DFA Minimization
Martha Kosa

By now you have learned how to design DFAs and NFAs. NFAs could sometimes be simpler to design than DFAs, and could have fewer states. However, running a string through an NFA could be more complex because of the need to keep track of multiple active configurations. This is an example of a tradeoff. You will see many more tradeoffs during your study of computer science. Every NFA can be converted to an equivalent DFA. You learned and practiced with the conversion algorithm.

Suppose that you have designed a DFA or have converted an NFA to an equivalent DFA and that you have tested it to make sure that it accepts the language that you need. Is it the most efficient solution? How is efficiency measured? You study space and time complexity (mostly time complexity, because memory is not as restricted now as it used to be) with respect to algorithms. The time complexity for processing a string in a DFA is O(n), where n is the length of the string. Constant work is needed to process each character because each state has $|\Sigma|$ transitions associated with it, where Σ is the alphabet for the DFA. This will be independent of the number of states because every character in a string must be processed. What is the space complexity for a DFA? The alphabet is size $|\Sigma|$, the set of states Q is size $|Q|$, and the set of final states F is of size $O(|Q|)$ because all states could be final states. What accounts for most of the space complexity is the transition function. $|Q|$ states with $|\Sigma|$ transitions per state result in space. We add everything together, and keep the largest term, with a result of $O(|Q| * |\Sigma|)$ space complexity. How can we improve the complexity? There are two possibilities for improvement, reducing $|Q|$ and reducing $|\Sigma|$. Reducing $|\Sigma|$ would change the language, so that only leaves $|Q|$.

If we decrease the number of states in our machine, what does this mean? We are removing non-essential states. This may mean that some states are equivalent to each other and can be merged. What does it mean for states to be equivalent? Two states q_i and q_j are equivalent if an arbitrary string run through the DFA beginning in both states produces the same result, either both possibilities accepted or both possibilities rejected.

Let's briefly look at this formally. What logical quantifier corresponds to "arbitrary"? This means "every", so we use the universal quantifier $\forall$. We use the symbol $\equiv$ for "equivalent". Thus we have the following. Remember that "iff" means "if and only if"; "p iff q" means "if p then q and if q then p".

$$q_i \equiv q_j \quad \text{iff} \quad \forall\, w \in \Sigma^*, \quad \delta(q_i, w) \in F \text{ iff } \delta(q_j, w) \in F$$

We can negate this definition (and simplify a little via symmetry) to produce a definition for "not equivalent".

$$q_i \neq q_j \quad \text{iff} \quad \exists\, w \in \Sigma^* \ni \quad \delta(q_i, w) \notin F \text{ and } \delta(q_j, w) \in F$$

A final state can never be equivalent to a nonfinal state. We can let $w = \lambda$, the empty string. We can use the transitions to help us determine further nonequivalences because a symbol corresponds to a string of length 1. A binary tree can be used to help us classify the states. The nodes of the tree will contain sets of states. The root of the tree initially contains all the states. It can have up to two children. One child will contain the final states, and the other will contain the nonfinal states. Suppose all states are final or nonfinal. This means that all the states will be equivalent, and the minimized DFA will only have one state, and loop transitions for each symbol in Σ. Suppose our DFA has at least one

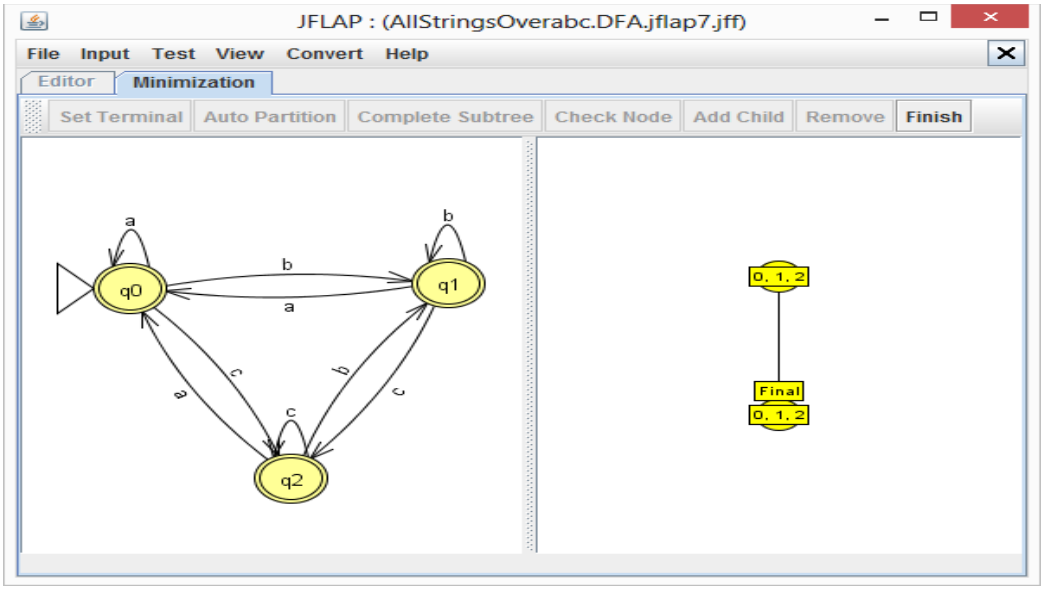
final state and at least one nonfinal state. Then the root will have two children; λ is the string that makes the difference. Now consider the each child. If

