Theory of Computation

Fall 2024, Final Exam. (Solutions)

June 4, 2024

1. (50 pts) True or False? Mark O for **True**, and $\times$ for **False**. Score = max{0, Right - $\frac{1}{2}$ Wrong}. No explanations are needed. On the answer sheet, draw the following table and fill in O or $\times$.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
×	O	×	O	O	×	O	×	×	O	×	O	O	O	O	×	O	O	×	O	O	O	O	×	O

We write $\leq_p$ for "polynomial-time reduction"; $\leq_m$ for "mapping reduction"; P for "polynomial time"; NP for "nondeterministic polynomial time"; DTM for "deterministic Turing machine"; TM for "Turing machine" (possibly nondeterministic).

(1) $\{\langle G, H\rangle \mid$ graphs G and H are isomorphic$\}$ is NP-complete.
False. GI is in NP, but not known to be NP-complete. See Class Notes.

(2) $\{\langle F, x\rangle \mid F$ is a 3-CNF which evaluates to true on truth assignment $x\}$ is in P.
True. Once a truth assignment x is given, checking the value of F is easy (in P).

(3) $\{\langle M, w\rangle \mid M$ is a DTM that does not accept input $w\}$ is Turing-recognizable.
False. The language is $\overline{A_{TM}}$, which is not Turing-recognizable.

(4) $\{\langle M, w\rangle \mid M$ is a DTM that accepts w in at most $2^{|w|}$ steps$\}$ is not in P.
True. $P \neq EXPTIME$ - Time Hierarchy Theorem.

(5) If $A \leq_p B$ and B is in PSPACE, then A is in PSPACE.
True. Property of $\leq_p$.

(6) Recursive languages are closed under homomorphism.
False. $\{\langle M, w, (\#)^n\rangle \mid M \textit{ acepts } w \textit{ in } \leq n \textit{ steps}\}$ is recursive. However, if the homomorphism maps $\#$ to ϵ and leave all the other symbols intact, the resulting language becomes $\{\langle M, w\rangle \mid M \textit{ acepts } w\} = A_{TM}$.

(7) If $L \leq_p \{0^n1^n \mid n \geq 0\}$, then L is in P.
True. $\{0^n1^n \mid n \geq 0\}$ is in P.

(8) $REGULAR_{TM} = \{\langle M\rangle \mid$ the language recognized by Turing machine M is regular$\}$ is Turing-recognizable but not Turing-decidable.
False. The language is not Turing-recognizable. See Class Note (Chapter-4, p.53).

(9) It is a theorem that $NP \cap co\text{-}NP = P$.
False. It is not known whether $NP \cap co\text{-}NP = P$.

(10) If L is in NP, so is L^*.
True. Given x, guess $x = x_1x_2...x_n$, and check $\forall i, x_i \in L(M)$.

(11) All decidable languages are NP-hard.
False. 0^* is a decidable language, which is clearly not NP-hard.

(12) If L_1 and L_2 are not recursive, it is possible that $L_1 \cup L_2$ is recursive.
True. $A_{TM} \cup \overline{A_{TM}} = \Sigma^*$.

(13) $\{\langle M, w\rangle \mid$ DTM M moves right exactly twice on input $w\}$ is recursive.
True. The TM can only reads from (at most) the first three input symbols.

(14) The set of Turing-recognizable languages is countably infinite.
True. Each corresponds to a TM. The set of TMs is countably infinite.

(15) If there is a DTM operating in $DSPACE(\log n)$ to check whether or not two vertices in a directed graph are connected, then $DSPACE(\log n) = NSPACE(\log n)$.
True. Graph reachability problem is complete for $NSPACE(\log n)$. If the hardest problem in $NSPACE(\log n)$ is also in $DSPACE(\log n)$, then $DSPACE(\log n) = NSPACE(\log n)$.

(16) There is a language in BPP that is not in $PSPACE$.
False. $BPP \subseteq PSPAcE$. See Class Notes.

