LECTURE 26

13 NFA and DFA Equivalence

It appears that the use nondeterminism makes finite automata more powerful. We will show that this is not so, i.e. the languages accepted by NFAs are exactly the same as those accepted by DFAs. Given any NFA, it can be transformed into an equivalent DFA that accepts the same language.

A clue as to how to do this can be seen from the example that we did when tracing several computation paths. We are given the following NFA.

If we trace all possible computation paths on input \( ababb \), we get the picture shown in figure 13(a), but the computation can be described by “collapsing” sets of states into single states, as shown in figure 13(b).

We can see the same computation represented in (b), but this time the states that appear at a given level of the tree have been grouped together into one single set.

The instantaneous configurations that appear in the second trace are the same that we would obtain if we followed a DFA (not an NFA) with states labeled \( \{0\} \), \( \{0, 1\} \), \( \{2\} \), and \( \{1, 2\} \). Some of the transitions of this new DFA can be seen from the trace, such as \( \delta(\{0\}, a) = \{0, 1\} \).

We can write down all possible transitions starting from the start state, and create the transition table for an equivalent DFA.

The following table is created by starting with the start state of the new machine, which is \( \{0\} \), and adding all the transitions for each of the symbols from the alphabet. If a new state (that has not yet been seen) appears, it is added to the table.

\[
\begin{array}{c|cc}
\delta & a & b \\
\hline
\{0\} & \{0, 1\} & \emptyset \\
\{0, 1\} & \{0, 1\} & \emptyset \\
\end{array}
\]

The state \( \{0, 1\} \), appeared in the transition out of \( \{0\} \), so we add one new row labeled \( \{0, 1\} \). Since there are no transitions from state 0 labeled \( b \), we say that the new state is the empty set. We now compute the transition for the next row of the table, i.e. state \( \{0, 1\} \).
This time a new state called \( \{2\} \) was found, and added to the table. We will continue building the transition table in this way until there are no more states to add. The final transition table is given below.

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {0} )</td>
<td>( {0, 1} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( {0, 1} )</td>
<td>( {0, 1} )</td>
<td>( {2} )</td>
</tr>
<tr>
<td>( {2} )</td>
<td>( \emptyset )</td>
<td>( {1, 2} )</td>
</tr>
<tr>
<td>( {1, 2} )</td>
<td>( {0} )</td>
<td>( {1, 2} )</td>
</tr>
</tbody>
</table>

We can use this transition table to draw a transition diagram of our DFA which is equivalent (accepts the same language) as the original NFA.