DFA Minimization

The preceding sections established that the family of languages accepted by DFAs is the same as that accepted by NFAs and NFA-$\lambda$s. The flexibility of nondeterminism and $\lambda$-transitions aid in the design of machines to accept complex languages. The nondeterministic machine can then be transformed into an equivalent deterministic machine using Algorithm 5.6.3. The resulting DFA, however, may not be the minimal DFA that accepts the language. This section presents a reduction algorithm that produces the minimal state DFA accepting the language $L$ from any DFA that accepts $L$. To accomplish the reduction, the notion of equivalent states in a DFA is introduced.

**Definition 5.7.1**

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. States $q_i$ and $q_j$ are equivalent if $\delta(q_i, u) \in F$ if, and only if, $\delta(q_j, u) \in F$ for every $u \in \Sigma^*$. Two states that are equivalent are called indistinguishable. The binary relation over $Q$ defined by indistinguishability of states is an equivalence relation; that is, the relation is reflexive, symmetric, and transitive. Two states that are not equivalent are said to be distinguishable. States $q_i$ and $q_j$ are distinguishable if there is a string $u$ such that $\delta(q_i, u) \in F$ and $\delta(q_j, u) \notin F$, or vice versa.

The motivation behind this definition of equivalence is illustrated by the following states and transitions:

```
\[ \begin{array}{c}
q_0 \xrightarrow{a,b} q_i \\
q_i \xrightarrow{a} q_k \\
q_k \xrightarrow{b} q_n \\
q_n \xrightarrow{a,b} q_m \\
\end{array} \]
```

The unlabeled dotted lines entering $q_i$ and $q_j$ indicate that the method of reaching state is irrelevant; equivalence depends only upon computations from the state. The states $q_i$ and $q_j$ are equivalent since the computation with any string beginning with $b$ from state halts in an accepting state and all other computations halt in the nonaccepting state.

States $q_m$ and $q_n$ are also equivalent; all computations beginning in these states end in accepting states.

The intuition behind the transformation is that equivalent states may be merged. Applying this to the preceding example yields:

To reduce the number of states, we must determine $q_j$, $i < j$, has associated class $i$ determined that the index $[i, j]$ is in the class $q_m$ and $q_n$.

The algorithm determines the class of each state and examines each nonaccepting state $q_j$ as distinguishable or distinguishable.

**Algorithm 5.7.2**

Determine of Equivalence

**Initialization**

For every pair of states $q_i, q_j$:

1. $D[i, j] = S[i, j] = \emptyset$

**End for**

For every pair $i$, if $i$ is an accepting state:

1. For every pair $i, j$:
   1. If there is an edge from $i$ to $j$:
      1. $D[m, n] = \emptyset$ otherwise:
      2. $D[m, n] = \emptyset$

For $m$ and $n$ in $D[i, j]$

 insertion of all states in $D[i, j]$ into the diagram.
To reduce the size of a DFA $M$ by merging states, a procedure for identifying equivalent states must be developed. In the algorithm to accomplish this, each pair of states $q_i$ and $q_j$, $i < j$, has associated with it values $D[i, j]$ and $S[i, j]$. $D[i, j]$ is set to 1 when it is determined that the states $q_i$ and $q_j$ are distinguishable. $S[m, n]$ contains a set of indices. Index $[i, j]$ is in the set $S[m, n]$ if the distinguishability of $q_i$ and $q_j$ follows from that of $q_m$ and $q_n$.

The algorithm begins by marking each pair of states $q_i$ and $q_j$ as distinguishable if one is accepting and the other is rejecting. The remainder of the algorithm systematically examines each nonmarked pair of states. When two states are shown to be distinguishable, a call to a recursive routine $DIST$ sets $D[i, j]$ to 1. The call $DIST(i, j)$ not only marks $q_i$ and $q_j$ as distinguishable, it also marks each pair of states $q_m$ and $q_n$ for which $[m, n] \in S[i, j]$ as distinguishable through a call to $DIST(m, n)$.

