Theory of Computation

Fall 2012, Final Exam. (Solutions)

Due: January 7, 2013

1. (20 pts) Consider the following classes of languages numbered (1)-(7)
   (1) Empty (2) Finite (3) Regular (4) Context-free
   (5) Recursive (6) Recursively enumerable (r.e.) (7) All languages
   For each of the following languages specify the *lowest-numbered* class to which it *surely* belongs; no need to provide any justification for your answers. For example, for a context-free language L that is not regular, the right number is 4. Similarly, suppose a language L is recursively enumerable, although it could possibly be recursive, the available information does not guarantee that it is recursive, then the right answer is 6.

   (a) The complement of a non-r.e. language.
   Answer: 7

   (b) The complement of a language in NP (i.e., nondeterministic polynomial time).
   Answer: 5

   (c) The complement of a context-free language.
   Answer: 5

   (d) The intersection of a recursive language and a language that is not r.e..
   Answer: 7

   (e) The intersection of a recursive language and an r.e. language.
   Answer: 6

   (f) $L = \{a^i b^j c^k d^l \mid i = k \text{ and } j = l\}$
   Answer: 5

   (g) $L = \{a^i b^j c^k d^l \mid i = l \text{ and } j = k\}$
   Answer: 4

   (h) $L = \{a^i b^j c^k d^l \mid i \times j \times k \times l \text{ is divisible by } 5\}$
   Answer: 3

   (i) The complement of the language generated by the following grammar:
   $S \to a_i S w_i$; $S \to \epsilon$, where $a_i \in \Sigma$, $w_i \in \Sigma^*$, $1 \le i \le k$, for some k.
   Answer: 4

   (j) The complement of $\{a^i b^i c^i \mid i \ge 0\}$ $(\Sigma = \{a, b, c\})$.
   Answer: 4

2. (9 pts) There are three ways to define acceptance of a PDA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Fill in the final instantaneous description for each of the following:

   (a) $\{w \mid A$ accepts w by *Final State*$\} = \{w \mid [q_0, w, Z_0] \vdash^* \underline{[f, \epsilon, \alpha], f \in F, \alpha \in \Gamma^*}\}$

   (b) $\{w \mid A$ accepts w by *Empty Stack*$\} = \{w \mid [q_0, w, Z_0] \vdash^* \underline{[q, \epsilon, \epsilon], q \in Q}\}$

   (c) $\{w \mid A$ accepts w by *Final State and Empty Stack*$\} = \{w \mid [q_0, w, Z_0] \vdash^* \underline{[f, \epsilon, \epsilon], f \in F}\}$

3. (15 pts) Let $S = \{\langle M \rangle \mid M$ is a DFA that accepts w^R whenever it accepts $w\}$. Show that S is decidable. (w^R is the reversal of w.)
   Answer (Proof sketch): If M is a DFA, let M^R be the DFA that accepts the reverse of all the strings that M accepts. Such a DFA can be constructed by reverting each of the transitions in the DFA and by switching the initial and the final states. Notice that $\langle M \rangle \in S \Leftrightarrow L(M) = L(M^R)$ $\Leftrightarrow M, M^R$ are equivalent. The membership of $\langle M \rangle$ can be checked by using the equivalence checking algorithm of DFA.

4. (15 pts) Let $T = \{\langle M \rangle \mid M$ is a Turing machine that accepts w^R whenever it accepts $w\}$. Show that T is undecidable. Do not use Rice's theorem. (Hint: Establish a reduction from the Halting Problem.)

Answer: Let $A_{TM} = \{\langle M, w\rangle \mid TM M \text{ accepts } w\}$

We reduce A_{TM} to T. It follows that T is undecidable. Given $\langle M, w\rangle$, we need M' such that $\langle M, w\rangle \in A_{\mathsf{TM}} \iff \langle M'\rangle \in T$. The description of M' is as follows.

(a) On input x, check if $x = \mathtt{01}$ or $\mathtt{10}$

(b) If $x \neq \mathtt{01}$ and $x \neq \mathtt{10}$, reject.

(c) If $x = \mathtt{01}$, accept.

(d) If $x = \mathtt{10}$, simulate M on w. Accept x if M accepts w, reject x if M rejects w.

It is easy to see that $L(M') = \{\mathtt{01}, \mathtt{10}\}$ if M accepts w and $L(M') = \{\mathtt{01}\}$ if M does not accept w. So the above condition for reduction is satisfied, and hence $A_{\mathsf{TM}} \leq_m T$.

