Chapter 2

NFA to DFA to Minimal DFA

This chapter shows how each NFA can be converted into an equivalent DFA, and how each DFA can be reduced to a DFA with a minimum number of states. Although an NFA might be easier to construct than a DFA, the NFA is usually not efficient to run, as an input string may follow several paths. Converting an NFA into an equivalent DFA ensures that each input string follows only one path. The NFA to DFA algorithm in JFLAP combines similar states reached in the NFA into one state in the DFA. The DFA to minimum state DFA algorithm in JFLAP determines which states in the DFA have similar behavior with respect to incoming and outgoing transitions and combines these states, resulting in a minimal state DFA.

2.1 NFA to DFA

In this section we use JFLAP to show how to convert an NFA into an equivalent DFA. The idea in the conversion is to create states in the DFA that represent multiple states in the NFA. The start state in the DFA represents the start state in the NFA and any states reachable from it on $\lambda$. For each new state in the DFA and each letter of the alphabet, one determines all the reachable states from the corresponding NFA states and combines them into a new state for the DFA. This state in the DFA will have a label that will contain the state numbers from the NFA that would be reachable in taking the same path.

2.1.1 Idea for the Conversion

Load the NFA in file ex2.1a as shown in Figure 2.1. We will refer to this example in explaining the steps in converting this NFA to a DFA.

First examine the choices that occur when the NFA processes input. Select **Input : Step with Closure** and enter the input string "aabbbaa" and press return. Clicking **Step** once shows that processing $a$ can result in arriving in both states $q_0$ and $q_1$. Clicking **Step** six more times shows
that there are always three configurations (one of which is rejected), and results in two paths of acceptance in states $q_2$ and $q_3$.

The states in the constructed DFA will represent combined states from the NFA. For example, processing an $a$ resulted in either state $q_0$ or $q_1$. The DFA would have a state that represents both of these NFA states. Processing $aabbbaa$ resulted in reaching final states $q_2$ and $q_3$. The DFA would have a state that represented both of these NFA states. Dismiss the tab for the step run (select File : Dismiss Tab) to go back to the NFA editor.

### 2.1.2 Conversion Example

Now we will convert the NFA to a DFA (select Convert : Convert to DFA), showing the NFA on the left and the first state of the DFA on the right. The initial state in the DFA is named $q_0$ and has the label 0, meaning it represents the $q_0$ state from the NFA.

**Tip**  
The NFA may be tiny. Adjust the size in one of two ways: either resize the window, or drag the vertical bar between the NFA and the DFA to the right. In addition, the states in the DFA can be dragged closer to each other, resulting in larger states.

We will now add the state that is reachable from $q_0$ on the substring $a$. Select the Expand Group on Terminal tool *. Click and hold the mouse on state $q_0$, drag the cursor to where you want the next state, and release it. When prompted by **Expand on what terminal?**, enter "a" and press return. When prompted by **Which group of NFA states will that go to on a?**, enter the numbers of the states that are reachable from $q_0$ on an $a$. In this case enter "0,1". (These NFA states could also be entered with a blank separator and with or without the $q$, such as “$q0,q1$”.) The new state $q_1$ appears in Figure 2.2.

Use the Attribute Editor tool you learned about in Chapter 1 to move states around if you don’t like their placement.
2.1. NFA TO DFA

Try expanding DFA state $q_0$ on the terminal $b$. Since there are no paths from NFA state $q_0$ on a $b$, a warning message is displayed.

Next expand the DFA state $q_1$ on the terminal $a$. Note that DFA state $q_1$ represents both states $q_0$ and $q_1$ from the NFA. In the NFA, state $q_0$ on an $a$ reaches states $q_0$ and $q_2$, and state $q_1$ on an $a$ reaches no state. The union of these results $(0, 1)$ are the states reachable by DFA state $q_1$, which happens to be the DFA state $q_1$. Upon the completion of the expansion a transition loop labeled $a$ is added to DFA state $q_1$. Now expand DFA state $q_1$ on $b$. The result of these two expansions is shown in Figure 2.3. Why is DFA state $q_2$ a final state? If a DFA state represents any NFA state that is a final state, then the substring processed is accepted on some path, and thus the DFA state also must be a final state. NFA state $q_2$ is a final state, so DFA state $q_2$ (representing NFA states $q_1$ and $q_2$) is a final state.

