Autumn 1997–98, Prof. Boneh
Solution Set
CS 154, Problem Set 1

1.1 (3 points)

PROBLEM: **If $M_1 = (Q_1, \Sigma, \delta_1, q_0^{(1)}, F_1)$ recognizes the regular language L_1, and $M_2 = (Q_2, \Sigma, \delta_2, q_0^{(2)}, F_2)$ recognizes the regular language L_2, the automaton recognizing $L_1 \cap L_2$ is defined as $M = (Q_1 \times Q_2, \Sigma, \delta_M, (q_0^{(1)}, q_0^{(2)}), F_M)$, where $\delta_M((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ and $F_M = (q_1, q_2)|q_1 \in F_1$ and $q_2 \in F_2)$. Prove that M recognizes $Ł_1 \cap L_2$.**

We go in two directions: **(1)** all words accepted by M are in $L_1 \cap L_2$, and **(2)** all words in $L_1 \cap L_2$ are accepted by M. (If we had wanted, would could have proven the contrapositive of the second statement instead: **(2')** all words not accepted by M are not in $L_1 \cap L_2$.)

(1) When doing this problem, you may have wondered, "Why isn't this obvious?" The reason it seems obvious is due, in large part, to how we named the states of M. Calling each state of M a pair makes it seem obvious that the first part of the pair is what M_1 does and the second part is what M_2 does. And in fact, that is the intended interpretation. But state names are just labels, without any meaning associated with them; the fact that we intend a state to be interpreted a certain way doesn't mean it's actually legitimate to do so. We must prove it is. In other words, if we just called the states of M "a, b, etc." then the proof wouldn't be so obvious.

So we need to prove that if M is in state $\left(q_i^{(1)}, q_i^{(2)}\right)$ upon reading the first i characters of x, then M_1 is in state $q_i^{(1)}$, and M_2 in state $q_i^{(2)}$, upon reading the first i characters of x. The base case, $i = 0$, is easy, since we're given the start states of each machine. M starts in $\left(q_0^{(1)}, q_0^{(2)}\right)$, while M_1 starts in $q_0^{(1)}$ and M_2 starts in $q_0^{(2)}$. Thus, our claim is true for $i = 0$.

Suppose it's true for some arbitrary i, that is, when reading the first i characters of x, M is in state $\left(q_i^{(1)}, q_i^{(2)}\right)$, M_1 is in state $q_i^{(1)}$, and M_2 is in state $q_i^{(2)}$. Then what about $i + 1$? By the definition of δ_M, $\delta_M\left((q_i^{(1)}, q_i^{(2)}), x_i\right) = \left(\delta_1(q_i^{(1)}, x_i), \delta_2(q_i^{(2)}, x_i)\right)$. So after reading $i + 1$ characters of x, M will be in state $\left(\delta_1(q_i^{(1)}, x_i), \delta_2(q_i^{(2)}, x_i)\right)$. But upon reading the $i + 1^{st}$ characeter of x, M_1 will have gone from $q_i^{(1)}$ to $\delta_1(q_i^{(1)}, x_i)$ and M_2 from $q_i^{(2)}$ to $\delta_2(q_i^{(2)}, x_i)$. Thus, the induction holds: after reading $i+1$ characters of x, M is in a state (a, b) while M_1 is in state a and M_2 is in state b.

In particular, after reading all the characters of x, M will be in an accepting state (q_1, q_2), since M accepts x (by hypothesis). This means that after reading all of x, M_1 is in state q_1 and M_2 is in state q_2 (by our induction, above). But by construction of F_M, if $(q_1, q_2) \in F_M$, then $q_1 \in F_1$ and $q_2 \in F_2$. Thus, M_1 ends up in an accepting state upon reading x, and M_2 does too, so $x \in L_1$ and $x \in L_2$; that is, $x \in L_1 \cap L_2$.

Whew! Notice that the heart of the proof was proving formally (via induction), that state (a, b) in M corresponds to state a in M_1 and state b in M_2.

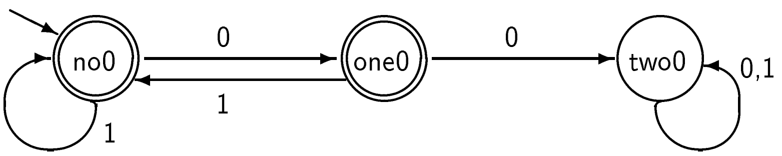
(2) If $x \in L_1 \cap L_2$ then the sequence of states in $M_1 = \left(q_0^{(1)}, q_1^{(1)}, \ldots, q_n^{(1)}\right)$ ends in an accepting state. Likewise, $\left(q_0^{(2)}, q_1^{(2)}, \ldots q_n^{(2)}\right)$ ends in an accepting state. By construction of M, when M is given x it starts in $\left(q_0^{(1)}, q_0^{(2)}\right)$. By the definition of δ_M, M then proceeds to $\left(\delta_1(q_0^{(1)}, x_1), \delta_2(q_0^{(1)}, x_1)\right) = \left(q_1^{(1)}, q_1^{(2)}\right)$. Applying this definition repeatedly, M ends up in state $\left(q_n^{(1)}, q_n^{(2)}\right)$. Since $q_n^{(1)} \in F_1$ and $q_n^{(2)} \in F_2$ (by hypothesis), $\left(q_n^{(1)}, q_n^{(2)}\right) \in F_M$ by the definition of

F_M. Thus M starts in a starting state, ends in an accepting state, and makes transitions according to δ_M, and thus the computation of M on x is an accepting computation.