Questions to Think About:

1. If all states are final, what is the language accepted by the DFA and the minimized DFA?
2. If all states are nonfinal, what is the language accepted by the DFA and the minimized DFA?
3. Remember that Q represents the set of states in the DFA and that F represents the set of final states in the DFA. How can you write the set of nonfinal states using Q, F, and a set operation?

Try It!

1. If JFLAP is not already active, start JFLAP and click the **Finite Automaton** button.
2. Create an automaton with alphabet {a,b,c} that accepts all possible strings over the alphabet.
3. Save your DFA using a descriptive file name.
4. Verify that your automaton is a DFA by selecting ***Convert > Convert to DFA***. You should get a message saying "Not an NFA". If you don't, fix your automaton and save it.
5. Select ***Convert > Minimize***. Your view should look similar to the following.



6. Click the **Finish** button. Fill in the missing transitions. How many do you need? Why?
7. Save the minimized DFA if you wish.
8. Dismiss the tab for the minimization work.
9. Change all the final states to nonfinal states in your DFA and save your DFA using a descriptive file name.
10. Perform the minimization process again.

Questions to Think About:

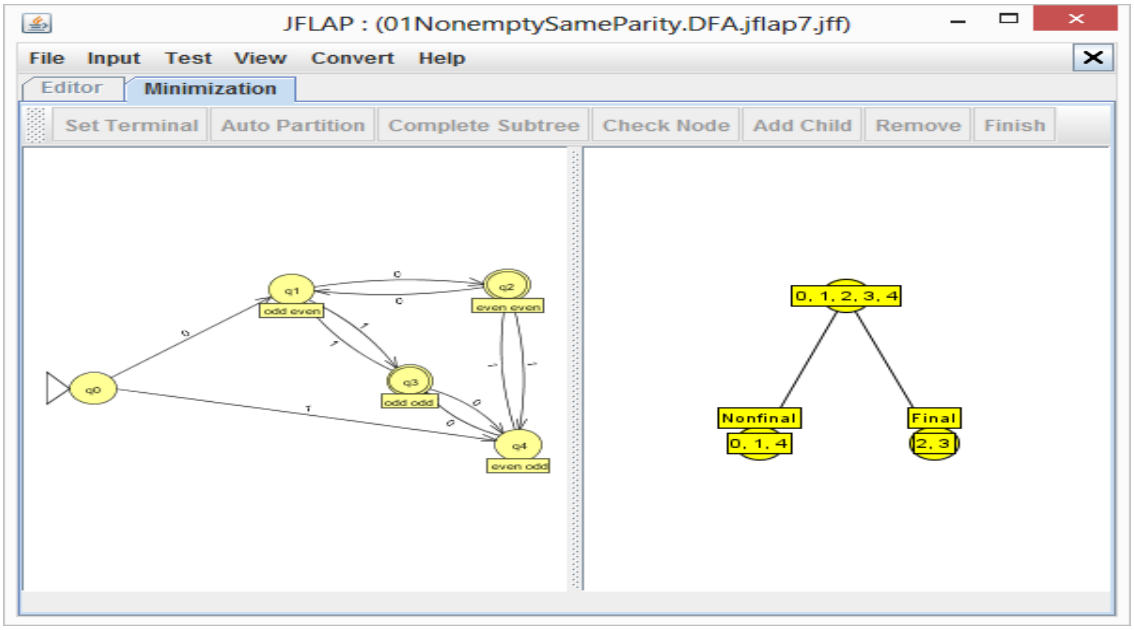
1. What language is accepted by the first minimized DFA? Use formal notation to describe it.
2. What language is accepted by the second minimized DFA? Use formal notation to describe it.

