Assignment 1

ECS 220 — Prof. Rogaway — Winter 2006

Mark Gondree
gondree@cs.ucdavis.edu

January 17, 2006

Problem 1

Since these machines compute a function f, and do not accept or reject, we do not need a set of accepting states F. We define the computation to be done, in both cases, when the entire input has been read. Each machine has the form $M = (Q, \Sigma_1, \Sigma_2, \delta, v, q_0)$, where

Q is a finite set of states
$q_0 \in Q$ is the machine's start state
Σ_1 is the input alphabet
Σ_2 is the output alphabet
$\delta : Q \times \Sigma_1 \to Q$ is the transition function, defined as in a normal DFA
v is defined below, for each case.

A) Let $M = (Q, \Sigma_1, \Sigma_2, \delta, v, q_0)$ be a *Moore Machine*. $Q, \Sigma_1, \Sigma_2, \delta, q_0$ are defined as above, and

$$v : Q \to \Sigma_2 \text{ is a function to annotate states with characters from } \Sigma_2$$

$v(q)$ gives the symbol annotating state q.

The function $f : \Sigma_1^* \to \Sigma_2^*$ computed by M is the following:

$$f(x_1 \ldots x_n) = y \text{ if } y = v(q_1) \ldots v(q_n) \text{ where, for } 0 < i \leq n,\ \delta(q_{i-1}, x_i) = q_i$$

B) Let $M = (Q, \Sigma_1, \Sigma_2, \delta, v, q_0)$ be a *Mealey Machine*. $Q, \Sigma_1, \Sigma_2, \delta, q_0$ are defined as above, and

$$v : Q \times \Sigma_1 \to \Sigma_2 \text{ is a function to annotate transitions with characters from } \Sigma_2$$

$v(q, x)$ gives the symbol annotating the transition out of q upon reading character x.

The function $f : \Sigma_1^* \to \Sigma_2^*$ computed by M is the following:

$$f(x_1 \ldots x_n) = y \text{ if } y = v(q_0, x_1) \ldots v(q_{n-1}, x_n) \text{ where, for } 0 < i \leq n,\ \delta(q_{i-1}, x_i) = q_i$$

C) f is computable by a Mealey machine iff f is computable by a Moore machine.

Proof.

$\Longleftarrow$ Given a Moore machine $A = (Q, \Sigma_1, \Sigma_2, \delta, v, q_0)$ computing f, we can build a Mealey machine $B = (Q, \Sigma_1, \Sigma_2, \delta, v' q_0)$ that computes f by defining:

$$v'(q_i, a) = v(q_j) \text{ when } \delta(q_i, a) = q_j$$

$\Longrightarrow$ Given a Mealey machine $A = (Q, \Sigma_1, \Sigma_2, \delta, v, q_0)$ computing f, we can build a Moore machine $B = (Q', \Sigma_1, \Sigma_2, \delta', v', q_0')$ that computes f by defining:

$$\begin{aligned}
&Q' \subseteq \{\langle q_j, b\rangle \mid q_j \in Q, b \in \Sigma_2\}\\
&q_0' = \langle q_0, \varepsilon\rangle\\
&v'(\langle q_j, b\rangle) = b\\
&\delta'(\langle q_j, b\rangle, a) = \langle \delta(q_j, a), v(q_j, a)\rangle
\end{aligned}$$

□

Problem 2

No – there are minimal machines with equal numbers of states that accept the same language that cannot be transformed into each other via renaming. Here is an example:

