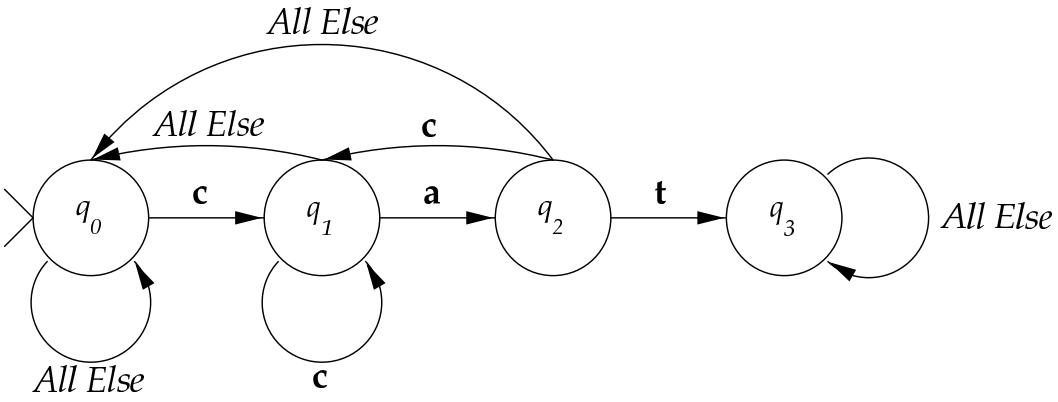
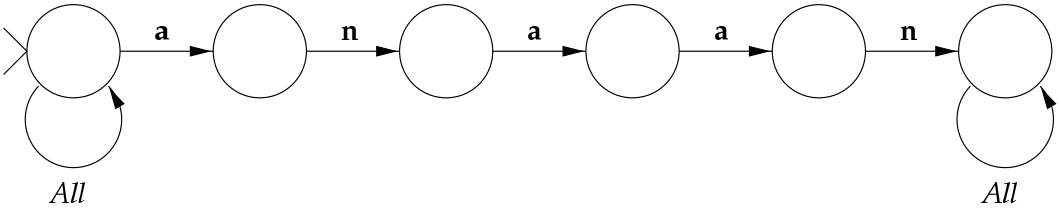
Homework 1 Solutions

1. There are two important thing to remember on this problem: First, the DFSA must provide for the case where you start down the automata but find that it is a partial match. For instance, **caricature** will not be accepted if the *All else* transtions back to q_0 are not present. Second, words like **imboccatura** (Italian for "mouth") and **cacatúa** (Spanish for "cockatoo") require transitions from q_1 and q_2 to q_1.

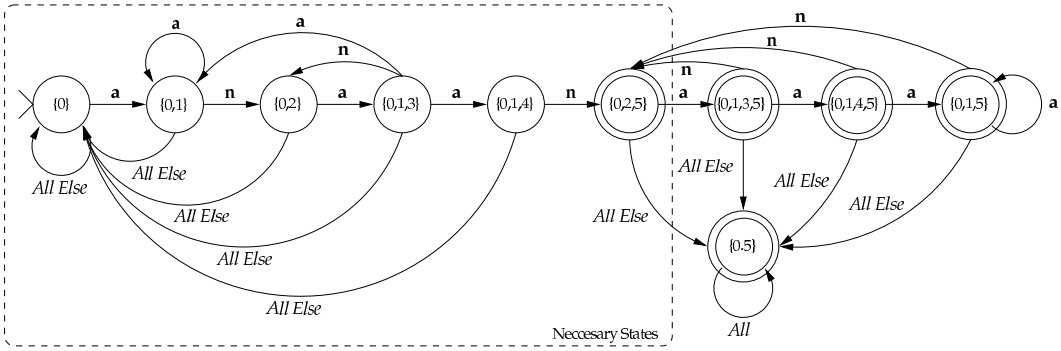


2. This shows the extraordinary power of representing finite state automatas in non-deterministic terms. While we know that NFSAs aren't any better than DFSAs, the NFSA representation is often much simpler.



3. It was possible to do this question without using the formal constructive proof found in Lewis and Papadimitriou, Theorem 2.2.1, but it is also very easy to get it wrong. By using that construction, we can develop a DFSA to recognize **anaan.** I've included the state sets that resulted from that construction.

   The construction, if done exactly as specified, will produce a DFSA with ten states. However, according to a sentence at the top of page 109 in Lewis and Papadimitriou (section 2.6, not assigned), an NFSA that is designed for the purpose of recognizing substrings *always* has a corresponding DFSA with the same number of states (though the DFSA invariably has many more transitions). Looking at the ten states, then, we can eliminate four as unnecesary (the six essential states are inside of the dotted line).



4. Regular expressions are very well suited for substring matching. The two expressions are: $\Sigma^* cat \Sigma^*$ and $\Sigma^* anaan \Sigma^*$. Note the close correspondence between the regular expression form and the NFSA form.

5. The problem with this approach is the way in which partial matches are taken care of. For instance, consider the substring **aaaaab** working on a file containing **aaaaaaaaaaaaaaaaaac.** The pattern matcher would consider most of the a's in the file five times (only the last four towards the end would be considered fewer times). This results in more characters being considered than are actually in the file.

6. We have actually solved this problem already for the substring **anaan**, above. One approach is to have a three step process. First, create an NFSA to handle the substring. This is a simple:

   (a) Assume the substring has n characters
   (b) Create an NFSA with $n+1$ states, labeled q_0 through q_n
   (c) From each state q_i to q_{i+1}, create a transition on character $i+1$
   (d) Create an *All* transition from state q_0 to itself
   (e) Create an *All* transition from state q_n to itself

   This will produce a result much like the answer to question 2, above. Second, apply the construction of Lewis and Papadimitriou, Theorem 2.2.1 to translate the NFSA into the equivalent DFSA. Third, implement the DFSA as a program, which they are well suited for (while NFSAs are not).

7. There are two stages to the construction of section 2.3. First, each state must be expanded along its epsilon transitions. A DFSA has no epsilon transition, however, so this step has no effect. Second, while the construction creates transitions between *sets* of states, a DFSA has no cloning, and will therefore only have sets containing one state. The transitions and states in the new machine, therefore, will be the same as those in the original machine. The construction, when applied to a DFSA, produces a machine identical to the one with which it started.

8. The easiest way to prove this is a proof by contradiction, using the fact that regular langauges are closed under intersection:

   Assume that $a^n b a^m b a^{m+n}$ is a regular. We know that the language $a^* b b a^*$ is regular. Therefore, the intersection of the two must also be regular. This intersection can be expressed as $a^n b b a^n$, which we know is not regular. Therefore, our assumption is wrong, and $a^n b a^m b a^{m+n}$ is not regular.