15-453 Formal Languages, Automata, and Computation

Some Notes on Deterministic and Non-Deterministic Finite Automata

Frank Pfenning

Lecture 5
January 28, 2000

Deterministic and non-deterministic finite automata recognize the same languages. In this note we elaborate on the proof of this fundamental theorem as sketched in [Sip96, Theorem 1.19, pp. 54–56].

1 Review of Definitions

Definition 1 (DFA)
A deterministic finite automaton *(DFA) is specified by a tuple* $(Q, \Sigma, \delta, q_0, F)$ *where*

Q	*is a finite set of states,*
Σ	*is a finite alphabet,*
$\delta : Q \times \Sigma \to Q$	*is the transition function,*
$q_0 \in Q$	*is the initial state,*
$F \subseteq Q$	*is the set of final states.*

Definition 2 (DFA Computation)
A computation *of a DFA* $D = (Q, \Sigma, \delta, q_0, F)$ *starting in state* r_0 *and ending in state* r_n *is a sequence of transitions*

$$r_0 \overset{a_1}{\Longrightarrow} r_1 \overset{a_2}{\Longrightarrow} r_2 \overset{a_3}{\Longrightarrow} \cdots \overset{a_n}{\Longrightarrow} r_n$$

such that

$$\delta(r_i, a_{i+1}) = r_{i+1} \qquad \text{for } 0 \leq i < n.$$

We write

$$r_0 \overset{a_1 a_2 \ldots a_n}{\Longrightarrow} r_n$$

to abbreviate such a computation and $r \overset{\epsilon}{\Longrightarrow} r$ *for the special case that* $n = 0$.

Definition 3 (Language Recognized by DFA)
A DFA $D = (Q, \Sigma, \delta, q_0, F)$ recognizes a language $L \subseteq \Sigma^*$ *if*

$$L = \{w \mid q_0 \overset{w}{\Longrightarrow} r \text{ such that } r \in F\}$$

We write $\mathcal{L}(D)$ *for the language recognized by a DFA* D.

A non-deterministic automaton is very similar, except that it may transition to more than one state for a given input symbol. Furthermore, the machine can make a *silent transition* (also called ϵ-transition) without consuming an input symbol. We write Σ_ϵ for $\Sigma \cup \{\epsilon\}$ where ϵ is not in Σ.

Definition 4 (NFA)
A non-deterministic finite automaton *(NFA) is specified by a tuple* $(Q, \Sigma, \delta, q_0, F)$ *where*

Q	*is a finite set of states,*
Σ	*is a finite alphabet,*
$\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$	*is the transition function,*
$q_0 \in Q$	*is the initial state,*
$F \subseteq Q$	*is the set of final states.*

Definition 5 (NFA Computation)
A computation *of an NFA* $D = (Q, \Sigma, \delta, q_0, F)$ *starting in state* r_0 *and ending in state* r_n *is a sequence of transitions*

$$r_0 \stackrel{x_1}{\Longrightarrow} r_1 \stackrel{x_2}{\Longrightarrow} r_2 \stackrel{x_3}{\Longrightarrow} \cdots \stackrel{x_n}{\Longrightarrow} r_n$$

such that

$$r_{i+1} \in \delta(r_i, x_{i+1}) \qquad \text{for } 0 \leq i < n.$$

Note that $x_i \in \Sigma_\epsilon$*, that is, it may be a symbol in the alphabet or the empty word. We write*

$$r_0 \stackrel{x_1 x_2 \ldots x_n}{\Longrightarrow} r_n$$

to abbreviate such a computation and $r \stackrel{\epsilon}{\Longrightarrow} r$ *for the special case that* $n = 0$*. Note that* $\epsilon w = w \epsilon = w$*.*

Definition 6 (Language Recognized by NFA)
An NFA $N = (Q, \Sigma, \delta, q_0, F)$ recognizes a language $L \subseteq \Sigma^*$ *if*

$$L = \{w \mid q_0 \stackrel{w}{\Longrightarrow} r \text{ for some } r \text{ such that } r \in F\}$$

We write $\mathcal{L}(N)$ *for the language recognized by a DFA* N*.*

Note that the definitions of the languages recognized by a DFA and NFA are identical except for the differing underlying notion of computation.

2 DFAs and NFAs Recognize the Same Languages

The main theorem, namely that DFAs and NFAs recognize the same languages, decomposes into two lemmas. The first is easy to prove, whereas the second requires considerable amount of work.

Lemma 1 (NFAs can simulate DFAs)
Let $D = (Q, \Sigma, \delta, q_0, F)$ *be a DFA. Then there is an NFA* N *such that* $\mathcal{L}(N) = \mathcal{L}(D)$*.*

Proof: Given D, we construct N with the same set of states, initial state and accepting states. The transition function δ' behaves like δ, that is, it does not take advantage of the possibility of non-deterministic computation.