(17) The class RP remains the same if the error probability is made 2^{-n} in the definition. (Here, as usual, n is the length of the input.)
True. See Class Notes.

(18) There is an algorithm that can take an undirected graph and two vertices s, t as input and output whether or not there is a path between s and t in $O(\log^2 n)$ deterministic space.
True. Graph reachability problem is complete for $NSPACE(\log n)$, which is in $DSPACE(\log^2 n)$ – Savitch's Theorem

(19) Every NP-hard language is recognizable by a Turing machine.
False. EQ_{TM} is also NP-hard.

(20) There is a function $f : \{0,1\}^* \rightarrow \{0,1\}$ that cannot be computed by any Turing machine.
True. The set of such functions is not countably infinite.

(21) Suppose L is TM recognizable but not TM decidable. Then any TM that recognizes L must fail to halt on an infinite number of strings.
True. If not (i.e., if the set if finite), we can design a decider for L in the following way: first compare input to the elements of that finite set and if it is there reject. Otherwise simulate recognizer of L on the input (always halt) and return what it returns.

(22) $\{ \langle M \rangle \mid M$ is a DTM and if we start M with a blank input tape, then it will finally write some non-blank symbol on its tape.$\}$ is decidable.
True. If it the TM never prints a non-blank symbol, after at least $|Q| + 1$ steps (Q is the set of states of the TM), either the TM halts or a state would repeat.

(23) Given two context-free languages L_1 and L_2, then $L_1 \cap L_2$ is Turing decidable.
True. CFLs are recursive, and recursive languages are closed under intersection.

(24) PCP instance $\{[\frac{a}{c}], [\frac{a}{aa}], [\frac{cba}{a}]\}$ has a match (i.e., solution).
False. Easy observation.

(25) $co\text{-}NP \subseteq IP$, where IP stands for *interactive proof* systems.
True. See Class Notes.

2. (10 pts) Let $A_1, A_2 \subseteq \{0,1\}^*$ be Turing-recognizable languages such that $A_1 \cup A_2 = \{0,1\}^*$ and $A_1 \cap A_2 \neq \emptyset$. Prove that $A_1 \leq_m (A_1 \cap A_2)$.
(Hint: find a reduction f such that $x \in A_1$ iff $f(x) \in A_1 \cap A_2$. To find f, simulate M_1 and M_2 (which accept A_1 and A_2, respectively) in parallel. Then ...)
Solution: Let $y \in A_1 \cap A_2$. Consider the following mapping f:
On input x, simulate M_1 and M_2 in parallel, one of the following two must hold (because $A_1 \cup A_2 = \{0,1\}^*$):

- If M_1 accepts x, then $x \in A_1$, maps x to y, i.e. , $f(x) = y \in A_1 \cap A_2$;
- If M_2 accepts x, then $x \in A_2$, maps x to x (i.e., $f(x) = x$). Note that $x \in A_1$ iff $x \in A_1 \cap A_2$ since we already know $x \in A_2$.

3. (10 pts) Suppose we want to prove that $ODD_{TM} = \{\langle M \rangle \mid M$ is a TM and $L(M)$ does not contain any string of odd length$\}$ is undecidable. Let us reduce A_{TM} to ODD_{TM}. In other words, given a decider R for ODD_{TM}, let us show how to build a decider D for A_{TM}.

On input $\langle M, w \rangle$
Construct TM $N_{M,w}$ that works as follows:
(1) on input x, $N_{M,w}$ | ① (5 pts) .... |
Run R on $N_{M,w}$ | ②(5 pts).......................... |

- (**Question**) Complete the reduction by filling in missing details for the two blanks ① and ② in the above.

Solution:

① Simulate M on w; accept x if M accepts w.
② Accept if R rejects; reject if R accepts.

Note that $L(N_{M,w}) = \Sigma^*$ or $\emptyset$.

- If $L(N_{M,w}) = \Sigma^*$, $R(N_{M,w})$ rejects, meaning that M accepts w;
- If $L(N_{M,w}) = \emptyset$, $R(N_{M,w})$ accepts, meaning that M does not accept w.