Expand DFA state $q_2$ on $a$. This state is represented by NFA states $q_1$ and $q_2$. NFA state $q_1$ does not have an $a$ transition. NFA state $q_2$ on an $a$ reaches state $q_3$ and due to the $\lambda$-transition also reaches state $q_2$.

**Note** In using the Expand Group Terminal tool, if the destination state already exists, then drag to the existing state and you will be prompted only for the terminal to expand. Thus, to add a loop transition, just click on the state.

Expand DFA state $q_2$ on $b$ by clicking on state $q_2$. You are prompted for the $b$, but not the states reachable, as that is interpreted as your selected state (itself in this case). The resulting DFA is shown in Figure 2.4.

There is another way to expand a state—the State Expander tool "*". When one selects this tool and clicks on a state, all arcs out of the state are automatically expanded. In Figure 2.5 state $q_3$ was selected and expanded on both $a$ and $b$, resulting in a new state $q_4$.
Is the DFA complete? Select the Done? button. If the DFA is not complete, a message indicating items missing is displayed. At this time, one transition is missing.

Expand DFA state \( q_4 \) on \( b \) by going back to the Expand Group on Terminal tool. Note that \( q_4 \) on \( b \) goes to the existing DFA state \( q_2 \). Click on state \( q_4 \), drag to state \( q_2 \), and release. You will be prompted for the terminal only.

Is the DFA complete? Select the Done? button. The DFA is complete and is exported to a new window. The complete DFA is shown in Figure 2.6. Alternatively, the Complete button can be selected at any time during the construction process and the complete DFA will be shown.

The constructed DFA should be equivalent to the NFA. To test this, in the DFA window select Test: Compare Equivalence. Select file ex2.1a, the name of the NFA, and then press return. The two machines are equivalent.
2.2. DFA TO MINIMAL DFA

2.1.3 Algorithm to Convert NFA $M$ to DFA $M'$

We describe the algorithm to convert an NFA $M$ to a DFA $M'$. We first define the closure of a set of states to be those states unioned with all states reachable from these states on a $\lambda$-transition.

1. The initial state in $M'$ is the closure of the initial state from $M$.

2. For each state $q'$ in $M'$ and each terminal $x$ do the following:

   (a) States $q$ and $r$ are states in $M$. For each state $q$ that is in state $q'$, if $q$ on an $x$ reaches state $r$ on an $x$, then place state $r$ in new state $p'$.

   (b) $p' = \text{closure}(p')$

   (c) If another state is equivalent to state $p'$ (represents the same states from $M$), then set $p'$ to the state already existing.

   (d) Add the transition to $M'$: $q'$ to $p'$ on an $x$.

3. Each state $q'$ in $M'$ is a final state if it contains a final state from $M$.

2.2 DFA to Minimal DFA

In this section we show how to convert a DFA to a minimal state DFA. Consider two states $p$ and $q$ from a DFA, each processing a string starting from their state. If there is at least one string $w$ such that states $p$ and $q$ process this string and one accepts $w$ and one rejects $w$, then these states are distinguishable and cannot be combined. Otherwise, states $p$ and $q$ “act the same way,” meaning that they are indistinguishable and can be combined.

2.2.1 Idea for the Conversion

Load the DFA in Figure 2.7 (file ex2.2a). We will refer to this example to explain the steps to convert this DFA to a minimal state DFA.

We would like to examine pairs of states to see if they are distinguishable or not. To do this we will need two separate windows for this DFA. JFLAP lets you open only one copy of each file, so if you try to open the same file again, JFLAP will show just the one window. Instead we will make a duplicate copy of this file by saving it with a different name (select File : Save as and type the filename “ex2.2a-dup”). The current window is now associated with the duplicate file. Load the original file ex2.2a again and it will appear in a separate window (possibly on top of the first window). Move the two windows so you can see both of them.
We will examine the two states $q_0$ and $q_1$ to see if they are distinguishable. In one of the windows, change the start state to $q_1$. Examine the two DFA. Are there any strings that one DFA accepts and the other DFA rejects?

