Theory of Computation

Fall 2016, Final Exam. Solutions

January 9, 2017

1. (30 pts) True or False? Score = max{0, Right - $\frac{1}{2}$ Wrong}. No explanations are needed.

   (1) If L^* is decidable then L is decidable.
   False

   (2) It is decidable whether given a TM M and an input x, M enters some state more than 100 times.
   True

   (3) Every recursively enumerable language can be accepted by a TM whose head only moves to the right.
   False

   (4) The language $L = \{\langle M \rangle : L(M) \in NP\} \in NP$.
   False

   (5) If $L_1 \leq_p L_2$ and $L_2 \leq_p L_1$, then $L_1 = L_2$.
   False

   (6) The following problem is decidable: Given a TM M and a string w, does M accept w within $|w|$ steps?
   True

   (7) For any $k > 1$, there is no language that is decided by a TM with k tapes, but is undecidable by any TM having $k - 1$ tapes.
   True

   (8) $\{\langle G, D \rangle : G$ is a CFG, D is a DFA, and $L(G) \subseteq L(D)\}$ is decidable.
   True

   (9) If $L \subseteq \{0\}^*$ then L is decidable.
   False

   (10) $CFL \subseteq P$.
   True

   (11) If A and B are in NP, then $A \cdot B$ is also in NP.
   True

   (12) The set of all r.e. languages is countable.
   True

   (13) The language EQPDA $= \{\langle C, D \rangle : C$ and D are PDAs with $L(C) = L(D)\}$ is Turing-decidable.
   False

   (14) Recursive languages are closed under the Kleene star (i.e., if L is recursive, so is L^*).
   True

   (15) PCP over the alphabet $\{0, 1\}$ is decidable.
   False

2. (12 pts) Recall that, if $w = a_1 \cdots a_n \in \Sigma^n$ is a string, $w^R = a_n \cdots a_1$ is the "reversal" of w. If $L \subseteq \Sigma^*$ is a language, we let $L^R = \{w^R : w \in L\}$. Let $A_{TM} = \{\langle M, w \rangle : M$ accepts $w\}$ and $A_R = \{\langle M \rangle : L(M) = (L(M))^R\}$, here M denotes a TM.
   Prove the (1) (6 pts) $A_{TM} \leq_m A_R$, and (2) (6 pts) $\overline{A_{TM}} \leq_m A_R$.
   Solution:
   • ($A_{TM} \leq_m A$). We must map $\langle M, w \rangle$ into $\langle M' \rangle$ such that M accepts w iff $L(M') = (L(M'))^R$. The mapping must be Turing-computable. So let M' on input x behave as follows:

   If $x = 01$ then accept.

   Run M on w

   If M accepts w, then accept

   If M rejects w, then reject

Now if M accepts w then $L(M') = \Sigma^*$ so $L(M') = (L(M'))^R$; while if M does not accept w then $L(M') = \{01\}$ so $L(M') \neq (L(M'))^R$.

• $(\overline{A_{TM}} \leq_m A_R)$. We must map $\langle M, w\rangle$ into $\langle M'\rangle$ such that (a) if M does not accept w then $L(M') = (L(M'))^R$, and (b) if M does accept w then $L(M') \neq (L(M'))^R$. The mapping must be Turing-computable. So let M' on input x behave as follows:

Run M on w

If M accepts w and $x = 01$, then accept

Reject

Now if M does not accepts w then $L(M') = \emptyset$; so $L(M') = (L(M'))^R$; while if M does accept w then $L(M') = \{01\}$ so $L(M') \neq (L(M'))^R$.

3. (10 pts) Prove that bounded halting $BH = \{(M, x, 1^k)$: NTM M halts on x in k steps$\}$ is NP-complete (i.e., $BH \in NP$ (5 pts) and BH is NP-hard (5 pts)).
Solution: BH is in NP - the membership certificate is the binary string of length k which corresponds to the accepting computation. Given that string, it is easy to verify whether the computation is accepting deterministically in polynomial time (by simulating M on x for k steps using a universal TM, slightly modified).

   BH is NP-hard. Let L be an NP language. By definition, there exists a NTM M_L and a polynomial P_L such that M_L accepts any string of length n in $P_L(n)$ steps. Given a string x (an instance of L), a corresponding instance of BH is the triple $(M_L, x, 1^{P_L(|x|)})$. It is easy to verify that the transformation from x to the corresponding triple can be done by a TM in polynomial time - it suffices to copy M_L (a constant string), copy x from the input and output $P_L(|x|)$ ones (notice that polynomials are polynomial-time computable).

