CSE 547T Automata Theory March 21, 2017

Homework 3 Practice Problem Solutions

WARNING: if you haven't at least tried hard to solve the practice problems before reading these solutions, you are missing the point. If you can't make *any* progress, talk to me or to the TAs before reading these solutions. Otherwise, you should come up with a solution of your own that you can compare to the one shown here.

1. We will construct a Turing machine M that, given the input 0^i, accepts iff i is a prime number.

   The TM M has two tapes, its input tape and a *divisor tape*. M starts with 0^i on its input tape and proceeds as follows:

   (a) First, M checks whether its input is ε or 0 (corresponding to $i = 0$ and $i = 1$, respectively). In either case, M rejects.

   (b) Otherwise, M initializes its divisor tape with the string 00 and proceeds to the following loop.

   (c) If the input length is the same as the length of the string on the divisor tape, M halts and accepts.

   This comparison may be performed as follows. M seeks to the beginnings of the strings on both input and divisor tapes and advances the heads of both tapes in tandem. If one head sees Δ before the other, the strings are of different lengths; otherwise, they are of the same length.

   (d) M next determines whether the value on the input tape is divisible by the value on the divisor tape. If so, M halts and rejects.

   This check may be performed as follows. M seeks to the beginnings of the strings on both input and divisor tapes and again advances the heads of both tapes in tandem. If the divisor head sees Δ while the input head sees 0, then M resets the divisor head to the beginning of its string and continues. If the input head sees Δ while the divisor head sees 0, then the divisor *does not* evenly divide the input. If both heads see Δ at the same time, then the divisor *does* divide the input.

   (e) If the input is not divisible by the current divisor, M increments the value on the divisor tape and goes to step 3.

   The increment can be performed by seeking to the end of the divisor and appending a single 0.

   Observe that M rejects the numbers 0, 1, and every n such that trial division by all values between 2 and $n-1$ discovers a divisor of n. All these numbers are composite. M accepts precisely those $n \geq 2$ that fail the trial division test for all values $< n$; these are the primes.

2. **(a)** Let L_1 and L_2 be two RE languages. By definition, there exist Turing machines M_1 and M_2 accepting L_1 and L_2, respectively.

   We will construct a nondeterministic TM M_{12} accepting the language $L_1 \cdot L_2$; since NTMs and deterministic TMs are equally powerful, this is enough to show that $L_1 \cdot L_2$ is RE. At a high level, M_{12} proceeds as follows. On input x, it first nondeterministically breaks x into two parts yz, then simulates M_1 on y and M_2 on z (in some order). If both machines accept their respective inputs, them M_{12} accepts x. Otherwise, M_{12} either rejects x (if both machines halt) or runs forever (if at least one of M_1 and M_2 does so).

This construction is not quite complete because it is not clear how M_{12} nondeterministically breaks x into parts, such that every move it makes has only *bounded* nondeterminism (i.e. a fixed number of choices, indpendendent of $|x|$). Here is a computational phase that will do the break. M_{12} starts the phase with its head immediately to the left of the input x. It then makes a series of nondeterministic moves as follows. On each move, the TM either (1) marks the current cell and ends the phase or (2) moves its head one cell to the right. In any case, the phase ends if the head runs past the right end of the input.

If $x = x_1 \ldots x_n$, and the mark is made on the ith input character x_i, then the two strings are $y = x_1 \ldots x_i$ and $z = x_{i+1} \ldots x_n$. Every possible split $x = yz$ can be formed this way. Note in particular that we allow $y = \varepsilon$ if the mark is made on the blank before the input begins, or $z = \varepsilon$ if the mark is made on x_n. (If no mark is made, M_{12} may simply fail.)

To see that M_{12} accepts $L_1 \cdot L_2$, note first that if $x \in L_1 \cdot L_2$, then for some choice of mark position, and hence some division $x = yz$, $y \in L_1$ and $z \in L_2$, and so y and z will be accepted by M_1 and M_2 respectively, causing M_{12} to accept. Conversely, if $x \notin L_1 \cdot L_2$, then no division $x = yz$ can cause both machines to accept, and so M_{12} will not accept x.

(b) Let L be an RE language accepted by Turing machine M.

We will construct a nondeterministic TM M^* accepting the language L^*. On input x, M^* first checks whether $x = \varepsilon$; if so, it accepts immediately. Otherwise, M^* nondeterministically breaks x into k substrings $y_1 \ldots y_k$, $1 \leq k \leq |x|$, and serially simulates M on each y_i. If all y_i are accepted, then M^* accepts x. If any y_i is rejected, M^* rejects x, and if any y_i causes M to run forever, then M^* naturally runs forever.

Once again, we need to specify a computational phase by which M^* breaks x into its parts. At the beginning of this phase, M^* starts with its head over the leftmost character of x. It then performs a series of nondeterministic moves as follows. On each move, M^* may either (1) move the head right one cell, or (2) mark the current cell and *then* move the head right. The phase ends when the head reaches the last character of x, which is *always* marked.

If $x = x_1 \ldots x_n$, and marks are made at positions $j_1 \ldots j_k$, then we set $y_i = x_{j_{i-1}+1} \ldots x_{j_i}$, where we take $j_0 = 0$. Note that this scheme guarantees that every y_i is nonempty, as desired.