We will examine several strings to see if there is any difference in acceptance and rejection. In both DFA windows, select **Input : Multiple Run**. In both windows, enter the following strings and any additional ones you’d like to try: “a”, “aab”, “aaaab”, “baa”, “baaa”, and “bba”. Select **Run Inputs** and examine the results. Do the strings have the same result in both DFAs? There is at least one string in which the result is **Accept** for one DFA, and **Reject** in the other DFA. Thus the two states $q_0$ and $q_1$ are distinguishable and cannot be combined.

Now we will examine the two states $q_2$ and $q_5$ to see if they are distinguishable. Dismiss the tab in both windows to go back to the DFA window. In one window change the start state to $q_2$, and in the other window change the start state to $q_5$. Select **Input : Multiple Run** again. Notice that the strings from the last run still appear in the window. Select **Run Inputs** to try these same strings. Type in additional strings and try them as well. Are these states distinguishable or indistinguishable? They are distinguishable if there is one string that accepts in one and does not accept in the other. All strings must be tested to determine if the states are indistinguishable. Clearly it is impossible to test all strings, so a reasonable test set should be created.
2.2.2 Conversion Example

We go through an example of converting a DFA to a minimum state DFA. Remove the previous windows (without saving them) and load the file ex2.2a again, which should have the start state $q_0$. Select **Convert : Minimize DFA**. The window splits into two showing the DFA on the left and a tree of states on the right.

We assume that all states are indistinguishable to start with. The root of the tree contains all states. Each time we determine a distinction between states, we split a node in the tree to show this distinction. We continue to split nodes until there are no more splits possible. Each leaf in the final tree represents a group of states that are indistinguishable.

The first step in distinguishing states is to note that a final and a nonfinal state are different. The former accepts $\lambda$ and the other does not. Thus the tree has already split the set of states into two groups of nonfinal and final states as shown in Figure 2.8.

For additional splits, a terminal will be selected that distinguishes the states in the node. If some of the states in a leaf node on that terminal go to states in one leaf node and other states on that same terminal go to states that are in another leaf node, then the node should be split into two groups of states (i.e., two new leaf nodes).

Let's first examine the leaf node of the nonfinal states (0, 1, 2, 4, 5, 7). What happens for each of these states if they process a $b$? State $q_0$ on a $b$ goes to state $q_2$, state $q_1$ on a $b$ goes to state $q_0$, and so on. Each of these states on a $b$ goes to a state already in this node. Thus, $b$ does not distinguish these states. In JFLAP, click on the tree node containing the nonfinal states. (Click on the circle, not the label or the word Nonfinal.) The states in this node are highlighted in the DFA. Try to split this node on the terminal $b$. Select **Set Terminal** and enter $b$. A message appears informing you that $b$ does not distinguish these states.

Again select **Set Terminal** and enter the terminal $a$. Since $a$ does distinguish these states, the node is split, resulting in two new leaf nodes. The set of states from the split node must be entered into the new leaf nodes, into groups that are indistinguishable. A state number can be entered by
first selecting the leaf node it will be assigned to, and then clicking on the corresponding state in the DFA. Click on the left leaf node and then click on state $q_0$ in the DFA. The state number 0 should appear in the leaf node, as shown in Figure 2.9.

State $q_0$ on an $a$ goes to state $q_5$, which is in the node we are splitting. Note that states $q_1$, $q_4$, and $q_7$ on an $a$ also go to a state in the node we are splitting. Add all of them to the same new leaf node as 0 by clicking on these states in the DFA. The remaining states, $q_2$ and $q_6$ on an $a$, go to a final state, thus distinguishing them. Click on the right new leaf node, and then click on states $q_2$ and $q_6$ to enter them into this node, resulting in the tree shown in Figure 2.10. To see if we have done this correctly, click on Check Node. Figure 2.10 shows the resulting tree after splitting this node on $a$. 
We must continually try to split nodes on terminals until there is no further splitting. Each time we split a node, we have created new groups that might now allow another group to be split that could not be split before.

Next we try to split the leaf node with states 0, 1, 4, and 7. Which terminal do you try? In this case either a or b will cause a split. We will try a. Select Set Terminal and enter a. Enter the split groups. State $q_0$ on an a goes to state $q_5$, which is in leaf node group 2, 5, and states $q_1$, $q_4$, and $q_7$ on an a go to states in the leaf node we are splitting. Let's enter these states a different way. Select Auto Partition and the states will automatically be entered in as shown in Figure 2.11.

