Theory of Computation

Shukla Banik

Department of Computer Science & Engineering
Siksha 'O' Anushandhan (Deemed to be University)

shuklabanik@soa.ac.in

October 16, 2020

EQUIVALENCE OF NFAS AND DFAS

Two machines are equivalent if they recognize the same language.

Theorem 1.39 [Refer to Text Book]
Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

PROOF Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing some language A. We construct a DFA $M = (Q', \Sigma, \delta', q_0{}', F')$ recognizing A. Before doing the full construction, let's first consider the easier case wherein N has no ε arrows. Later we take the ε arrows into account.

1. $Q' = \mathcal{P}(Q)$.
   Every state of M is a set of states of N. Recall that $\mathcal{P}(Q)$ is the set of subsets of Q.
2. For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \{q \in Q |\, q \in \delta(r, a) \text{ for some } r \in R\}$.
   If R is a state of M, it is also a set of states of N. When M reads a symbol a in state R, it shows where a takes each state in R. Because each state may go to a set of states, we take the union of all these sets. Another way to write this expression is

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

3. $q_0{}' = \{q_0\}$.
   M starts in the state corresponding to the collection containing just the start state of N.
4. $F' = \{R \in Q' |\, R \text{ contains an accept state of } N\}$.
   The machine M accepts if one of the possible states that N could be in at this point is an accept state.

EQUIVALENCE OF NFAS AND DFAS

Now we need to consider the ε arrows. To do so, we set up an extra bit of notation. For any state R of M, we define $E(R)$ to be the collection of states that can be reached from members of R by going only along ε arrows, including the members of R themselves. Formally, for $R \subseteq Q$ let

$$E(R) = \{q \mid q \text{ can be reached from } R \text{ by traveling along 0 or more } \varepsilon \text{ arrows}\}.$$

Then we modify the transition function of M to place additional fingers on all states that can be reached by going along ε arrows after every step. Replacing $\delta(r, a)$ by $E(\delta(r, a))$ achieves this effect. Thus

$$\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}.$$

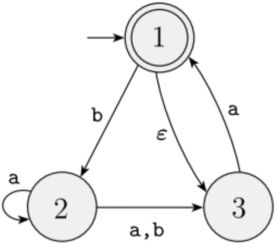
Additionally, we need to modify the start state of M to move the fingers initially to all possible states that can be reached from the start state of N along the ε arrows. Changing $q_0{}'$ to be $E(\{q_0\})$ achieves this effect. We have now completed the construction of the DFA M that simulates the NFA N.

The construction of M obviously works correctly. At every step in the computation of M on an input, it clearly enters a state that corresponds to the subset of states that N could be in at that point. Thus our proof is complete.

NFA to DFA conversion

Example 1.41 (Refer to text book)

Convert the following NFA to a DFA.



To construct a DFA D that is equivalent to N_4, we first determine D's states. N_4 has three states, $\{1,2,3\}$, so we construct D with eight states, one for each subset of N_4's states. We label each of D's states with the corresponding subset. Thus D's state set is

$$\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}.$$

NFA to DFA conversion

Next, we determine the start and accept states of D. The start state is $E(\{1\})$, the set of states that are reachable from 1 by traveling along ε arrows, plus 1 itself. An ε arrow goes from 1 to 3, so $E(\{1\}) = \{1,3\}$. The new accept states are those containing N_4's accept state; thus $\{\{1\},\{1,2\},\{1,3\},\{1,2,3\}\}$.

Finally, we determine D's transition function. Each of D's states goes to one place on input a and one place on input b. We illustrate the process of determining the placement of D's transition arrows with a few examples.

In D, state $\{2\}$ goes to $\{2,3\}$ on input a because in N_4, state 2 goes to both 2 and 3 on input a and we can't go farther from 2 or 3 along ε arrows. State $\{2\}$ goes to state $\{3\}$ on input b because in N_4, state 2 goes only to state 3 on input b and we can't go farther from 3 along ε arrows.

NFA to DFA conversion

State $\{1\}$ goes to $\emptyset$ on a because no a arrows exit it. It goes to $\{2\}$ on b. Note that the procedure in Theorem 1.39 specifies that we follow the ε arrows *after* each input symbol is read. An alternative procedure based on following the ε arrows before reading each input symbol works equally well, but that method is not illustrated in this example.

State $\{3\}$ goes to $\{1,3\}$ on a because in N_4, state 3 goes to 1 on a and 1 in turn goes to 3 with an ε arrow. State $\{3\}$ on b goes to $\emptyset$.

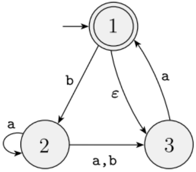
State $\{1,2\}$ on a goes to $\{2,3\}$ because 1 points at no states with a arrows, 2 points at both 2 and 3 with a arrows, and neither points anywhere with ε arrows. State $\{1,2\}$ on b goes to $\{2,3\}$. Continuing in this way, we obtain the diagram for D in Figure 1.43.

*[Refer to Text Book]

NFA to DFA conversion

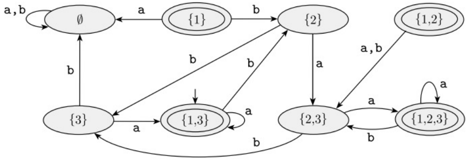
The final DFA Transition table is:

States	a	b
$1*$	ϕ	2
2	2, 3	3
3	1, 3	ϕ
$\rightarrow 1, 3*$	1, 3	2
$1, 2*$	2, 3	2, 3
2, 3	1, 2, 3	3
$1, 2, 3*$	1, 2, 3	2, 3
ϕ	ϕ	ϕ

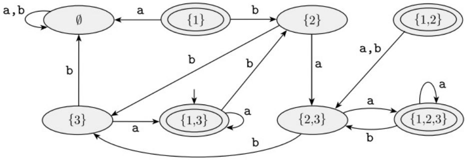


The final DFA Transition table is:

States	a	b
$1*$	ϕ	2
2	$2,3$	3
3	$1,3$	ϕ
$\rightarrow 1,3*$	$1,3$	2
$1,2*$	$2,3$	$2,3$
$2,3$	$1,2,3$	3
$1,2,3*$	$1,2,3$	$2,3$
ϕ	ϕ	ϕ



NFA to DFA conversion



We may simplify this machine by observing that no arrows point at states $\{1\}$ and $\{1, 2\}$, so they may be removed without affecting the performance of the machine. Doing so yields the following figure.