3. What is the relationship between the previous two languages?
4. What happens to the language accepted by a DFA if you switch the roles of every state? That is, every final state becomes a nonfinal state, and vice versa.
5. Can you predict what happens to the language accepted by an NFA if you switch the roles of every state? Why or why not?

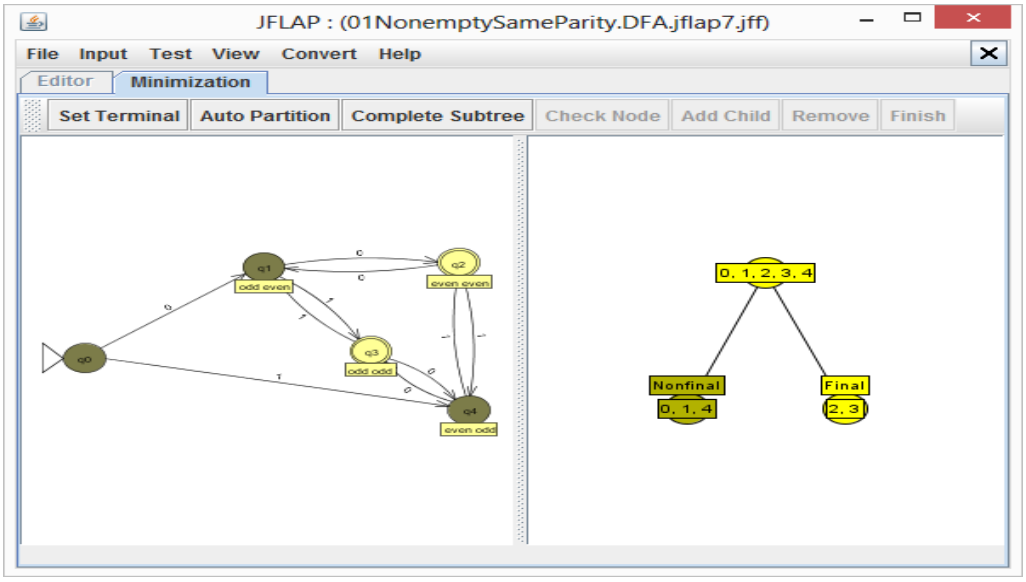
Let's do another example where we don't collapse all the states. We will use the alphabet {0,1} for binary strings, and we will build an automaton to accept nonempty strings that have odd numbers of both 0's and 1's or even numbers of both 0's and 1's. What happens in the initial state? Nothing has been seen yet; this corresponds to λ. This state cannot be a final state. When a 0 (respectively, 1) appears, this changes the current number of 0's (respectively, 1's) from odd to even or vice versa. This suggests the possibility of four additional states of the form xy, where x and y can represent odd or even independently.

Try It!

