

## Supplementary Notes on Minimization of DFAs

Lecture 6

February 7, 2008

*Review these notes together with the lecture slides.*

Let  $M = (Q, \Sigma, \delta, q_0, F)$  be any DFA with alphabet  $\Sigma$ . Let  $x, y, z \in \Sigma^*$  and  $p, q \in Q$ .

$M$  defines an equivalence relation  $\sim_M$  over  $\Sigma^*$  as follows:

$$x \sim_M y \text{ iff } M \text{ ends in the same state on both } x \text{ and } y$$

Note that there is one equivalence class of  $\sim_M$  for every state in  $Q$ ; thus the number of equivalence classes of  $\sim_M$  is finite.

For brevity let us denote the language recognized by  $M$ ,  $L(M)$ , by  $L$ . Using  $L$  we can define another equivalence relation  $\sim_L$  over  $\Sigma^*$  as follows

$$x \sim_L y \text{ iff } \forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L$$

We use  $\sim_L$  to make the following definition:

**Definition 1** *Two strings (words)  $x$  and  $y$  in  $\Sigma^*$  are indistinguishable by  $L$  iff  $x \sim_L y$ .*

*Otherwise, we say that  $x$  and  $y$  are distinguishable.*

We first prove the following lemma.

**Lemma 1** *Each equivalence class of  $\sim_M$  is contained in some equivalence class of  $\sim_L$ .*

**Proof:** Suppose  $x \sim_M y$ . Let  $M$  end in the state  $q$  on both  $x$  and  $y$ . For any string  $z$ , let  $\hat{\delta}(q, z)$  denote the state reached from  $q$  on  $z$ . Thus,  $M$  ends in the same state  $\hat{\delta}(q, z)$  on both  $xz$  and  $yz$ . So,  $xz \in L$  iff  $yz \in L$ , and therefore  $x \sim_L y$ . ■

Using the above lemma, we can also prove that the number of equivalence classes of  $\sim_L$  is also finite. (Use a proof by contradiction: if there is an equivalence class  $C$  of  $\sim_L$  which does not contain an equivalence class of  $\sim_M$ , then what happens to the classes of  $\sim_M$  corresponding to states reached on strings in  $C$ ?)

We can now prove a version of the Myhill-Nerode theorem, stated below.

**Theorem 2** *Let  $L$  be a regular language over alphabet  $\Sigma$ . The equivalence relation  $\sim_L$  defines a DFA  $M_L$  recognizing  $L$ , where the states of  $M_L$  are the equivalence classes of  $\sim_L$ .  $M_L$  is the unique, minimal DFA for language  $L$  (up to isomorphism).*

**Proof:** Let  $[x]_L$  denote the equivalence class of string  $x$  under  $\sim_L$ .

Define  $M_L = (Q', \Sigma, \delta', q'_0, F')$  where:

$$\begin{aligned} Q' &= \{[x]_L \mid x \in \Sigma^*\} \\ \delta'([x]_L, a) &= [xa]_L \\ q'_0 &= [\epsilon]_L \\ F' &= \{[x]_L \mid x \in L\} \end{aligned}$$

First, note that  $M_L$  recognizes  $L$ . On receiving input  $x$ ,  $M_L$  moves to the state  $[x]_L$ . Thus, if  $x \in L$ ,  $M_L$  moves to a state in  $F'$  and therefore it accepts. If  $x \notin L$ , by definition of  $\sim_L$ ,  $M_L$  will not move to a state in  $F'$ .

Next, we show that  $M_L$  has the minimum number of states amongst all DFAs for  $L$ . To see this, let  $M$  be any other DFA recognizing  $L$ . By Lemma 1, every state of  $M$  (equivalence class of  $\sim_M$ ) is contained in some  $[x]_L$ . Further, every  $[x]_L$  contains some equivalence class of  $\sim_M$ . Hence,  $M$  has at least as many states as  $M_L$ . Further, if  $M$  and  $M_L$  have the same number of states, then the relations  $\sim_M$  and  $\sim_L$  must be identical.

Thus,  $M_L$  is the unique, minimal DFA for  $L$ , up to isomorphism. ■

*(turn page over)*

## Table Filling Algorithm

We give a detailed description of the Table-Filling Algorithm below.

Let  $M = (Q, \Sigma, \delta, q_0, F)$  be the input DFA.

1. Remove all states from  $Q$  that are unreachable from  $q_0$ . For convenience, we continue to refer to the resulting set of states as  $Q$ .
2. Initialize a table of all unordered pairs of states of  $M$  by leaving all entries unmarked.
3. For every pair  $(p, q)$  where  $p \in F$  and  $q \notin F$ , mark  $(p, q)$  to be distinguishable; viz., as a “d”.
4. Repeat until no new entries are marked “d”:
5. For every pair of distinct states  $(p, q)$  and every  $\sigma \in \Sigma$ :
6. If  $(\delta(p, \sigma), \delta(q, \sigma))$  is marked “d”, then mark  $(p, q)$  as “d”.
7. For each state  $q$ , define  $[q]$  as the set of states  $\{p \mid (p, q) \text{ is not marked “d”}\}$
8. Construct a new DFA  $M' = (Q', \Sigma, \delta', q'_0, F')$  where:  
 $Q' = \{[q] \mid q \in Q\}$   
 $\delta'([q], \sigma) = [\delta(q, \sigma)]$   
 $q'_0 = [q_0]$   
 $F' = \{[q] \mid q \in F\}$
9. The algorithm’s output is  $M'$ .