From Non-deterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA)

Given the following non-deterministic finite automata, convert it to the corresponding deterministic finite automata:

This is a NFA because in State q0 there are 2 transition arrows with the letter “b”. There are also some implied trap states in this NFA. To be completely accurate, a DFA must explicitly state all trap states.

Following the algorithm outlined in the text, the NFA is converted to the following DFA in these steps:

1. Start from the state q0. Notice that the state q0 is written as a set notation.

2. From this start state, draw an transition arrow for each letter of the alphabet.

3. Go back to the NFA to figure out all the state that a letter “a” could lead to from State q0. Repeat the same process for the letter “b”.
   - From state q0, the letter “a” will loop back to itself.
   - From state q0, the letter “b” will take the machine to either q0 or q1 from the original NFA. Combine the 2 states as {q0, q1}
4. Now add 2 more transition arrows for the new state just added.

\[
\begin{array}{c}
\{q_0\} \quad \{q_0, q_1\} \\
\uparrow \quad \uparrow
\end{array}
\]

\[
\begin{array}{c}
b \quad b \\
\downarrow \quad \downarrow
\end{array}
\]

5. Since the new state is a combination of \( q_0 \) and \( q_1 \), we must combine the results in figuring out all the states that the letter “a” and the letter “b” could lead to.

- From state \( q_0 \) in the original NFA, the letter “a” will lead back to itself.
- From state \( q_1 \) in the original NFA, the letter “a” will lead to an implied trap state, i.e. there wasn’t an arrow for the letter “a”. Trap states in the new DFA is denoted by the empty set \( \emptyset \).
- From state \( \{q_0, q_1\} \), the letter “a” will lead to the union of above results: \( \{q_0\} \cup \emptyset = \{q_0\} \)
- From state \( q_0 \) in the original NFA, the letter “b” will lead to either \( q_0 \) or \( q_1 \).
- From state \( q_1 \) in the original NFA, the letter “b” will lead to \( q_2 \).
- From state \( \{q_0, q_1\} \), the letter “b” will lead to the union of \( \{q_0, q_1\} \) and \( \{q_2\} = \{q_0, q_1, q_2\} \)

\[
\begin{array}{c}
\{q_0\} \quad \{q_0, q_1\} \quad \{q_0, q_1, q_2\} \\
\uparrow \quad \uparrow \quad \uparrow \\
b \quad a \quad b
\end{array}
\]

6. Since the state \( q_2 \) was a final state in NFA, any state in DFA that has \( q_2 \) in it is a final state in DFA.

\[
\begin{array}{c}
\{q_0\} \quad \{q_0, q_1\} \quad \{q_0, q_1, q_2\} \\
\uparrow \quad \uparrow \quad \uparrow \\
b \quad a \quad b
\end{array}
\]

7. Continue the process until no additional arrows can be added. Each state in the DFA has exactly 2 arrows, one for each letter of the alphabet.

\[
\begin{array}{c}
\{q_0\} \quad \{q_0, q_1\} \quad \{q_0, q_1, q_2\} \quad \{q_0, q_2\} \\
\uparrow \quad \uparrow \quad \uparrow \quad \uparrow \\
a \quad b \quad a \quad b
\end{array}
\]
NFA with lambda transitions

The second NFA example has a lambda transition arrow. The lambda transition means that you can go from one state to another state without reading any letters from the input.

Following the algorithm outlined in the text, the NFA is converted to the following DFA in these steps:

1. Start from the state q0.

2. Since there is no transition arrow for letter “b” from state q0 in the NFA, add a trap state for “b” in DFA. Trap states are denoted by $\emptyset$

3. From state q0 in NFA, the letter “a” will transition to state q1. However, once the machine is in state q1, the lambda transition is a free ride. The machine could also skip to q2 without reading any additional letters from the input. Thus, the transition from q0 for letter “a” is \{q1, q2\}.
4. From the new state \( \{q_1, q_2\} \), the letter “a” will loop in state \( q_1 \) and \( q_2 \) (because of the lambda)

5. From the new state \( \{q_1, q_2\} \), the letter “b” will lead to \( q_0 \). The DFA is now complete.