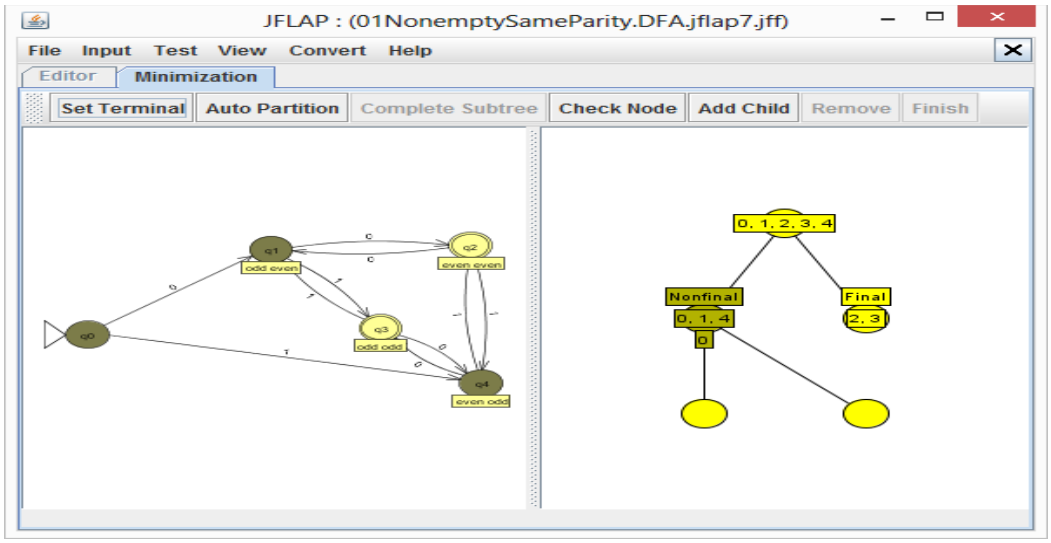
1. If JFLAP is not already active, start JFLAP and click the **Finite Automaton** button.
2. Create an automaton with alphabet {0,1} that accepts only the strings described above.
3. Save your DFA using a descriptive file name.
4. Verify that your automaton is a DFA by selecting ***Convert > Convert to DFA***. You should get a message saying "Not an NFA". If you don't, fix your automaton and save it.
5. Select ***Convert > Minimize***. Your view should look similar to the following.



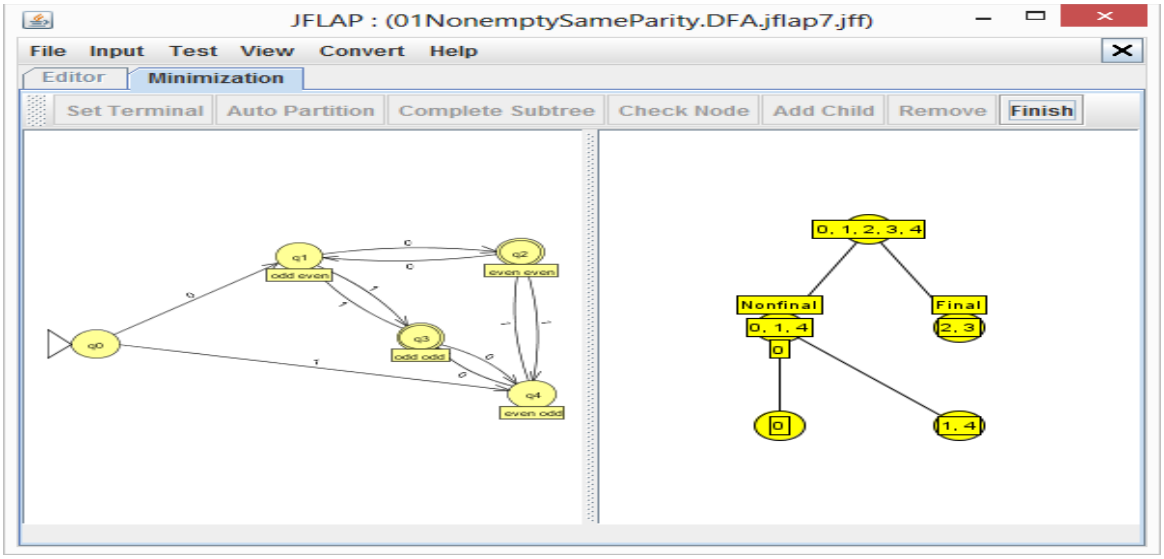
6. Click on the label **0,1,4** in the right-hand pane. Your view should look similar to the following.