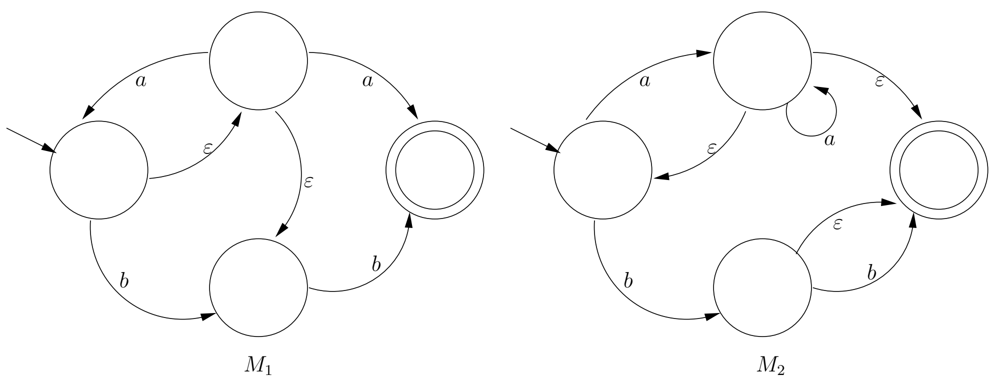

Figure 1: Two NFAs with equivalent languages. This is easiest to see by building their regular expressions: $L(M_1) = \{a^*a, a^*b, a^*bb\}$ and $L(M_2) = \{aa^*, aa^*b, aa^*bb, b, bb\}$; its clear these are the same.

Each compute the same language. Each is minimal: four states is minimal since $b \not\approx_L bb$ (append b), $a \not\approx_L b, bb$ (append a), $\varepsilon \not\approx_L a, b, bb$ (append ε). Its clear there is no renaming function which can transform M_1 into M_2 or vice-versa: M_2 has an ε-move to the final state, where M_1 has no such move.

Problem 3

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a 2-way finite automaton (2WFA), where $\delta : Q \times (\Sigma \cup \{\rhd, \lhd\}) \to Q \times \{L, R\}$, and Q, Σ, q_0, F are defined as in a normal DFA.

I will follow the convention established in class, namely that $x \in L(M)$ when M accepts input $\rhd x \lhd$, and M accepts an input when its head reads the right-marker $\lhd$ when the machine is in a final state $q \in F$. If the machine never does this, it rejects the input.

Definition 1 (right-left transcript)**.** Let $T_M^i(x) \in (Q \times \{L, R\})^*$ be the right-left transcript of M under the ith character of x on input $\rhd x \lhd$. $T_M^i(x)$ is a list of moves. The jth move is (q, L) if M's head was pointing at the $i+1$th input character, is now pointing at the ith input character, is in state q, and has pointed at this character $j-1$ times before. Alternately, it is (q, R) if its head was pointing at the $i-1$th input character, etc. In short, $T_M^i(x)$ lists the values of δ that caused M to point to x's ith position, in the order of M's execution on x.

Definition 2 (right-only transcript)**.** Let $Right(T_M^i(x)) \in (Q \times \{R\})^*$ be those moves of $T_M^i(x)$ of the form (q, R) for any $q \in Q$. It is an edited right-left transcript, dropping all moves that came from the right. This is the the right-only transcript of M under i on input x.

