

Equivalence of DFAs and NFAs

Sipser Ch 1: p54–58

Definition. Two machines are *equivalent* if they accept the same language.

Theorem. *Every DFA D has an equivalent NFA N .*

Proof. Let DFA $D = (Q, \Sigma, \delta, q_0, F)$ be given. We define $N = (Q, \Sigma, \delta', q_0, F)$ to be such that $\delta' : Q \times \Sigma \rightarrow 2^Q$ where $\delta'(q, a) = \{\delta(q, a)\}$ for every $q \in Q$ and every $a \in \Sigma$. Clearly, for every $w \in \Sigma^*$, D accepts w iff N accepts w . \square

Theorem. *Every NFA N has an equivalent DFA D .*

Proof. First some definitions. For any state q , define the ε -closure of q , written $E(q)$, to be the set of states reachable from q via a directed path of ε -edges. If R is a any subset of states, define the ε -closure of R , written $E(R)$, to be $\bigcup_{r \in R} E(r)$.

Sipser's Method. Let NFA (possibly with ε -transitions) $N_\varepsilon = (Q, \Sigma, \delta, q_0, F)$ be given. Construct DFA $D = (2^Q, \Sigma, \delta', E(q_0), F')$ where δ' and F' are defined as

$$\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a)) \quad \text{for every } R \in 2^Q \text{ and every } a \in \Sigma,$$

and

$$F' = \{R \in 2^Q : R \cap F \neq \emptyset\}.$$

See Sipser Example 1.41, pages 56–58.

2-Step Method. Let NFA (possibly with ε -transitions) $N_\varepsilon = (Q, \Sigma, \delta, q_0, F)$ be given.

Step 1. If N_ε has no ε -transitions, let $N = N_\varepsilon$ and go to Step 2. Otherwise, construct an NFA (without ε -transitions) $N = (Q, \Sigma, \delta', q_0, F')$ as follows. The transition function $\delta' : Q \times \Sigma \rightarrow 2^Q$ is defined to be

$$\delta'(q, a) = \bigcup_{p \in E(q)} \delta(p, a) \quad \text{for every } q \in Q \text{ and every } a \in \Sigma$$

and F' is defined to be

$$F' = \{q \in Q : E(q) \cap F \neq \emptyset\}.$$

Step 2. Starting off of the NFA N from Step 1, construct a DFA $D = (2^Q, \Sigma, \delta'', \{q_0\}, F'')$ as follows. The transition function $\delta'' : 2^Q \times \Sigma \rightarrow 2^Q$ is defined to be

$$\delta''(R, a) = \bigcup_{r \in R} \delta'(r, a) \quad \text{for every } R \in 2^Q \text{ and every } a \in \Sigma$$

and F'' is defined to be

$$F'' = \{R \in 2^Q : R \cap F' \neq \emptyset\}.$$

We can show that for any string $w \in \Sigma^*$, we have N_ε accepts w iff N accepts w iff D accepts w . \square

Example. Let's demonstrate the 2-step method on the NFA N_4 on page 57 that Sipser uses. Here is the transition table for N_4 . (Convention: A start state has an arrow pointing to it and a final state is starred.)

δ	a	b	ε
$\rightarrow^* 1$	\emptyset	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	\emptyset
3	$\{1\}$	\emptyset	\emptyset

In Step 1, we remove the only ε -transition to get the following transition table for an NFA equivalent to N_4 that has no ε -transitions.

δ'	a	b
$\rightarrow^* 1$	$\{1\}$	$\{2\}$
2	$\{2, 3\}$	$\{3\}$
3	$\{1\}$	\emptyset

In Step 2, we convert the NFA without ε -transitions into an equivalent DFA. While we are at it, we make sure to eliminate states of the DFA not reachable from the start state. Here is the transition table for the DFA.

δ''	a	b
$\rightarrow^* \{1\}$	$\{1\}$	$\{2\}$
$\{2\}$	$\{2, 3\}$	$\{3\}$
$\{3\}$	$\{1\}$	\emptyset
$\{2, 3\}$	$\{1, 2, 3\}$	$\{3\}$
$^* \{1, 2, 3\}$	$\{1, 2, 3\}$	$\{2, 3\}$
\emptyset	\emptyset	\emptyset

It turns out that in this case the 2-step method gives exactly the same machine as obtained by Sipser's method, with the provision that state $\{1\}$ in the the 2-step method corresponds to state $\{1, 3\}$ in Sipser's method.

Remark. The number of states in the final DFA D can be an exponential function on the number of states of the given NFA N_ϵ . In practice, we only introduce a state of D after determining that it is reachable from the start state, like shown in our example above. However, it has been proved that there exist languages whose smallest DFA has exponentially more states than its smallest NFA.