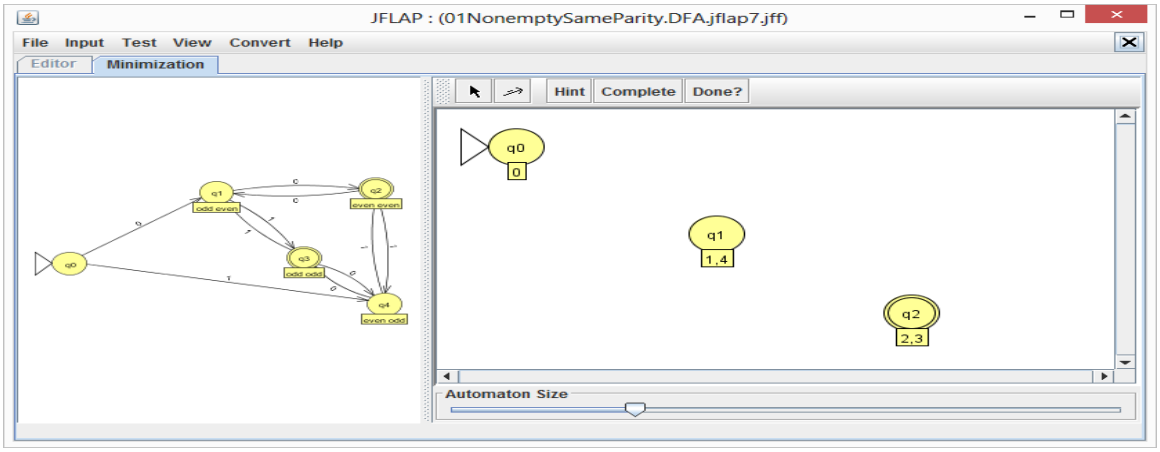
7. Now we need to see which states out of the nonfinal ones can be distinguished. Can a character distinguish among them? If so, which one? Try 0. Click the **Set Terminal** button. Type 0 in the text box, and click the OK button in the pop-up window. Your view should look similar to the following. Notice the **0** label below the **0,1,4** one.



8. Now we need to determine which states go where in the two new tree nodes. You can resize the panes as needed. $\delta(q_0, 0) = q_1$, which is nonfinal. $\delta(q_1, 0) = q_2$, which is final. $\delta(q_4, 0) = q_3$, which is final. This means that q_0 is not equivalent to q_1 or q_4. Click on the **0,1,4** label again and then the **Auto Partition** button. Your view should look similar to the following.



9. Since state q_0 is in a node by itself, we have no further work there. Let us then look at the **1,4** node. Can states q_1 and q_4 be distinguished from each other? $\delta(q_1, 0) = q_2$, which is final. $\delta(q_4, 0) = q_3$, which is final. We don't know whether these states are equivalent, so let's see if 1 can distinguish between the two states. $\delta(q_1, 1) = q_3$, which is final. $\delta(q_4, 1) = q_2$, which is final. We can't determine the answer yet. The equivalence of states q_1 and q_4 depends on the equivalence of states q_2 and q_3.
10. Let us look at the **2,3** node. Can states q_2 and q_3 be distinguished from each other? $\delta(q_2, 0) = q_1$, which is nonfinal. $\delta(q_3, 0) = q_4$, which is nonfinal. As before, we don't know whether these states are equivalent, so let's see if 1 can distinguish between the two states. $\delta(q_2, 1) = q_4$, which is nonfinal. $\delta(q_3, 1) = q_1$, which is nonfinal. We can't determine the answer yet. The equivalence of states q_2 and q_3 depends on the equivalence of states q_1 and q_4.
11. Whoa! We have infinite recursion. If we cannot prove nonequivalence, we must have equivalence. Thus, states q_1 and q_4 are equivalent to each other, and states q_2 and q_3 are equivalent to each other. This is why JFLAP only has the **Finish** button enabled. Click the **Finish** button. Your view should look similar to the following.



12. Our minimized DFA will have 3 states. One state will correspond to state q_0 in the original DFA, one state will correspond to states q_1 and q_4, and one state will correspond to states q_2 and

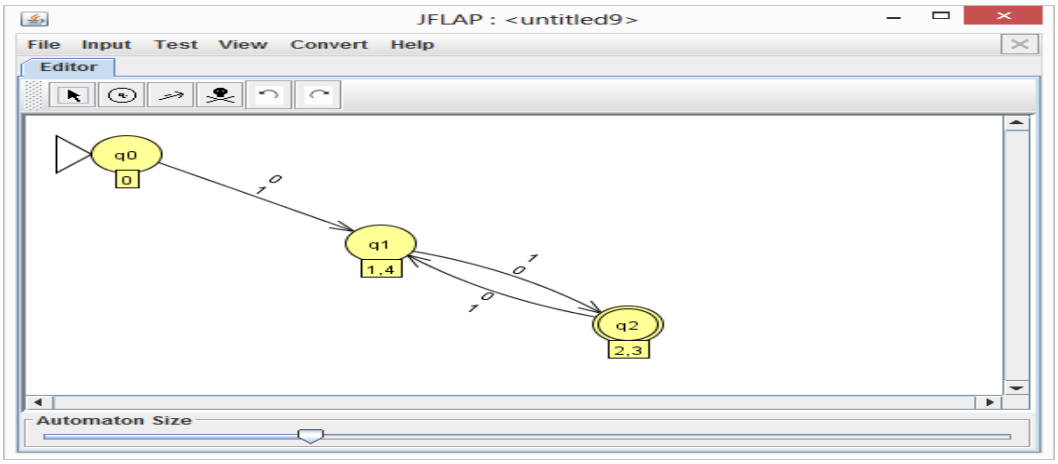
q_3. Now we need to determine the start state, final states, and transitions in the minimized DFA to preserve the original DFA's accepted language.