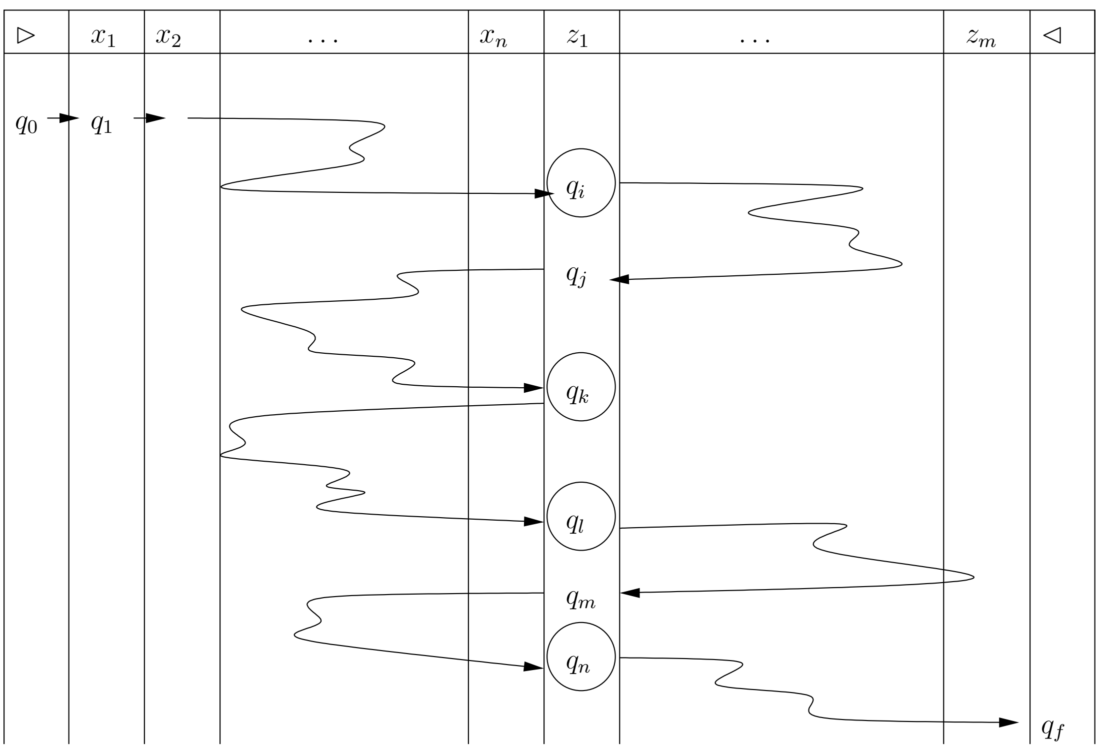

Figure 2: The trace of a 2WFA M on input $x = x_1 \ldots x_n z_1 \ldots z_m$. The right-left transcript $T_M^{n+1}(x)$ is given by reading down the relevant data in the column below character z_1. The right-only transcript $Right(T_M^{n+1}(x))$ is given by reading down the data about the circled states in this column.

Lemma 1. $\mathcal{L}$ is regular iff $\mathcal{L} = L(M)$ for some 2WFA M.

Proof.

$\implies$ If $\mathcal{L}$ is regular, then it is accepted by some DFA, and every DFA is (essentially) a 2WFA.

$\impliedby$ If M is a 2WFA, then $\mathcal{L} = L(M)$ is regular: Lemma 2 claims that every 2WFA partitions Σ^* into a finite number of equivalence classes that respect the equivalence $\approx_{\mathcal{L}}$, and (by the Myhill-Nerode theorem) if $\Sigma^* / \approx_{\mathcal{L}}$ is finite, then $\mathcal{L}$ is regular.

$\square$

Lemma 2. Any 2WFA partitions Σ^* into a finite number of equivalence classes, that each respect the equivalence $\approx_{\mathcal{L}}$.

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a 2WFA.

Consider $x, y \in \Sigma^*$. Let $[x] = [y]$ (their classes are the same) if, for all $z \in \Sigma^*$, either M enters an infinite loop on xz and yz, or $Right(T_M^{|x|+1}(xz)) = Right(T_M^{|y|+1}(yz))$.

For any z, if the right-only transcript is ε then M has entered an infinite loop before reading the first character of z. For any z, if the right-only transcript is infintely long, then M has entered an infinite loop. Infinitely long transcripts can be considered 0-length transcripts, a special finite-length transcript. So, we only need to consider finite-length right-only transcripts.

So, $[x] \neq [y]$ in the case that some z causes two finite, unequal, right-only transcripts. If $q \in Q$ appears twice in any right-transcript, then M will enter an infinite loop and the transcript will not be finite. The number of finite right-only transcripts is bound by $(|Q|+1)! \geq \sum_{i=0}^{|Q|} |Q|!/i!$, the sum of the ways to order $|Q| - i$ objects drawn from a set of size $|Q|$. This is finite, thus there are a finite number of different classes.

