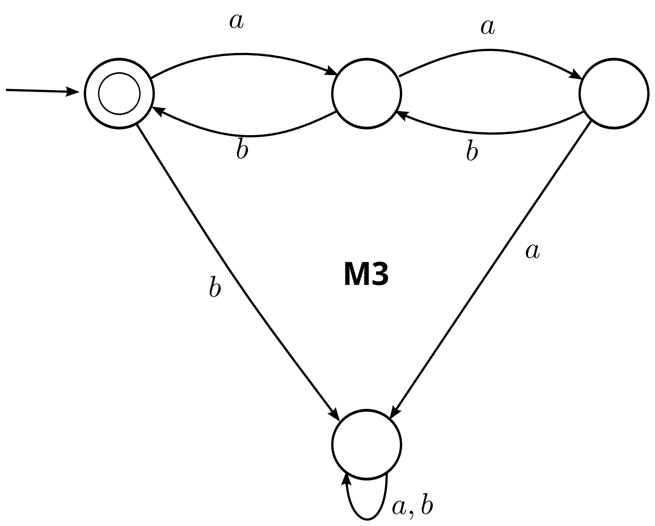
   (b) Draw a state diagram for an NFA that accepts L. Your NFA should have substantially fewer states than your DFA from part (a).



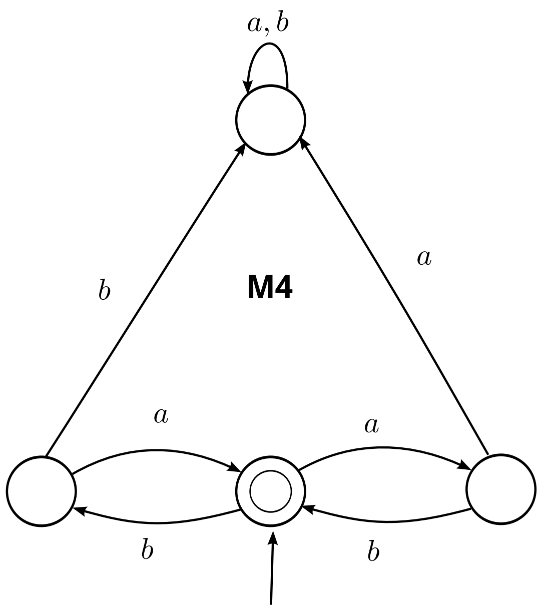
M1



M2



M3



M4

4. Binary addition is regular! Let our alphabet Σ be the set of all size 3 binary vectors:

$$\Sigma = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

A correct addition of two binary numbers can be represented by a string in Σ^*. For example:

$$\begin{array}{cccc} & 0 & 1 & 0 & 1 \\ + & 0 & 1 & 1 & 0 \\ \hline & 1 & 0 & 1 & 1 \end{array}$$

would be represented by the following string of four symbols from Σ:

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Let the language L be the set of all strings in Σ^* representing correct binary additions. Show that L is regular by constructing a DFA for this language. You can just give the state diagram; you don't need to give the formal description. Briefly explain what each state of your DFA represents. *Note:* Remember that a DFA scans its input string from left to right but addition is performed from right to left; this is what makes this problem interesting. It's possible to solve this problem with a DFA with very few states...

Part 2: Programming Exercise

Implement a generic DFA in the programming language of your choice. I recommend python and the examples here will use python. For this portion of the assignment, you may work with a partner. Your code should accept two files as input. The first file is a description of the machine. The sample format for this file is shown below

```
ee 0
eo 0
oo 0
oe 1

ee a oe
ee b eo
eo a oo
eo b ee
oo a eo
oo b oe
oe a ee
oe b oo
```

Each non empty line has either two or three strings separated by spaces. The lines with two strings represent states. The first string is the state name. The second string is either 0 or 1. A state is a final state if the second string is one. The first state listed is the start state (e.g., ee) The lines with three strings represent the transition function. ee a oe is interpreted as "if the machine is in state ee and reads an a, the machine transitions to state oe."

You may assume that all states are listed before all transitions. The second file is just a list of string that are inputs to the machine. See the example below.

```
ab
aabb
abb
baa
aaabbb
aabb
ababb
a
bb
aa
bbb
```

Your code should test each input string against the machine and label each string as accepted or not accepted.

```
python dfa_sol.py machine1.txt input1.txt
ab: False
aabb: False
abb: True
baa: False
aaabbb: False
aabb: False
ababb: False
a: True
bb: False
aa: False
bbb: False
```

The key to this project is (1) understanding how a DFA works and (2) understanding how a python dictionary or other similar hash map works. Note that both the State class and the Machine class use dictionaries (indicated by {} to store info. If you are unfamiliar with dictionary syntax, giyf, or use one of the python refs in the lab, or a partner (your partner can either tell you how they work, or fetch the book for you). For State objects, the key for the dictionary should be a symbol and the value the name of another state. For Machine objects, the dictionary should store names of states as keys and return the actual state objects as values. Feel free to ask questions, but with the start code provided, you should have a basic idea of how to proceed

If you are working with a partner, only one person needs to hand in the lab. Include both names at the top of your program.