Minimization of DFSAs, closure of RLs under $\cap$

February 8, 2012

Goals

- Learn to test equivalence of simple DFSAs
- Learn to minimize simple DFSAs
- Closure under intersection: see the product construction
- Understand the proof that the product construction works

Simple DFSAs

Recall that, under our definitions, the transition function $\delta$ of a DFSA was a partial function; if the next state of the automaton was undefined for any symbol in a string $w$, the automaton was taken to reject $w$. That is because the extended transition function was defined as

$$
\hat{\delta}(q, x) \triangleq \begin{cases} 
 q & |x| = 0 (\Rightarrow x = \varepsilon) \\
 \delta(q, a) & |x| = 1 \\
 \hat{\delta}(\delta(q, a), w) & x = aw, |a| = 1, |w| \geq 1 \\
 \text{undefined} & x = aw, |a| = 1 \\
\end{cases}
$$

and we said that a string was not in the language if $\hat{\delta}(q_0, w)$ was undefined. We did this merely for our convenience, and this is not an uncommon move; but it complicates proofs (as you will see if you try to use this representation in your homework). Instead, for purposes of proofs and pedagogy, we often force $\delta$ to be a total function. This requires that we explicitly add a dead state to every automaton—a non-accepting state from which there is no escape—to allow the automaton to consume the rest of the input in peace. Wherever we would have left the state table undefined before, we can send the automaton to the dead state. We can then define the extended transition function as

$$
\hat{\delta}(q, x) \triangleq \begin{cases} 
 q & |x| = 0 (\Rightarrow x = \varepsilon) \\
 \delta(q, x) & |x| = 1 \\
 \hat{\delta}(\delta(q, a), w) & x = aw, |a| = 1, |w| \geq 1 \\
\end{cases}
$$
We also found last time that we could have some inaccessible states in an automaton: these are states which cannot be reached by any path from the start state. We can easily construct an algorithm to find only the accessible states and remove the others.

If we make the transition function of a DFSA total and remove inaccessible states, we call the DFSA a simple DFSA.

**Theorem 1.** Given a DFSA $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, construct a simple DFSA $A' = \langle Q' \cup \{q_\bot\}, \Sigma, \delta', q_0, F \rangle$, where $Q' \subseteq Q$ is the set containing only the accessible states, $q_\bot \notin Q$, $\delta'(q,a) = q_\bot$ if $\delta(q,a)$ undefined, otherwise $\delta'(q,a) = \delta(q,a)$. Then $L(A) = L(A')$.

**Proof.** Left as an exercise.

For the purposes of minimizing a DFSA, we will assume that we have a simple DFSA to start with; that is, one with a dead state, but without any inaccessible states.

**Equivalence of states**

**Definition 2.** The acceptable suffixes of a state $q$, $S(q)$, is the set of strings $w$ such that $\hat{\delta}(q,w) \in F$.

We can use this to build a notion of equivalence of states.

**Definition 3.** Two DFSA states $q_1$ and $q_2$ are equivalent if $S(q_1) = S(q_2)$. Otherwise, the two states are distinguishable.

The intuition here is that we could drop the two states and replace them with a single state and still have an acceptor for the same language; let’s look at an example.

**Example 4.** Consider the automaton:

\[
\begin{array}{c|ccc}
 & a & b \\
q_0 & q_1 & q_3 \\
q_1 & q_3 & q_2 \\
q_2 & q_3 & q_2 \\
q_3 & q_3 & q_3 \\
\end{array}
\]

This automaton accepts the language $ab^*$. Note, however, that states $q_1$ and $q_2$ are equivalent. Thus we might think we could eliminate one of the two. Note also that for two states to be equivalent does not require that they have the same inputs leading to them; just that they have the same set of acceptable suffixes.