This direction is easier than the other direction because we are recapitulating the construction of M, rather than deconstructing M into its components.

1.2 (2 points)

PROBLEM: **(1) Design a deterministic automaton that accepts all binary words that do not contain two consecutive zeros. (2) Build a regular expression for the language from part (1).**



The states represent how many 0s we've seen since the last 1. Once it reaches two, we're doomed, so TWO0 is a pit state.

Formally, we have $M = (\{\text{NO}0, \text{ONE}0, \text{TWO}0\}, \{0,1\}, \delta, \text{NO}0, \{\text{NO}0, \text{ONE}0\})$, where δ is as follows:

q	σ	$\delta(q,\sigma)$
NO0	0	ONE0
NO0	1	NO0
ONE0	0	TWO0
ONE0	1	NO0
TWO0	0	TWO0
TWO0	1	TWO0

Regular expression: $(\mathbf{1} \cup \mathbf{01})^*(\mathbf{0} \cup \epsilon)$. I formed this by figuring out all strings that ended in NO0. There are two possible loops: the 1 loop and the 0–ONE0–1 loop. We can repeat these (or choose not to do them at all). The strings ending in ONE0 look the same, except they have a 0 appended to them. This gives $(\mathbf{1} \cup \mathbf{01})^* \cup (\mathbf{1} \cup \mathbf{01})^*\mathbf{0}$, which I simplified.

$(\mathbf{1}^*(\mathbf{01} \cup \epsilon))^*(\mathbf{0} \cup \epsilon)$ is another possible correct answer. There are many more.

1.3 (2 points)

PROBLEM: **Given $M = (Q, \Sigma, \delta, q_0, F)$ accepting L, describe the language accepted by $M' = (Q, \Sigma, \delta, q_0, Q \setminus F)$.**

M' turns all accepting states into non-accepting states, and vice versa. For a given string x, if M ends up in a non-accepting state, M' ends in an accepting state, while if M accepts, M' does not. Thus, M' accepts exactly the strings not accepted by M, or $\overline{L}$ (which is defined to be $\Sigma^* \setminus L$).

1.4 (3 points)

PROBLEM: **Given $w = w_1 w_2 \cdots w_n$, w^r is defined to be $w_n w_{n-1} \cdots w_1$. If L is a regular language, prove that $L^r = \{x^r | x \in L\}$ is also a regular language.**

To prove L^r is regular, it's enough to give a DFA accepting L^r. Since we know we can convert NFAs to DFAs, it's also enough to give an **NFA** accepting L^r, which is what we'll do.

Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA accepting L. We construct an NFA M' accepting L^r as follows (note I call the start state s instead of q_0; it's to avoid confusion in the proof below):

$$M' = (Q \cup \{q'\}, \Sigma, \delta', q', \{s\})$$

where q' is a new state and δ' is defined as follows:

- $\delta'(q', \epsilon) = F$
- $\delta'(q', \sigma) = \emptyset$ for all $\sigma \in \Sigma$
- $\delta'(q_1, \sigma)$ is the set of all states q_2 (possibly empty!) for which $\delta(q_2, \sigma) = q_1$.

Intuitively, the old final states become start states (since they and only they are connected to the new start state, by an empty transition no less), the old start state because the only final state, and all arrows in M are reversed in M'.

Of course, we have to prove that M' accepts L^r. We have an if-and-only-if proof: M' accepts x iff $x \in L^r$.

$\Rightarrow$

Here we assume M' accepts x and prove x is in L^r. The intuition of this proof is to show that by reversing arrows, M' accepts the reverse of what M does. It's a little complicated by the non-determinism of q', but we can take care of that quickly.

Since M' accepts x, there is some accepting computation $q_0, q_1, q_2, \ldots, q_{n+1}$ (the $n+1$ is because one transition is on ϵ)x. By definition of M', q_0 is q', and since the only transitions out of q' are to states in F, $q_1 \in F$. By construction of δ', q_2 is a state such that $\delta(q_2, x_1) = q_1$. By following this rule, we see (this can be proved formally by induction) that in general, q_i is a state that has the property $\delta(q_{i+1}, x_i) = q_i$. Because of this, if M is given x starting in state q_{n+1}, M will end in state q_1.

But since $x \in M'$, the accepting computation (for M') must end in $F' = s$, so M *is* given x starting in state $q_{n+1} = s$. Since $q_1 \in F$, as we noted, M accepts x, since it starts in a start state, ends in a final state, and follows transitions using δ.

$\Leftarrow$

In this part we assume $x \in L^r$ and prove M' accepts it. Since $x \in L^r$, x^r is in L, so M accepts x^r. Let $q_0, q_1, \ldots q_n$ be the accepting computation for x^r in M. We claim $q', q_n, \ldots, q_1, q_0$ is an accepting computation of x in M'. Since M accepts, $q_n \in F$ so clearly the first transition, on ϵ, is legal. Furthermore, since M accepts L^r, this means that the transitions of M are of the form $\delta(q_i, x_{n-i}) = q_{i+1}$. Therefore, by the definition of δ', $\delta'(q_{i+1}, x_{n-i}) = \{q_i\}$ is a legal transition of M'. Given this, for a string $x_1 x_2 \cdots x_n$, it is easy to see that $q_n, q_{n-1}, \ldots q_0$ is a legal sequence of states in M'. Thus $q', q_n, \ldots, q_1, q_0$ is a computation of M' on x, as claimed, and since q' is the start state of M' and q_0 is a final state, it's an accepting computation.

Thus M' accepts the language L^r, and because of this L^r must be regular.