5. (10 pts) Prove or disprove the following statement ($\leq_m$ denote the many-one reduction):

$$L \text{ is recursive iff } L \leq_m 0^*1^*$$

Answer:
($\Rightarrow$) If L is recursive (i.e. decidable), then we have a decider for L (i.e., a TM that always halts which decides L). We can modify the decider to get a reduction to the language 0^*1^* The decider takes a string w as input, and if $w \in L$, it accepts. Else it rejects. Now modify the decider to output the string 01 on acceptance and the string 10 on rejection. This machine suffices as a reduction machine. The output string is in 0^*1^* iff $w \in L$.
($\Leftarrow$) If $L \leq_m 0^*1^*$, then we can use the reduction to reduce to the language 0^*1^*. This language is regular and hence decidable. So L is decidable, by reduction to a decidable language.

6. (15 pts) Prove that the following language is not recursive:

$$L_{comp} = \{\langle M_1, M_2\rangle \mid L(M_1) = \overline{L(M_2)}\}$$

Note that the words in L_{comp} encode two Turing machines M_1 and M_2 such that the language of M_1 is the complement of the language of M_2. (Hint: To prove this result you may assume that the language $L_{all} = \{\langle M\rangle \mid L(M) = \Sigma^*\}$ is not recursive.)

Solution: We give a reduction from L_{all} to L_{comp}. Since L_{all} is known to be non-recursive, it follows that L_{comp} is non-recursive.

The reduction f works as follows: Given the machine M which is an instance of L_{all}, the reduction f creates an instance of L_{comp} by setting $M_1 = M$ and choosing M_2 as a Turing machine with the empty language. We can choose M_2 to be any Turing machine without any final states, e.g., a Turing with only an initial state, no final states and no transitions. The reduction f is easily computable by a halting Turing machine. We observe that $\overline{L(M_2)} = \Sigma^*$ so $L(M_1) = \overline{L(M_2)}$ if and only if $L(M_1) = \Sigma^*$. It follows that $< M_1, M_2 >$ is in L_{comp} if and only if $M \in L_{all}$.

7. (10 pts) Suppose that

- $A \subseteq \Sigma^*$ is NP-complete,
- $B \subseteq \Sigma^*$ is in P,
- $A \cap B = \emptyset$;, and
- $A \cup B \neq \Sigma^*$

Prove that $A \cup B$ is NP-complete. (You need to prove that (1) $A \cup B$ is in NP, and (2) $A \cup B$ is NP-hard.)
Answer:
(Proof of (1)) Let M_A and M_B be TMs operating in NP and P, respectively, accepting A and B, respectively. For every $x \in \Sigma^*$, let M be a TM which first nondeterministically chooses M_A or M_B to simulate on input x, and accepts accordingly. Clearly M, operating in NP, accepts $A \cup B$.
(Proof of (2)) (Simplified version) From $A \cup B \neq \Sigma^*$, we let x be a word in $\Sigma^* - (A \cup B)$. Since A is NP-complete, every language in L in NP is reducible to A through a polynomial time mapping say σ_L such that $w \in L$ iff $\sigma_L(w) \in A$. We now define a polynomial time mapping δ_L from L to $A \cup B$ such that $w \in L$ iff $\delta_L(w) \in (A \cup B)$. δ is defined as follows:

If $\delta_L(w) \notin B$, then $\delta_L(w) = \sigma_L(w)$;
if $\delta_L(w) \in B$, then $\delta_L(w) = x$, which is not in $A \cup B$. Note that $B \cap A = \emptyset$.
Hence, $w \in L$ iff $\delta_L(w) \in (A \cup B)$, implying that $A \cup B$ is NP-hard.

8. (6 pts) Assuming that the addition function $x + y$ and the multiplication function $x \cdot y$ are primitive recursive, prove that the following function is primitive recursive:
$x \ominus y = x - y$ if $x \geq y$; $\quad x \ominus y = 0$ if $x < y$.
Answer: First note that the following function

$$P(x) = x - 1 \text{ if } x \geq 1; \quad P(x) = 0, \text{ if } x = 0$$

is primitive recursive since $P(x)$ can be defined as $P(0) = \mathbf{0}$, and $P(x+1) = \pi_2(P(x), x) = x$.
Then let $x \ominus y = T(x, y)$.

$$T(x, 0) = \pi_1(x); \quad T(x, y+1) = P \circ \pi_2(x, T(x, y), y)$$