Let $N = (Q, \Sigma, \delta', q_0, F)$ where

$$\begin{array}{rcll} \delta'(r, a) & = & \{\delta(r, a)\} & \text{for } a \in \Sigma \\ \delta'(r, \epsilon) & = & \{\,\} & \end{array}$$

We need to show that $\mathcal{L}(N) = \mathcal{L}(D)$. While this is obvious in this case, we go through the details to exemplify the technique of *computation induction*, that is, induction over the structure of a computation.

Again, there are two directions to prove, $\mathcal{L}(N) \subseteq \mathcal{L}(D)$ and $\mathcal{L}(D) \subseteq \mathcal{L}(N)$.

1. $\mathcal{L}(N) \subseteq \mathcal{L}(D)$. Assume $w \in \mathcal{L}(N)$. We have to show that $w \in \mathcal{L}(D)$. We show by induction over the structure of the computation that for all states r and s,

   $$\text{if } r \stackrel{w}{\Longrightarrow}_N s \text{ then } r \stackrel{w}{\Longrightarrow}_D s$$

   where $\stackrel{w}{\Longrightarrow}_N$ represents computation in N and $\stackrel{w}{\Longrightarrow}_D$ computation in D. From this the claim follows, since the initial and final states of N and D agree.

   (a) Base case: The computation of N has the form $r \stackrel{\epsilon}{\Longrightarrow}_N r$ where $s = r$. Then $r \stackrel{\epsilon}{\Longrightarrow}_D r$.

   (b) Induction step: The computation of N has the form $r \stackrel{a}{\Longrightarrow}_N r' \stackrel{w'}{\Longrightarrow}_N s$ where $w = aw'$, $a \in \Sigma$, and $r' \in \delta'(r, a) = \{\delta(r, a)\}$. Then $r \stackrel{a}{\Longrightarrow}_D r' \stackrel{w'}{\Longrightarrow}_D s$ where the first step exists since $\delta(r, a) = r'$, and the remainder of the computation by induction hypothesis on $r' \stackrel{w'}{\Longrightarrow}_N s$.

   Note that these are the only possible cases since $\delta'(r, \epsilon) = \{\,\}$.

2. $\mathcal{L}(D) \subseteq \mathcal{L}(N)$. Assume $w \in \mathcal{L}(D)$. We have to show that $w \in \mathcal{L}(N)$. We show by induction over the structure of the computation that for all states r and s,

   $$\text{if } r \stackrel{w}{\Longrightarrow}_D s \text{ then } r \stackrel{w}{\Longrightarrow}_N s$$

   From this the claim follows since, since the initial and final states of N and D agree.

   (a) Base case: The computation of D has the form $r \stackrel{\epsilon}{\Longrightarrow}_D r$ where $s = r$. Then $r \stackrel{\epsilon}{\Longrightarrow}_N r$.

   (b) Induction step: The computation of D has the form $r \stackrel{a}{\Longrightarrow}_N r' \stackrel{w'}{\Longrightarrow}_D s$ where $w = aw'$, $a \in \Sigma$, and $r' = \delta(r, a)$. Then $r \stackrel{a}{\Longrightarrow}_N r' \stackrel{w'}{\Longrightarrow}_D s$ where the first step exists since $\delta(r, a) = \{r'\}$, and the remainder of the computation by induction hypothesis on $r' \stackrel{w'}{\Longrightarrow}_D s$.

$\square$

We now summarize the important principle of computation induction. Say we want to prove a property P for all computations $r \stackrel{w}{\Longrightarrow} s$ of a given DFA or NFA. The argument then has the following structure:

1. Base case: Show that property P holds for $r \stackrel{\epsilon}{\Longrightarrow} r$.

2. Step case: Assume that property P holds for $r' \stackrel{w'}{\Longrightarrow} s$. Show that for each possible first step $r \stackrel{x}{\Longrightarrow} r'$ the property P holds for $r \stackrel{x}{\Longrightarrow} r' \stackrel{w'}{\Longrightarrow} s$.

We then know that P must hold for all computations. As in our proof above, the property P frequently expresses that a computation in a related machine exists, thereby establishing that the computations of one machine can simulate those of another.

Lemma 2 (DFAs can simulate NFAs)
Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Then there is a DFA D such that $\mathcal{L}(D) = \mathcal{L}(N)$.

Proof: Assume we are given and NFA $N = (Q, \Sigma, \delta, q_0, F)$. We first explicitly construct a DFA $D = (Q', \Sigma, \delta', R_0, F')$ and then show that it accepts the same language as N. The machine D is designed to keep track of all states in Q that N may be in after an initial substring has been consumed. Therefore, the states of N are sets of states of D.