**Theorem 5.** (State partition algorithm.) Two states are equivalent if and only if they are in the same block in the output of the following algorithm:

1. Mark all non-accepting states as being in a different block from all accepting states

2. While there is some pair of states $q_1, q_2$ which are not currently marked as being in different blocks, such that, for some $a$, $\delta(q_1,a)$ is in a different block from $\delta(q_2,a)$:
   (a) Mark $q_1, q_2$ as being in different blocks

**Proof.** See Hopcroft et al. text. Idea: for any two states $q_1, q_2$, that are distinguished by a string $w = ax$ (i.e. $w \in S(q_1)$ but $w \notin S(q_2)$), the algorithm will ensure that $\delta(q_1,a), \delta(q_2,a)$ are in different blocks.
Minimization of DFSAs

**Theorem 6.** Given some simple DFSA $A$, let $P$ be the output of the state partition algorithm on $A$. Then the language of $A$ is exactly the language of a DFSA $A' = \langle P, \Sigma, \delta', q_0', F' \rangle$, where:

1. $F' = \{ A \in P | \text{ } A \cap F \neq \{ \} \}$
2. $q_0'$ is the unique $A \in P$ such that $q_0 \in A$
3. $\delta'(A, a) = B$, where $B$ is the unique block of $P$ such that, for all $q \in A$, $\delta(q, a) \in B$

**Proof.** Left as an exercise. Idea: the states in each block are either all accepting or all non-accepting; furthermore, the acceptable suffixes of a block will always be the same as the acceptable suffixes of all the states in the block. □

**Theorem 7.** For any language $L$, construct some simple DFSA $A$ such that $L = L(A)$. Construct $A'$ by the method outlined in Theorem 6. Then there is no smaller simple DFSA for $L$.

**Proof.** Suppose there were some smaller DFSA $A^*$. For each state in $A'$, there is at least one equivalent state in $A^*$, since the start states must always be equivalent in any two equivalent DFSAs, and, therefore, given any string $w$ with $\delta'(q_0', w) = q'$ and $\delta^*(q_0^*, w) = q^*$, $q'$ and $q^*$ must also be equivalent. By the Pigeonhole Principle, if $A^*$ is smaller than $A'$, there must be some state $q$ in $A^*$ that is equivalent to two states $q_1$, $q_2$ in $A'$. But then $q_1$ and $q_2$ must be equivalent: if not, there would be some suffix $w$ distinguishing the two such that (without loss of generality) $\delta(q_1, w)$ is an accepting state and $\delta(q_2, w)$ is not an accepting state. But then either $q, q_1$ or $q, q_2$ are distinguishable; by contradiction, we see that there can be no such $A^*$. □

It is easy to see that, not only is $A'$ the smallest DFSA for $L$, it is also isomorphic (i.e., same up to state-relabeling) to any DFSA constructed in the same way by starting from some other initial DFSA $B$. Thus there is really only one minimal DFSA constructed in this way.

A closure property of regular languages

**Theorem 8.** For any pair of regular languages $L_1$ and $L_2$, the language $L = L_1 \cap L_2$ is also regular.

**Proof.** (Product construction.) Let $A_1 = \langle Q_1, \Sigma_1, \delta_1, q_{01}, F_1 \rangle$ be a simple DFSA for $L_1$; let $A_2 = \langle Q_2, \Sigma_2, \delta_2, q_{02}, F_2 \rangle$ be a simple DFSA for $L_2$. Construct $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ as follows:

\[
\begin{align*}
Q &= Q_1 \times Q_2 \\
\Sigma &= \Sigma_1 \cup \Sigma_2 \\
\delta\left(\langle q_1, q_2 \rangle, a \right) &= \langle \delta(q_1, a), \delta(q_2, a) \rangle \\
q_0 &= \langle q_{01}, q_{02} \rangle \\
F &= F_1 \times F_2
\end{align*}
\]

We show that $w \in L(A) \leftrightarrow w \in L(A_1) \& w \in L(A_2)$ by showing that $\hat{\delta}(\langle q_1, q_2 \rangle, w) = \langle \hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w) \rangle$. By induction on $|w|$. In the base case, $|w| = 0$. Then $\hat{\delta}(\langle q_1, q_2 \rangle, w) = \langle q_1, q_2 \rangle$ by definition and by definition of $F$ the claim holds. Suppose the claim holds for $|w| = k$. Let $w = ax$. Then $\hat{\delta}(\langle q_1, q_2 \rangle, ax) = \hat{\delta}(\langle \delta_1(q_1, a), \delta_2(q_2, a) \rangle, x)$. By the inductive hypothesis this is $\langle \hat{\delta}_1(\delta_1(q_1, a), x), \hat{\delta}_2(\delta_2(q_2, a), x) \rangle = \langle \hat{\delta}_1(q_1, x), \hat{\delta}_2(q_2, x) \rangle$ and the claim holds by induction. □