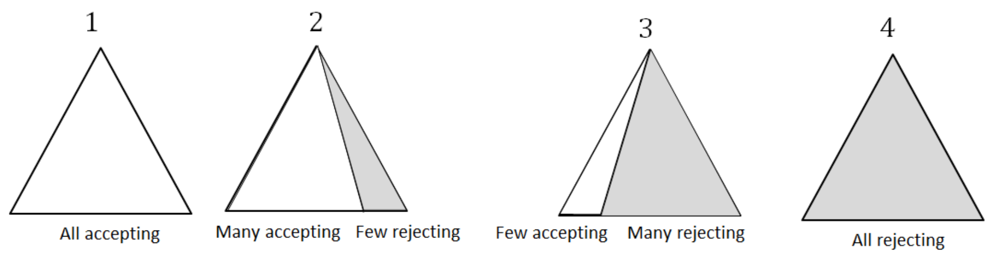
4. (10 pts) Assume that L_1 is NP-complete and $\overline{L_1} \in NP$. Prove that for all $L \in NP$, it must be the case that $\overline{L} \in NP$. Here $\overline{L}$ denotes the complement of L.

Solution:

- L_1 is NP-complete $\Rightarrow \forall L \in NP,\ L \leq_p L_1$ – Definition of NP-hardness
- $L \leq_p L_1 \Rightarrow \overline{L} \leq_p \overline{L_1}$ – Simple property of $\leq_p$
- $\overline{L} \leq_p \overline{L_1}$ and $\overline{L_1} \in NP \Rightarrow \overline{L} \in NP$.

5. (10 pts) Consider the four types of computations (labelled 1-4) shown in the figure. Consider the following table in which each column is associated with a complexity class C. For a machine M, $L(M) \in C$, and an input w, we put i in *(accepting, C)* entry if type-i is an accepting computation for M on w in C. Likewise, we put j in (rejecting, C) entry if type-j is a rejecting computation for M on w in C. Fill in the blank entries with numbers from $\{1, 2, 3, 4\}$, or 0 if none applies. Note that you may need to put in multiple numbers in each entry.

	co-NP	RP	co-RP	ZPP	BPP
accepting	1	1,2	1	1	1,2
rejecting	2,3,4	4	3,4	4	3,4



6. (10 pts) Suppose the following is a fragment of a computation of a DTM M on some input.

$$c_0 \vdash \cdots \overbrace{1101q_a0110}^{c_1} \vdash \overbrace{110q_b11110}^{c_2} \vdash \overbrace{1100q_b1110}^{c_3} \vdash \cdots$$

(a) (4 pts) What are the two transitions executed from c_1 to c_2, and from c_2 to c_3? Write down the transition in the standard form (i.e., $\delta(...) = (...)$).

(b) (2 pts) What is the configuration immediately following (i.e., after) c_3?

(c) (4 pts) Explain how to check $1101q_a0110 \vdash (110q_b11110)^R$ using a pushdown automaton (PDA). Here R denotes the "reversal" of a string. Give your explanation in English or Chinese.

Solution:

(a) $\delta(q_a, 0) = (q_b, 1, L)$; $\delta(q_b, 1) = (q_b, 0, R)$

(b) $11000q_b110$

(c)
- While reading $1101q_a0110$, push $1, 1, 0, 1, q_a, 0, 1, 1, 0$ onto the stack. After that the stack contains $1101q_a0110$ (top).
- After reading $\vdash$, the PDA pops $0, 1, 1$ while reading the first three symbols of $(110q_b11110)^R = \underline{011}11q_b011$
- the PDA nondeterministically reads the next three symbols $1, 1, q_b$ of $011\underline{11q_b}011$ and stores them in the finite state, while popping three symbols $0, q_a, 1$ from the stack and stores them in the final state,
- the PDA checks the transition function of the Turing machine to make sure that $1q_a0 \vdash 11q_b$ is a legal move.
- finally, the PDA pops $0, 1, 1$ while reading $0, 1, 1$.