---

### Algorithm 5.7.2

#### Determination of Equivalent States of DFA

**Input:** DFA $M = (Q, \Sigma, \delta, q_0, F)$

1. **Initialization**
   - for every pair of states $q_i$ and $q_j$, $i < j$, do
     - $D[i, j] := 0$
     - $S[i, j] := \emptyset$
   - for every pair $i, j, i < j$, if one of $q_i$ or $q_j$ is an accepting state and the other is not an accepting state, then set $D[i, j] := 1$

2. **for every pair $i, j, i < j$, with $D[i, j] = 0$, do**
   1. if there exists an $a \in \Sigma$ such that $\delta(q_i, a) = q_m$ and $\delta(q_j, a) = q_n$ and $D[m, n] = 1$ or $D[n, m] = 1$, then $DIST(i, j)$
   2. else for each $a \in \Sigma$, do: Let $\delta(q_i, a) = q_m$ and $\delta(q_j, a) = q_n$
      - if $m < n$ and $[i, j] \neq [m, n]$, then add $[i, j]$ to $S[m, n]$
      - else if $m > n$ and $[i, j] \neq [n, m]$, then add $[i, j]$ to $S[n, m]$
   end for

3. **end if**

   $DIST(i, j)$;

   **begin**
   - for all $[m, n] \in S[i, j]$, $DIST(m, n)$
   **end**
The motivation behind the identification of distinguishable states is illustrated by the relationships in the diagram:

\[ q_i \xrightarrow{a} q_m \]
\[ q_j \xrightarrow{a} q_n \]

If \( q_m \) and \( q_n \) are already marked as distinguishable when \( q_i \) and \( q_j \) are examined in step 3, then \( D[i, j] \) is set to 1 to indicate the distinguishability of \( q_i \) and \( q_j \). If the status of \( q_m \) and \( q_n \) is not known when \( q_i \) and \( q_j \) are examined, then a later determination that \( q_m \) and \( q_n \) are distinguishable also provides the answer for \( q_i \) and \( q_j \). The role of the array \( S \) is to record this information: \([i, j] \in S[n, n] \) indicates that the distinguishability of \( q_m \) and \( q_n \) is sufficient to establish the distinguishability of \( q_i \) and \( q_j \). These ideas are formalized in the proof of Theorem 5.7.3.

**Theorem 5.7.3**
States \( q_i \) and \( q_j \) are distinguishable if, and only if, \( D[i, j] = 1 \) at the termination of Algorithm 5.7.2.

**Proof.** First we show that every pair of states \( q_i \) and \( q_j \), for which \( D[i, j] = 1 \) is distinguishable. If \( D[i, j] \) is assigned 1 in the step 2, then \( q_i \) and \( q_j \) are distinguishable by the null string. Step 3.1 marks \( q_i \) and \( q_j \) as distinguishable only if \( \delta(q_i, a) = q_m \) and \( \delta(q_j, a) = q_n \) for some input \( a \) when states \( q_m \) and \( q_n \) have already been determined to be distinguishable by the algorithm. Let \( u \) be a string that exhibits the distinguishability of \( q_m \) and \( q_n \). Then \( au \) exhibits the distinguishability of \( q_i \) and \( q_j \).

To complete the proof, it is necessary to show that every pair of distinguishable states is designated as such. The proof is by induction on the length of the shortest string that demonstrates the distinguishability of a pair of states. The basis consists of all pairs of states \( q_i, q_j \) that are distinguishable by a string of length 0. That is, the computations \( \delta(q_i, \lambda) = q_m \) and \( \delta(q_j, \lambda) = q_n \) distinguish \( q_i \) from \( q_j \). In this case, exactly one of \( q_i \) or \( q_j \) is accepting and the position \( D[i, j] \) is set to 1 in step 2.

