Lecture 19: NFAs

Q: What is an NFA?
   It is a finite state machine where several choices may exist for the next state at any point.
   a. The next state function takes a state and an input symbol or the empty string and produces the set of possible next states.
   b. For a string to be accepted by the NFA, only one of the possible paths has to lead to an accept state.

Every NFA is equivalent to a DFA

Q: What does equivalent mean?

Every NFA is can be transformed to a DFA

But the complexity of the resulting FSA may increase dramatically!

To convert a NFA to a DFA, use subsets of the original set of states: these will be the new states in the DFA.

NFA:

Start with the start state, s

<table>
<thead>
<tr>
<th>( \mathcal{P}(S) )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{p}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{q}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s,p}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s,q}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{p,q}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s,p,q}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Any new "subset" state that contains an accepting state (of the original NFA) is accepting.
A. Find an equivalent DFA

B. Find an equivalent DFA
C. Find DFA, A, that is equivalent to this NFA:

What is \( L(A) \), i.e. ?

Is there a simpler DFSA that accepts this language?

What is it?
There is a method that is guaranteed to find the simplest DFA equivalent to any given DFA.

Identify k-equivalence classes. Recreate the transitions on the k-equivalence classes

Let $A$ be a finite-state automaton with next-state function $N$. Given any states $s$ and $t$ in $A$,

1) $s$ is 0-equivalent to $t$ $\iff$ either $s$ and $t$ are both accepting states or they are both nonaccepting states.

2) for every integer $k \geq 1$, $s$ is $k$-equivalent to $t$ $\iff$ $s$ and $t$ are $(k-1)$-equivalent and for any input symbol $m$, $N(s,m)$ and $N(t,m)$ are also $(k-1)$-equivalent.

Find the 0-equivalence classes, the 1-equivalence classes, and the 2-equivalence classes for the DFA

Redraw DFA using 2-equivalence classes. Map transitions onto the equivalence classes.