To see that M^* accepts L^*, note first that if $x \in L^*$, then either $x = \varepsilon$, or $x = y_1 \ldots y_k$, where each y_i is a nonempty string in L and $1 \leq k \leq |x|$. In the first case, M^* accepts x immediately; in the second, for the given division of x, M accepts each of the constituent y_i's, and so M^* accepts x. Conversely, if $x \notin L^*$, then $x \neq \varepsilon$, and no division of x into parts is possible with every part in L. In this case, M will not accept at least one y_i, and so M^* will not accept x.

3. **(a)** We claim that the language

$$EVEN = \{x \mid x = e(M), M \text{ is a TM}, M \text{ accepts } \varepsilon \text{ in an even } \# \text{ of moves}\}$$

is *not* recursive. We will perform a reduction from the nonrecursive language

$$SA = \{x \mid x = e(M), M \text{ is a TM}, e(M) \in L(M)\}.$$

For the following construction, we will use a trick to extend a TM $T = (Q, \Sigma, \Gamma, q_0, \delta)$ with a *parity counter*, which remembers at all times whether T has made an odd or even number of moves. We replace each $q \in Q \cup \{h_a, h_r\}$ by two states $\langle q, 0 \rangle$ and $\langle q, 1 \rangle$, indicating that the number of moves made so far is even or odd respectively. The start state becomes $\langle q_0, 0 \rangle$. Each move $\delta(q, a) = (q', b, d)$ is replaced by the two moves

$$\delta(\langle q, 0\rangle, a) = (\langle q', 1\rangle, b, d)$$
$$\delta(\langle q, 1\rangle, a) = (\langle q', 0\rangle, b, d).$$

If (and only if) $x \in L(T)$, then the extended T will eventually enter one of the states $\langle h_a, 0\rangle$ or $\langle h_a, 1\rangle$. To ensure that the extended T accepts x in an even number of moves, we designate $\langle h_a, 0\rangle$ as the "real" accepting state and add the legal moves

$$\delta(\langle h_a, 1\rangle, a) = (\langle h_a, 0\rangle, a, S)$$

for all $a \in \Gamma$.

Let M_2 be a TM deciding EVEN. We will construct a TM M_1 deciding SA. On input x, M_1 first checks whether $x = e(M)$ for some Turing machine M; if not, M_1 rejects x. Otherwise, M_1 constructs the following TM M':

- On input y, M' first simulates M on $e(M)$.
- If M rejects $e(M)$, then M' rejects y.
- If M accepts $e(M)$, then M' accepts y.
- M' is extended using the parity counter trick, so that any string it accepts is accepted in an even number of moves.

Finally, M_1 simulates M_2 on $e(M')$ and accepts/rejects x iff M_2 accepts/rejects $e(M')$.

We claim that M_1 decides SA. First, suppose $x \in SA$. Now $x = e(M)$ for a TM M, so it passes the initial test of M_1. Moreover, M accepts $e(M)$, so M' accepts *every* input (in particular, ε) in an even number of moves. Hence, $e(M') \in$ EVEN, and so M_2 accepts $e(M')$. Conclude that M_1 accepts x.

Conversely, suppose $x \notin SA$. Either x does not encode a TM, in which case M_1 rejects x out of hand, or $x = e(M)$, where M is a non-self-accepting TM. If M rejects $e(M)$, then M' rejects *every* input, and so $e(M') \notin$ EVEN. If M runs forever on $e(M)$, then M' runs forever on every input, and again $e(M') \notin$ EVEN. Either way, M_2 will reject $e(M')$, and so M_1 will reject x.

Conclude that M_1 decides SA, which is impossible; hence M_2 cannot exist, and so EVEN is not recursive.

(b) We claim that the language

$$FOREVER = \{x \mid x = e(M), M \text{ is a TM}, M \text{ runs forever on some input}\}$$

is *not* recursive. We will perform a reduction from the nonrecursive language

$$HALTS = \{x \mid x = e(M)e(w), M \text{ is a TM}, M \text{ halts on input } w\}$$

Let M_2 be a TM deciding FOREVER. We will construct a TM M_1 deciding HALTS. On input x, M_1 first checks whether $x = e(M)e(w)$ for some Turing machine M and string w; if not, M_1 rejects x. Otherwise, M_1 constructs the following TM M':

- On input y, M' first simulates M on w.
- Once M halts, M' then accepts y.

Finally, M_1 simulates M_2 on $e(M')$ and **rejects/accepts** x iff M_2 **accepts/rejects** $e(M')$.

We claim that M_1 decides HALTS. First, suppose $x \in HALTS$. Then $x = e(M)e(w)$, so it passes the initial test of M_1. Moreover, M halts on input w, so M' will eventually halt (and accept) on every input y. Hence, $e(M') \notin FOREVER$. Conclude that M_2 rejects $e(M')$ and so M_1 accepts x.

Conversely, suppose that $x \notin HALTS$. If $x \neq e(M)e(w)$ for any TM M and string w, M_1 will reject x out of hand. Otherwise, $x = e(M)e(w)$, where TM M does not halt on input w. Hence, M' will never halt for any input, and so $e(M') \in FOREVER$. Conclude that M_2 accepts $e(M')$ and so M_1 rejects x.

We conclude that M_1 decides HALTS, which is impossible; hence M_2 cannot exist, and so FOREVER is not recursive.