Now assume that every pair of states distinguishable by a string of length \( k \) or less is marked by the algorithm. Let \( q_i \) and \( q_j \) be states for which the shortest distinguishing string is of length \( k + 1 \). Then \( u \) can be written \( au \) and the computations with input \( u \) have the form \( \delta(q_i, u) = \delta(q_i, au) = \delta(q_m, v) = q_i \) and \( \delta(q_j, u) = \delta(q_j, au) = \delta(q_n, v) = q_j \). Exactly one of \( q_i \) and \( q_j \) is accepting since the preceding computations distinguish \( q_i \) from \( q_j \). The same computations exhibit the distinguishability of \( q_m \) from \( q_n \) by a string of length \( k + 1 \). By induction, we know that the algorithm will set \( D[m, n] \) to 1.

If \( D[m, n] \) is marked before the states \( q_i \) and \( q_j \) are examined in step 3, then \( D[i, j] \) is set to 1 by the call \( DIST(i, j) \). If \( q_i \) and \( q_j \) are examined in the loop in step 3, \( D[m, n] \neq 1 \) at that time, then \([i, j] \) is added to the set \( S[m, n] \). By the inductive hypothesis, \( D[m, n] \) will eventually be set to 1. \( D[i, j] \) will also be set to 1 at this time by the call from \( DIST(m, n) \) since \([i, j] \) is in \( S[m, n] \).
A new DFA $M'$ can be built from the original DFA $M = (Q, \Sigma, \delta, q_0, F)$ and the distinguishability relation. The states of $M'$ are the equivalence classes consisting of distinguishable states of $M$. The start state is $[q_0]$, and $[q_i]$ is a final state if $q_i \in F$.

The transition function $\delta'$ of $M'$ is defined by $\delta'([q_i], a) = [\delta(q_i, a)]$. In Exercise 44, $\delta'$ is shown to be well defined. $L(M')$ consists of all strings whose computations have the form $\delta([q_0], \lambda) = [\delta(q_i, \lambda)]$ with $q_i \in F$. These are precisely the strings accepted by $M$. If $M'$ has states that are unreachable by computations from $[q_0]$, these states and all associated arcs are deleted.

**Example 5.7.1**

The minimization process is exhibited using the DFA $M$

![DFA Diagram]

that accepts the language $(a \cup b)(a \cup b^*)$.

In step 2, $D[0, 1], D[0, 2], D[0, 3], D[0, 4], D[0, 5], D[0, 6], D[1, 7], D[2, 7], D[3, 7], D[4, 7], D[5, 7], D[6, 7]$ are set to 1. Each index not marked in step 2 is examined in step 3. The table shows the action taken for each such index.

<table>
<thead>
<tr>
<th>Index</th>
<th>Action</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 7]</td>
<td>$D[0, 7] = 1$</td>
<td>Distinguished by $a$</td>
</tr>
<tr>
<td>[1, 2]</td>
<td>$D[1, 2] = 1$</td>
<td>Distinguished by $a$</td>
</tr>
<tr>
<td>[1, 3]</td>
<td>$D[1, 3] = 1$</td>
<td>Distinguished by $a$</td>
</tr>
<tr>
<td>[1, 4]</td>
<td>$S[2, 5] = {1, 4}$</td>
<td>Distinguished by $a$</td>
</tr>
<tr>
<td>[1, 4]</td>
<td>$S[3, 6] = {1, 4}$</td>
<td>Distinguished by $a$</td>
</tr>
<tr>
<td>[1, 5]</td>
<td>$D[1, 5] = 1$</td>
<td>Distinguished by $a$</td>
</tr>
<tr>
<td>[1, 6]</td>
<td>$D[1, 6] = 1$</td>
<td>Distinguished by $a$</td>
</tr>
<tr>
<td>[2, 3]</td>
<td>$D[2, 3] = 1$</td>
<td>Distinguished by $b$</td>
</tr>
</tbody>
</table>

(Continued)
After each pair of indices is examined, \([1, 4]\), \([2, 5]\), and \([3, 6]\) are left as equivalent pairs of states. Merging these states produces the minimal state DFA \(M'\) that accepts \((a \cup b)(a \cup b^*)\).