4. (10 pts) Let $F_1, F_2, F_3, \cdots$ be an effective enumeration of all primitive recursive functions. Answer the following two questions: (a) (4 pts) Is the function $f(i, n) = F_i(n)$ a total TM-computable function? Why? (b)(6 pts) Is $f(i, n)$ a primitive recursive function? Why? You must justify your answers.
Solution: (a) Yes, since all primitive recursive functions are total TM-computable functions. (b) No. Let $f'(n) = f(n, n) + 1 (= F_n(n) + 1)$. Clearly, $f'(n) \neq F_i(n), \forall i \geq 1$; hence, $f'(n)$ is not primitive recursive. Therefore, $f(i, n)$ is not primitive recursive either.

5. (16 pts) Given a TM M (with only left/right moves and without ϵ transitions) and an input x, define the following two sets

   (a) $\mathit{ValComps}_{M,x} = \{w_1\#w_2\#w_3\#w_4\# \cdots w_n\#$:

   (b) $\mathit{ValComps}^R_{M,x} = \{w_1\#w_2^R\#w_3\#w_4^R\# \cdots w_n\#$: (if n is odd; $w_1\#w_2^R\#w_3\#w_4^R\# \cdots w_n^R\#$ otherwise)

   - $w_1 = q_0x$ is the initial ID,
   - w_n is an accepting ID, and
   - $w_i \to w_{i+1}, \forall 1 \leq i < n.\}$

   We also define $\mathit{ValComps}_M = (\bigcup_{x\in\Sigma^*} \mathit{ValComps}_{M,x})$, and $\mathit{ValComps}^R_M = (\bigcup_{x\in\Sigma^*} \mathit{ValComps}^R_{M,x})$. Among the following language classes (regular, CFL, co-CFL, context-sensitive, recursive, r.e., and co-r.e.), identify the smallest class each of $\mathit{ValComps}_{M,x}$, $\mathit{ValComps}^R_{M,x}$, $\mathit{ValComps}_M$, and $\mathit{ValComps}^R_M$ belongs. You need to justify your answer.
Solution:

   (a) $\mathit{ValComps}_{M,x}$ is regular since it is finite.

   (b) $\mathit{ValComps}^R_{M,x}$ is regular since it is finite.

   (c) $\mathit{ValComps}_M$ is a CSL since it can be accepted by a linear bounded automaton (LBA).

   (d) $\mathit{ValComps}^R_M$ is a co-CFL since its complement is CF. The key is the ability for a PDA to check, given $w_i\#w_{i+1}^R$ for some i, whether $w_i \not\to w_{i+1}$. The PDA accepting the complement of $\mathit{ValComps}^R_M$ is to check, given a string x, (1) if x is not of the form $w_1\#w_2^R\#w_3\#w_4^R\#\cdots\#$, then accept; (2) if $\exists i$, $w_i \not\to w_{i+1}$, then accept.

6. (10 pts) Let $A_1, A_2 \subseteq \Sigma^*$ be two r.e. languages such that $A_1 \cup A_2 = \Sigma^*$ and $A_1 \cap A_2 \neq \emptyset$. Prove that $A_1 \leq_m (A_1 \cap A_2)$.
Solution: Let M_1 be a TM recognizing A_1 and M_2 a TM recognizing A_2. Further, since we know that $A_1 \cap A_2 \neq \emptyset$, let y be some string in $A_1 \cap A_2$. We describe a reduction f such that $x \in A_1$ iff $f(x) \in A_1 \cap A_2$ as follows:
On input x run the computations of M_1 and M_2 on x "in parallel", halting when either M_1 or M_2 halts and accepts x. if M_1 accepts x then output y else output x.

Notice that since $A_1 \cup A_2 = \Sigma^*$, we know either M_1 or M_2 must accept x; so the computation will definitely halt. Now, if when running in parallel M_1 and M_2 we find that $x \in A_1$ then $f(x) = y \in A_1 \cap A_2$. On the other hand, if we find that $x \in A_2$ then $f(x) = x \in A_1 \cap A_2$ if and only if $x \in A_1$. Thus, either way, $x \in A_1$ iff $f(x) \in A_1 \cap A_2$.

7. (7 pts) Based on what we learned in class about various complexity classes, fill in each of the following blanks with $\subset$, $\subseteq$ or $=$.
$L \subseteq NL = coNL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$

8. (5 pts) Prove that $f(x, y) = x^y$ is primitive recursive. You may assume that the multiplication function $g(x, y) = x \cdot y$ is primitive recursive.
Solution:

$$f(x, 0) = S(Z(x)) \quad (= 1)$$
$$f(x, S(y)) = g(x,\ f(x, y),\ y), \text{ where } g(x_1, x_2, x_3) = mul(\pi_1(x_1, x_2, x_3), \pi_2(x_1, x_2, x_3))$$