Harvard University
Computer Science 121

Problem Set 6

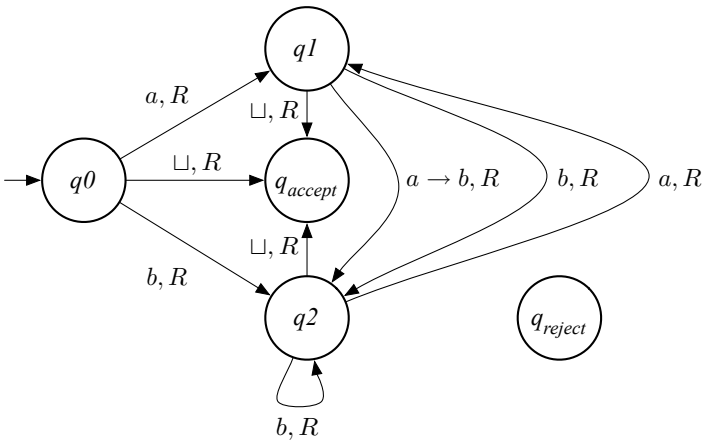
Due Wednesday, October 30, 2013 at 11:59 PM.
Submit your solutions electronically to cs121+ps6@seas.harvard.edu with "ps6 submission" in the subject line. The solutions to Parts A and B should be attached as separate PDF files, called lastname+ps6a.pdf and lastname+ps6b.pdf.
Late problem sets may be turned in until Friday, November 1, 2013 at 11:59 PM with a 20% penalty.
See syllabus for collaboration policy.

PART A (Graded by Louis and Francisco)

Note: You may use implementation-level descriptions (Lecture 14, slide 14) for this part of the problem set unless the problem specifies otherwise.

PROBLEM 1 (3+3 points)

Consider the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where $Q = \{q_0, q_1, q_2, q_{accept}, q_{reject}\}$, $\Sigma = \{a, b\}$, $\Gamma = \Sigma \cup \{\sqcup\}$, and δ is described by this state diagram:



(A) Give the sequence of configurations when M is run on input *aabbaaaa*.
(B) Describe informally how M transforms an arbitrary input string.

(A) $q_0$aabbaaaa
a$q_1$abbaaaa
ab$q_2$bbaaaa
abb$q_2$baaaa
abbb$q_2$aaaa
abbba$q_1$aaa

abbbab$q_2$aa
abbbabaq_1a
abbbabab$q_2\sqcup$
abbbabab$\sqcup q_{accept}$

(B) M sweeps left to right replacing the second in any pair of consecutive as with a b. This ensures that every a is followed by a b in the string left on the tape. Note that changes made during the sweep affect the remainder of the computation.

PROBLEM 2 (5+5 points)

(A) Prove that the decidable languages are closed under union.
(B) Prove that the Turing-recognizable languages are closed under intersection.

(A) Suppose L_1 and L_2 are decidable languages. Then there exist Turing machines M_1 and M_2 that decide them. We can construct a 2-tape Turing machine M which, on input w, copies w from the first tape to the second tape, repositions both heads at the start of the tapes, and then simulates M_1 on the first tape. If M_1 accepts, then M accepts. If M_1 rejects, then M simulates M_2 on the second tape. If M_2 accepts, then M accepts. Otherwise, M rejects.

Claim: M decides $L_1 \cup L_2$. Suppose $w \in L_1 \cup L_2$. Then either $w \in L_1$ and M_1 accepts w or $w \in L_2$ and M_2 accepts w or both, by definition of union. This implies that one of M_1 or M_2 will accept w. By the construction of M , M will therefore accept w. Conversely, suppose $w \notin L_1 \cup L_2$. Then by definition of union, $w \notin L_1$ and $w \notin L_2$ and M_1 and M_2 will both reject w. By construction of M, M will therefore reject w.

(B) Suppose L_1 and L_2 are Turing-recognizable languages. Then there exist Turing machines M_1 and M_2 that recognize them. We can construct a 2-tape Turing machine M which, on input w, copies w from the first tape to the second tape, repositions both heads at the start of the tapes, and then simulates the first step of M_1 on the first tape, simulates the first step of M_2 on the second tape, simulates the second step of M_1 on the first tape, simulates the second step of M_2 on the second tape, and so on. If one of the computations halts then computation continues on the other tape at every step. When computation halts on both tapes, M accepts if and only if both machines accepted.

Claim: M decides $L_1 \cap L_2$. Suppose $w \in L_1 \cap L_2$. Then by definition of intersection, M_1 and M_2 will both accept w. And hence by construction, M will also accept w. Conversely, suppose that $w \notin L_1 \cap L_2$. Then by definition of intersection, one of M_1 or M_2 will fail to accept w. Hence, by construction, M will also fail to accept w.

PROBLEM 3 (7+7 points)

(A) Let $L = \{\langle D, k\rangle : D \text{ is a DFA that accepts exactly } k \text{ strings, where } k \in \mathbb{N} \cup \{\infty\}\}$. Show that L is decidable. (*Hint:* Show how to find a p such that $\mathcal{L}(D)$ is infinite iff $\mathcal{L}(D)$ contains a string whose length is between p and $2p$.)

(B) Let $L = \{\langle M\rangle : M \text{ is a TM that accepts at least one string in } \Sigma^*\}$. Show that L is recognizable.

(A) From the pumping lemma (Sipser p. 78), we know that for p = number of states in a DFA, if the DFA accepts any string of length at least p, it must accept infinitely many strings.

The following is a decider for L:
D = On input $\langle D, k\rangle$:

1. Let p be the number of states of D. Simulate D on all strings w, where $p \leq |w| \leq 2p$.
2. If D accepts any string of that length, the language is infinite, so accept if $k = \infty$, and reject if $k \neq \infty$.
3. If D rejects all strings of that length, then simulate D on all strings of length less than p. Count the number of accepting strings, and let that number be c. If $k = c$, then accept.
4. Otherwise, reject.

Now if D accepts some string of length greater or equal to p, then D accepts infinitely many strings. If D rejects all strings of length between p and $2p$, then by the pumping lemma, we can deduce that D rejects all string of length greater or equal to p. (If D accepts w and $|w| \geq 2p$, then, letting w be the shortest string accepted by D of length at least p, we see that $|w| \leq 2p$ because otherwise we can pump down w to get a shorter string.)

Note: It was also possible to do cycle detection on the DFA graph directly to tell whether the language was infinite. If you do that, you need to be careful that you only look for reachable cycles from which it is also possible to reach an accept state.

(B) We'll construct a recognizer R for L. Given input $\langle M\rangle$, R will simulate M on all strings in Σ^* in parallel by using dovetailing, accepting if any of those computations accept. Thus R will eventually accept $\langle M\rangle$ if M accepts any string, and will run forever otherwise.

Note: remember that if you just run an arbitrary TM M on some string, it may not halt. Therefore dovetailing is crucial here–if we just run M on all strings in Σ^* in order, it may loop on the first one and never get to the others. As a reminder, dovetailing computations of M over some set of strings means running the M for 1 step on the first string, then for 2 steps on the first two strings, and so on. Eventually, this will get to all strings and all numbers of steps, so if M accepts any string w after k steps, this method will terminate.