Questions to Think About:

1. Which state should be the start state in the minimized DFA? Why?
2. Which states should be final states in the minimized DFA? Why?
3. Can a final state ever be equivalent to a non-final state? Why or why not?

Try It!

1. JFLAP has determined the start state and any final state(s) for you already. Because we have a DFA, each state must have a transition with every possible input symbol. How many transitions do you need to add?
2. If the label for a state contains one or more state from the original DFA, all you need to do is pick one of those states, and use the transitions that state in the original DFA as a basis for the transitions in the minimized DFA. If $\delta(q_i, a) = q_j$ in the original DFA, the transition in the minimized DFA will be from the state whose label contains q_i to the state whose label contains q_j. Add these transitions. If you get stuck, click the **Hint** button. Make sure you are complete by clicking the **Complete** button. If you are not, keep going until you get a message saying that all transitions are in place. Click the **Done!** Button. A new window should appear, and your view should be similar to the following.



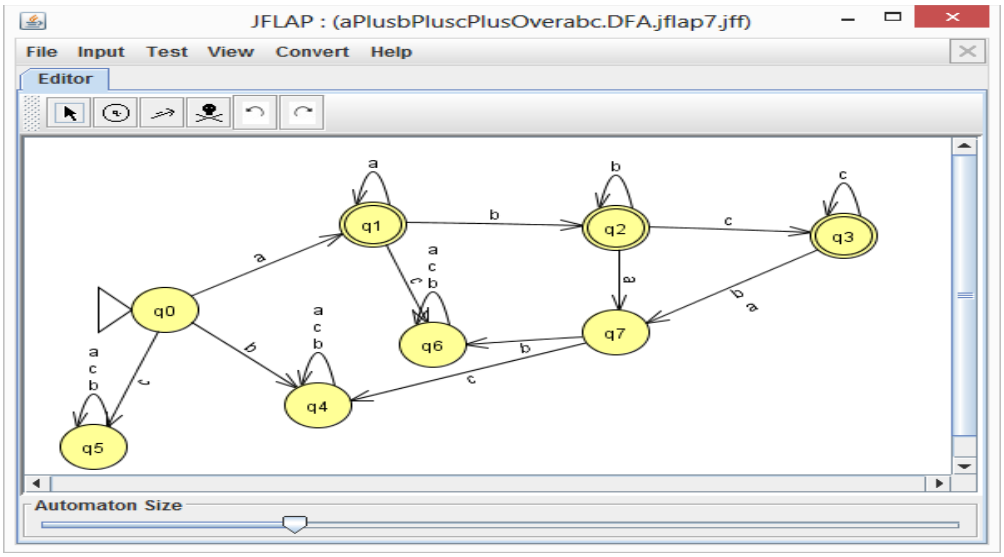
3. Go back to the original DFA, and modify it so that it accepts the empty string because the empty string has 0 (an even number) occurrences of 0's and 0 occurences of 1's. Save your modified DFA using a descriptive file name, and work through the steps to minimize that DFA. How many states does the new minimized DFA have?

Questions to Think About:

1. How many states are in the minimum-state DFA accepting all strings over alphabet Σ?
2. How many states are in the minimum-state DFA accepting no strings over alphabet Σ?
3. How many states are in the minimum-state DFA accepting a single string of length n?

Try It!

1. Minimize the DFA contained in **aPlusbPluscPlusOverabc.DFA.jflap7.jff**. It is pictured below. How many states result? Is the automaton substantially different from the original one?



2. Minimize the DFA contained in **UpTo3asOverab.DFA.jflap7.jff**. It is pictured below. How many states result? Is the automaton substantially different from the original one?