If $x_1 \in [x]$ and $x_2 \in [x]$, then $x_1 \approx_{\mathcal{L}} x_2$. By definition, any z causes $x_1 z$ and $x_2 z$ to produce the same right-transcripts. By definition, $x_1 \approx_{\mathcal{L}} x_2$ means, for any z, $x_1 z \in \mathcal{L} \iff x_2 z \in \mathcal{L}$. In fact, if the last moves in the two right-only transcripts agree, then $x_1 z$ and $x_2 z$ will either both be accepted or both be rejected (since, at this point, M never again moves far enough left to read the characters of x_1 or x_2, so its accept/reject behavior is entirely based on the last state of the right-only transcript and z, which are identical for $x_1 z$ and $x_2 z$).

Thus, $\#\{[x] \mid x \in \Sigma^*\}$ is finite and members of $[x]$ respect the $\approx_{\mathcal{L}}$ equivalence. $\square$

Problem 4

If $L \subseteq 1^*$ then L^* is regular.

Proof.

(1) If L is finite, then it is regular and L^* is regular.

(2) By lemma 3, if L is infinite and contains two words of coprime lengths, then L^* is regular.

(3) By lemma 4, if L is infinite and every word of L has a length that is divisible by some integer $m > 1$, then L^* is regular.

(4) By lemma 5, no other cases exist.

$\square$

Lemma 3. If $L \subseteq 1^*$ is infinite and contains two words of coprime lengths, then there is some sufficiently large c_0 such that all words of length at least c_0 are in L^*. So, any word in $1^* - L^*$ has a length less than c_0. There are a finite number of such unary words. So, $1^* - L^*$ is finite, and therefore regular. The complement of any regular language is regular, so L^* is regular

Proof. Take $1^p, 1^q \in L$ such that $\gcd(p, q) = 1$. Thus, $\{p^i \mod q \mid 0 \leq i < q\}$ contains all residues. Any c has some residue, $c \equiv x \mod q$. For some i, $p^i \equiv x \mod q$. Let $c = x + nq$ and $p^i = x + mq$. For $c_0 = q + p^{q-1}$, any $c > c_0$ can be represented as $c = p^i + (n - m)q$.

Because $c = p^i + (n - m)q$, we can form 1^c by concatenating $(n - m)$ copies of 1^q to p^{i-1} copies of p. Thus, for all $c > c_0, 1^c \in L^*$. $\square$

Lemma 4. If L is infinite and every word in L has a length that is divisible by some integer $m > 1$, then L^* is regular.

Proof. Let $S = \{x \in \mathbb{N} \mid 1^x \in L\}$, the set of lengths of words in L. Take a, b to be the smallest two elements in S. If $\gcd(a, b) = n$ then $n|x$ for any $x \in S$, by the premise. Also, it is clear that if $1^y \in L^*$ then $n|y$.

Let $S_n = \{x/n \mid x \in S\}$. S_n has two coprime elements, a/n and b/n. Let $L_n = \{1^y \mid y \in S_n\}$. By lemma 3, L_n^* is regular.

Thus, $1^x \in L^*$ if $x \equiv 0 \mod n$ and $1^{x/n} \in L_n^*$. This is a regular language — simply take the DFA accepting L_n^* and add a chain of $n - 1$ non-accepting states between every two states in the original machine. $\square$

Lemma 5. If L is not finite, then either there are two words $1^p, 1^q \in L$ such that $\gcd(p, q) = 1$ or there is some integer $m > 1$ such that $m|x$ for all $1^x \in L$.

Proof. Let $S = \{x \in \mathbb{N} \mid 1^x \in L\}$, the set of lengths of words in L. Assume that for any two $x, y \in S$, $\gcd(x, y) > 1$. Also assume that there is no integer $m > 1$ such that $m|x$ for all $x \in S$.

Since there is no m that divides every element in S, there must be some $z \in S$ such that $\gcd(x, y)$ does not share any factor m with z (the only other option being that every z shares a factor with $\gcd(x, y)$ but none of these factors are the same, which means $\gcd(x, y)$ has an infinite number of factors which is impossible for any finite number). Thus, $\gcd(\gcd(x, y), z) = 1$. So either $\gcd(x, z) = 1$ or $\gcd(y, z)$ = 1, which is a contradiction. $\square$