COSC341 TUTORIAL 1, SOLUTIONS

The theme of this tutorial is to experiment with small deterministic finite state automata (DFAs) in order to try and get some feeling about what they can do, and similarities and differences between them.

The alphabet $\Sigma = \{a, b\}$ is to be used throughout.

In answering the "What languages do they accept?" questions, a simple description in English is fine, e.g., "all non-empty strings" or "all strings starting with an a".

1. *How many different DFAs are there having only one state? What languages do they accept?*

   If a DFA has a single state, then it must be the start state and both transitions must loop on that state. So the only choice to make is whether or not it is an accepting state – that is, there are exactly two one-state DFAs. The one in which the start state is accepting accepts all string (i.e., its language is Σ^*), while the one where it is not accepting accepts no strings at all (i.e., its language is the empty set, $\{\}$ – the set containing **no** strings, not even the empty string).

2. *A state in a DFA is* unreachable *if there is no input sequence that leads to that state. Note that the start state is always reachable since the empty sequence leads to it. If we remove the unreachable states from a DFA how does that affect the language it accepts?*

   Since, by its very nature, a DFA can never enter an unreachable state, the question of whether or not a string is accepted by a DFA is unaffected by the presence of any unreachable states. That is, we can remove any unreachable states and the DFA still accepts exactly the same strings.

3. *How many different DFAs are there having only two states? What languages do they accept? Are they all different?*

   Call the two states 0 and 1. Since names don't matter we can (as usual) make the start state be state 0. So the choices we have to make are:

   - which states are accepting, and
   - what are the transitions between states?

   All of these choices are independent of one another, so every time we make a choice we should multiply the number of possible DFAs by the number of options we had to choose from. There are two choices of accepting/not-accepting for each of the two states. For each transition, given its start point

there are two choices for its end point. In other words there are four more binary choices to be made (two transitions at each of the two states). So the total number of binary choices we have to make is six, and the number of two-state DFAs is

$$2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^6 = 64.$$

Before turning to the languages that these DFAs accept, consider the same problem when there are k states and an alphabet of size n. Now we have k binary choices (one for each state as to whether its accepting or not), and kn, k-ary choices (there are kn transitions to have endpoints chosen and k possibilities for each). Multiplying all those numbers together gives

$$2^k \times k^{kn}$$

a number that gets rather large rather quickly!

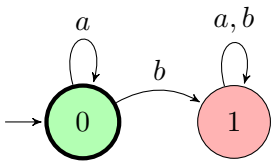
To figure out what languages are accepted by the 64 two-state DFAs over $\Sigma = \{a, b\}$, we obviously don't want to think about them one at a time.

First note that if both states are accepting then the language is Σ^* and if both are non-accepting it's $\{\}$, regardless of the transitions. So that takes care of 32 of the DFAs in one fell swoop!

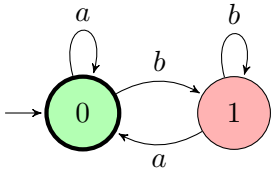
We're left with the DFAs with one accepting and one non-accepting state. If we interchange the two, then we just change the language to its complement (strings that were previously accepted aren't, and vice versa). So it's enough to consider those machines in which the start state is accepting and the other state isn't – that's gotten us down to 16 remaining machines.

In four of those we have a and b loops on the start state (i.e., the other state is unreachable), and these all accept Σ^*.

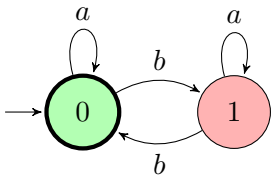
If we have one loop on the start state and one transition to the other state, we might as well have the loop be on a and the transition on b (and just exchange the letters in describing the languages accepted to deal with the other case). It remains to consider how the transitions from state 1 work. We have made use of all the symmetry we can, so there are four cases to consider shown below.



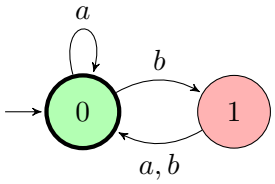
All repetitions of a 0 or more times



The empty word, and words ending with a

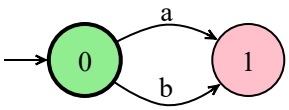


Words with an even number of b's



Words ending with a or an even number of b's

We must now consider the cases where both a and b cause transitions away from the start state, as shown below (where the transitions from state 1 have not yet been decided):



If both transitions from state 1 go back to 1, only the empty word is accepted, i.e. the DFA's language is $\{\epsilon\}$. If both a and b cause transitions back to state 0, then all words of even length (including 0) are accepted. If only a (or only b) causes a transition back to 1, then ϵ and words in which every second letter is an a (respectively b) are accepted.

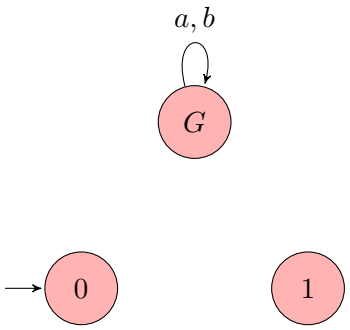
Finally, remember that to list all the languages accepted by two-state, two-letter DFAs we'd also have to complement the sets above and exchange the roles of a and b.

4. *A state in a DFA that is not accepting and all of whose transitions are loops back to itself is sometimes called a "garbage state". Why?*

   Once you enter such a state you can't get out, and you can't accept regardless of any further input. So this is effectively like a trash can.

5. *How many three state DFAs are there all of whose states are reachable and exactly one of which is a garbage state? What languages to they accept?*

   I'm not really sure I should have asked this question. Let's begin with the basic structure – G being the garbage state.



   Let me do the counting part. G is not accepting but the other two might (or might not be) so that's four possibilities to multiply by. We can't have both letters loop on 0 or else we violate reachability.

   Suppose that one goes to 0 and the other to 1 (two possibilities). Among the nine possibilities for transitions from 1, four fail to reach G, so five of them are allowed. So there are 10 automata of this type.

   If both go to 1, the same analysis for transitions from 1 applies so this gives five more automata.

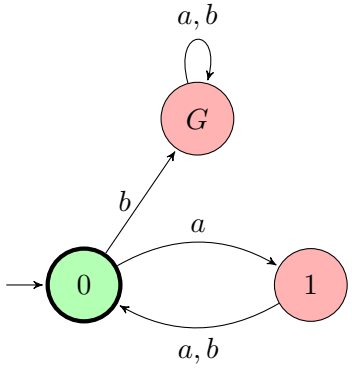
   If one goes to 1 and the other to G (two possibilities) then the transitions from 1 are unrestricted, giving nine possibilites, and 18 total of this type.

   So the total is:

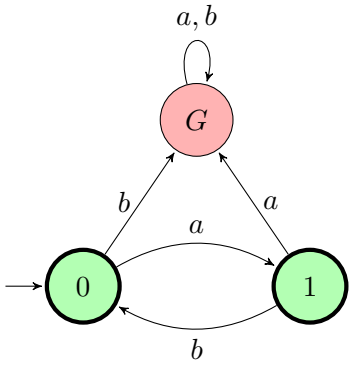
$$4 \times (10 + 5 + 18) = 132$$

   such automata.

   I'm sure it's not interesting to list all the accepted languages. Let me just do a couple of examples:



This accepts words of even length all of whose b's (if any) occur in odd-indexed positions (indexing from 0). If state 1 is also accepting the "of even length" condition can be dropped.



This accepts any words that are prefixes of $abababab\cdots$