Example 5.7.2

Minimizing the DFA \(M\) illustrates the recursive marking of states by the call to \(D\). The language of \(M\) is \(a(a \cup b)^* \cup ba(a \cup b)^* \cup bba(a \cup b)^*\).

The comparison of accepting states to nonaccepting states assigns 1 to \(D[0, 6], D[1, 4], D[1, 5], D[1, 6], D[2, 4], D[2, 5], D[2, 6], D[3, 4], D[3, 5]\). Tracing the algorithm produces
5.7 DFA Minimization

<table>
<thead>
<tr>
<th>Index</th>
<th>Action</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 1]</td>
<td>( S[4, 5] = {0, 1} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( S[1, 2] = {0, 1} )</td>
<td></td>
</tr>
<tr>
<td>[0, 2]</td>
<td>( S[4, 6] = {0, 2} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( S[1, 3] = {0, 2} )</td>
<td></td>
</tr>
<tr>
<td>[0, 3]</td>
<td>( D[0, 3] = 1 )</td>
<td>Distinguished by ( a )</td>
</tr>
<tr>
<td>[1, 2]</td>
<td>( S[5, 6] = {1, 2} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( S[2, 3] = {1, 2} )</td>
<td></td>
</tr>
<tr>
<td>[1, 3]</td>
<td>( D[1, 3] = 1 )</td>
<td>Distinguished by ( a )</td>
</tr>
<tr>
<td></td>
<td>( D[0, 2] = 1 )</td>
<td>Call to ( DIST(1, 3) )</td>
</tr>
<tr>
<td>[2, 3]</td>
<td>( D[2, 3] = 1 )</td>
<td>Distinguished by ( a )</td>
</tr>
<tr>
<td></td>
<td>( D[1, 2] = 1 )</td>
<td>Call to ( DIST(1, 2) )</td>
</tr>
<tr>
<td></td>
<td>( D[0, 1] = 1 )</td>
<td>Call to ( DIST(0, 1) )</td>
</tr>
</tbody>
</table>

Merging equivalent states \( q_4 \), \( q_5 \), and \( q_6 \) yields

\[
M: \begin{array}{c}
\begin{tikzpicture}
\node (q0) at (0, 0) {$q_0$};
\node (q1) at (1.5, 0) {$q_1$};
\node (q2) at (3, 0) {$q_2$};
\node (q3) at (4.5, 0) {$q_3$};
\node (q4) at (0, -1.5) {$q_4, q_5, q_6$};
\node (q5) at (0, -3) {$\bullet$};

\draw[->] (q0) -- node[above] {$b$} (q1);
\draw[->] (q0) -- node[below] {$a$} (q4);
\draw[->] (q1) -- node[above] {$b$} (q2);
\draw[->] (q1) -- node[below] {$a$} (q4);
\draw[->] (q2) -- node[above] {$b$} (q3);
\draw[->] (q2) -- node[below] {$a$} (q4);
\draw[->] (q3) -- node[above] {$\Leftrightarrow$} (q0);
\end{tikzpicture}
\end{array}
\]

The minimization algorithm completes the sequence of algorithms required for the construction of optimal DFAs. Nondeterminism and \( \lambda \)-transitions provide tools for designing finite automata to match complicated patterns or to accept complex languages. Algorithm 5.6.3 can then be used to transform the nondeterministic machine into a DFA, which may not be minimal. Algorithm 5.7.2 completes the process by producing the minimal state DFA.

For the moment, we have presented an algorithm for DFA reduction but have not established that it produces the minimal DFA. In Section 6.7 we prove the Myhill-Nerode Theorem, which characterizes the language accepted by a finite automaton in terms of equivalence classes of strings. This characterization will then be used to prove that the machine \( M' \) produced by Algorithm 5.7.2 is the unique minimal state DFA that accepts \( L \).