The main complication is the presence of ϵ-transitions in N. To model these we close every set of states under ϵ-transitions. Formally, for $R \subseteq Q$ we define

$$\mathcal{E}(R) = \{r' \in Q \mid r \stackrel{\epsilon}{\Longrightarrow}_N r' \text{ for some } r \in R\}.$$

The we define

$$\begin{array}{rcl} Q' & = & \{R \subseteq Q \mid \mathcal{E}(R) = R\} \\ \delta'(R, a) & = & \mathcal{E}(\bigcup_{r \in R} \delta(r, a)) \\ R_0 & = & \mathcal{E}(\{q_0\}) \\ F' & = & \{R \in Q' \mid r \in F \text{ for some } r \in R\} \end{array}$$

Next we have to show that this construction is correct, that is, for $D = (Q', \Sigma, \delta', R_0, F')$ we have $\mathcal{L}(D) = \mathcal{L}(N)$.

1. $\mathcal{L}(N) \subseteq \mathcal{L}(D)$. Let $w \in \mathcal{L}(N)$. We have to show that $w \in \mathcal{L}(D)$. We show this by making precise how D simulates N:

   Whenever $r \stackrel{w}{\Longrightarrow}_N s$ then for every $R \in Q'$ such that $r \in R$ we have $R \stackrel{w}{\Longrightarrow}_D S$ for an S with $s \in S$.

   From this the claim follows, since whenever $q_0 \stackrel{w}{\Longrightarrow}_N s$ with $s \in F$ then $\mathcal{E}(\{q_0\}) \stackrel{w}{\Longrightarrow}_D S$ with $s \in S$ and therefore $S \in F'$. We prove the property by computation induction.

   (a) Base case: the computation has the form $r \stackrel{\epsilon}{\Longrightarrow}_N r$. Then $R \stackrel{\epsilon}{\Longrightarrow}_D R$ for every R that contains r.

   (b) Induction step, first subcase: the computation has the form

   $$r \stackrel{a}{\Longrightarrow}_N r' \stackrel{w'}{\Longrightarrow}_N s \quad \text{for } a \in \Sigma \text{ and } r' \in \delta(r, a).$$

   Let $R \in Q'$ contain r. Then

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

   Hence we can apply the induction hypothesis to $r' \stackrel{w'}{\Longrightarrow}_N s$ and R' to obtain a computation $R' \stackrel{w'}{\Longrightarrow}_D S$ with $s \in S$. Therefore

   $$R \stackrel{a}{\Longrightarrow}_D R' \stackrel{w'}{\Longrightarrow}_D S$$

   with $s \in S$.

(c) Induction step, second subcase: the computation has the form

$$r \overset{\epsilon}{\Longrightarrow}_N r' \overset{w'}{\Longrightarrow}_N s \quad \text{where } r' \in \delta(r, \epsilon).$$

Let $R \in Q'$ such that $r \in R$. Since, by definition of Q', R is closed under the $\mathcal{E}$ operation, we have $r' \in R$. Hence we can apply the induction hypothesis to $r' \overset{w'}{\Longrightarrow}_N s$ and R to obtain a computation $R \overset{w'}{\Longrightarrow}_D S$ with $s \in S$.

2. $\mathcal{L}(D) \subseteq \mathcal{L}(N)$. Let $w \in \mathcal{L}(D)$. We have to show that $w \in \mathcal{L}(N)$. We show this via the following property:

   If $R \overset{w}{\Longrightarrow}_D S$ then $S = \bigcup_{r \in R} \{s \mid r \overset{w}{\Longrightarrow}_N s\}$.

   Again, this property is a central invariant of the construction of D to simulate N. We prove it by computation induction on the given deterministic computation.

   (a) Base case: the computation has the form $R \overset{\epsilon}{\Longrightarrow}_D R$. Then $R = \mathcal{E}(R) = \bigcup_{r \in R} \{r' \mid r \overset{\epsilon}{\Longrightarrow}_N r'\}$.

   (b) Induction step: the computation has the form

   $$R \overset{a}{\Longrightarrow}_D R' \overset{w'}{\Longrightarrow}_D S \quad \text{where } w = aw' \text{ and } a \in \Sigma.$$

   Then
   $$R' = \mathcal{E}\Big(\bigcup_{r \in R} \delta(r, a)\Big) = \bigcup_{r \in R} \{r' \mid r \overset{a}{\Longrightarrow}_N r'\}$$

   Note that R' is unique since D is deterministic. Now we can apply the induction hypothesis to $R' \overset{w'}{\Longrightarrow}_D S$ to conclude

   $$S = \bigcup_{r' \in R'} \{s \mid r' \overset{w'}{\Longrightarrow}_N s\} = \bigcup_{r \in R} \{s \mid r \overset{a}{\Longrightarrow}_N r' \overset{w'}{\Longrightarrow}_N s \text{ for some } r'\} = \bigcup_{r \in R} \{s \mid r \overset{w}{\Longrightarrow}_N s\}.$$

□

We now have the pieces to prove the overall theorem.

Theorem 1 (NFAs recognize regular languages)
A language L is regular if and only if $L = \mathcal{L}(N)$ for some NFA N.

Proof: By definition, a language L is regular iff it is recognized by a DFA D. The preceding two lemmas show that this is the case iff there is an NFA N recognizing L. □

References

[Sip96] Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, Boston, Massachusetts, 1996.