When the tree is complete (as it is now, convince yourself that none of the leaf nodes can be further split), then the only option visible is Finish. Select Finish and the right side of the window is replaced by the new states for the minimum DFA. There is one state for each leaf node from the tree (note the labels on the states correspond to the states from the original DFA), as shown in Figure 2.12. You may want to rearrange the states using the Attribute Editor.

Now add in the missing arcs in the new DFA using the Transition Creator tool. In the original DFA there is an a from state $q_0$ to state $q_5$, so in the new DFA a transition is added.
from state \( q_1 \) (representing the old state \( q_0 \)) to state \( q_2 \) (representing the old state \( q_5 \)). Selecting **Hint** will add one transition for you and selecting **Complete** will complete the DFA, as shown in Figure 2.13. Selecting **Done** will export the new DFA to its own window.

The minimum state DFA should be equivalent to the original DFA. Test this using the **Test : Compare Equivalence** option.

**Note**

When you select a node and select **Set Terminal**, the node you select is split and two children appear. It is possible that the node to be split might need more children; that is, there may be 3 or more distinguished groups split on this terminal. In that case, you must add the additional leaf nodes by selecting the **Add Child** option for each additional child desired.

### 2.2.3 Algorithm

We describe the algorithm to convert a DFA \( M \) to a minimal state DFA \( M' \).

1. Create the tree of distinguished states as follows:
   
   (a) The root of the tree contains all states from \( M \)
   
   (b) If there are both final and nonfinal states in \( M \), create two children of the root—one containing all the nonfinal states from \( M \) and one containing all the final states from \( M \).
   
   (c) For each leaf node \( N \) and terminal \( x \), do the following until no node can be split:
      
      i. If states in \( N \) on \( x \) go to states in \( k \) different leaf nodes, \( k > 1 \), then create \( k \) children for node \( N \) and spread the states from \( N \) into the \( k \) nodes in indistinguishable groups.

2. Create the new DFA as follows:
   
   (a) Each leaf node in the tree represents a state in the DFA \( M' \) with a label equal to the states from \( M \) in the node. The start state in \( M' \) is the state that contains the start
state from $M$ in its label. A state in $M'$ is a final state if it contains a final state from $M$ in its label.

(b) For each arc in $M$ from states $p$ to $q$, add an arc in $M'$ from the state that has $p$ in its label to the state that has $q$ in its label. Do not add any duplicate arcs.

2.3 Exercises

1. Convert the NFAs in the given files into DFAs.

   (a) ex2-nfa2dfa-a
   (b) ex2-nfa2dfa-b
   (c) ex2-nfa2dfa-c
   (d) ex2-nfa2dfa-d
   (e) ex2-nfa2dfa-e
   (f) ex2-nfa2dfa-f

2. Consider the language $L = \{w \in \Sigma^* | w$ does not have the substring $aab\}, \Sigma = \{a, b\}$. Load the DFA in file ex2.3a shown in Figure 2.14. This DFA recognizes $L$.

Also load the file ex2.3b. It is the NFA shown in Figure 2.15 that attempts to recognize $L$, but fails.

Give an input string that shows why this NFA is not equivalent to this DFA.
Figure 2.15: NFA from file ex2.3b.

3. Consider the DFA in file ex2-dfa2nfa. This DFA was converted from an NFA and the labels show the states of the original NFA. That NFA did not have any \( \lambda \)-transitions. Create the original NFA.

4. Convert the DFAs in the given files into minimal state DFAs.

   (a) ex2-dfa2mindfa-a
   (b) ex2-dfa2mindfa-b
   (c) ex2-dfa2mindfa-c
   (d) ex2-dfa2mindfa-d
   (e) ex2-dfa2mindfa-e

5. Consider the DFA in file ex2.3c. Explain why it is not a minimal DFA.

6. Consider the DFA in file ex2-dfa2mindfa-test. States \( q_2 \) and \( q_8 \) are distinguishable. Make a copy of this file and make \( q_2 \) the start state in one and \( q_8 \) the start state in the other. Give five input strings that are accepted by both of these DFAs and one input string that distinguishes the two DFAs, thus distinguishing states \( q_2 \) and \( q_8 \) in the original DFA. You can confirm your answer by running the DFAs on the input strings.