Indian Institute of Technology Kharagpur
Department of Computer Science and Engineering

Foundations of Computing Science (CS60005) **Autumn Semester, 2022-2023**

End-Semester Examination **Date: 18-Nov-2022, 2:00 PM – 5:00 PM** **Marks: 100**

Instructions:

- Write your answers in the answer booklet provided to you in the examination hall.
- There are a total of FIVE questions, each having 20 marks.
- Answer ALL the questions (or as many as you can) mentioning the question numbers clearly.
- Be brief and precise. Write the answers for all parts of a question together.
- State any results you use or assumptions you make.
- Write all the proofs/deductions in mathematically/logically precise language.
- Sketchy proofs or claims without reasoning receive no credit.

1. Only write down all the correct choice(s) / option(s) for each of the following questions. $10 \times 2 = 20$

   Note: *No justification is required for your given option(s); but a penalty of $\frac{1}{2}$ per question will be deducted for guessing wrong, including not making all choices (in case of multiple correct options).*

   (a) The propositional logic statement $\big[(p \vee q \vee r) \rightarrow s\big]$ is equivalent to which one/more of the following?

   (A) $\neg s \rightarrow (p \vee q \vee r)$
   (B) $\neg s \rightarrow (p \wedge q \wedge r)$
   (C) $(p \rightarrow s) \wedge (q \rightarrow s) \wedge (r \rightarrow s)$
   (D) $(p \rightarrow s) \vee (q \rightarrow s) \vee (r \rightarrow s)$

   Solution: (C)

   (b) According to political experts, "A person who is a radical (*radical*) is elected (*elected*) if (s)he is conservative (*conservative*), but otherwise is not elected". Which of the following is/are the correct logical representation(s) of the above statement made by political experts?

   (A) $\big(radical \wedge elected\big) \leftrightarrow conservative$
   (B) $radical \rightarrow \big(elected \leftrightarrow conservative\big)$
   (C) $radical \rightarrow \big((conservative \rightarrow elected) \vee \neg elected\big)$
   (D) $\big(conservative \rightarrow (radical \wedge elected)\big) \vee \neg elected$

   Solution: (B)

   (c) Let P and Q are two predicates, and $F_1 \Leftrightarrow F_2$ symbolically denotes equivalence of two predicate logic formulas, F_1 and F_2 (that is, F_1 if and only if F_2). Which of the following is/are incorrect?

   (A) $\forall x \ \big[P(x) \vee Q(x)\big] \Leftrightarrow \forall x \ P(x) \vee \forall x \ Q(x)$
   (B) $\exists x \ \big[P(x) \vee Q(x)\big] \Leftrightarrow \exists x \ P(x) \vee \exists x \ Q(x)$
   (C) $\forall x \ \big[P(x) \wedge Q(x)\big] \Leftrightarrow \forall x \ P(x) \wedge \forall x \ Q(x)$
   (D) $\exists x \ \big[P(x) \wedge Q(x)\big] \Leftrightarrow \exists x \ P(x) \wedge \exists x \ Q(x)$

Solution: (A),(D)

(d) Let X, Y, Z be sets. Which of the following formulas is/are wrong?

(A) $(X \cap Y \cap Z) \setminus Z = \emptyset$
(B) $(X \cup Y \cup Z) \setminus Z = X \cup Y$
(C) $(X \cap Y) \setminus Z = (X \setminus Z) \cap (Y \setminus Z)$
(D) $(X \cup Y) \setminus Z = (X \setminus Z) \cup (Y \setminus Z)$

Solution: (B)

(e) Let $A = \{a, b, c, d\}$. How many binary operations f on A are there such that $f(a, b)$ is either c or d?

(A) 15^4
(B) 4^{15}
(C) 2×4^{15}
(D) 2×15^4

Solution: (C)

(f) Define a relation ρ on $\mathbb{Z}$ (the set of integers) such that $a \ \rho \ b$ if and only if either $a = b$ or $|a - b| > 1$. Which of the following statements about ρ is/are incorrect?

(A) ρ is reflexive
(B) ρ is symmetric
(C) ρ is antisymmetric
(D) ρ is transitive

Solution: (C),(D)

(g) Let $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Q}$ denotes the set of natural numbers, integers and rational numbers, respectively. Which of the following sets is/are countable?

(A) $\mathbb{N} \times \mathbb{N}$
(B) $\mathbb{Z} \times \mathbb{Q}$
(C) $\mathbb{Q} \setminus \mathbb{N}$
(D) $\mathbb{Q} \times \mathbb{Q}$

Solution: (A),(B),(C),(D)

(h) What is the inverse of an element a in the group $G = \{a \in \mathbb{R} \mid a > 0\}$ ($\mathbb{R}$ denotes the set of real numbers) under the operation $\odot$ defined by $a \odot b = a^{\ln b}$?

(A) $1/a$
(B) $1/e^{\ln a}$
(C) $e^{1/\ln a}$
(D) $1/\ln a$

Solution: (C)

(i) Consider the set $\mathbb{Z}$ of all integers with the following operations, where s and t are constant integers.

$$\begin{aligned} a \oplus b &= a + b + s, \\ a \odot b &= a + b + tab. \end{aligned}$$

Which of the following is/are a necessary and sufficient condition for $(\mathbb{Z}, \oplus, \odot)$ to be a ring?

(A) $st = 1$.
(B) $s = t = -1$.
(C) s and t can have any values.
(D) s and t can have any values with $t \neq 0$.

Solution: (A)

(j) Which of the following functions $f : \mathbb{Z} \to \mathbb{Z}$ ($\mathbb{Z}$ denotes the set of integers) is/are a homomorphism of the ring $(\mathbb{Z}, +, \cdot)$ to itself?

(A) $f(x) = 1$
(B) $f(x) = x$
(C) $f(x) = 2x$
(D) $f(x) = x^2$

Solution: (B)

2. For each of the following languages, identify whether the language is recursive, *r.e.* but not recursive or not *r.e.* For all languages defned below, $\mathcal{M}$ denotes a Turing machine. Justify your answer.

   (a) $\{\mathcal{M} \mid \mathcal{M}$ on input x moves its head more than 100 cells away from the left endmarker$\}$.

   Solution: *Recursive.*

   Suppose a TM $\mathcal{M}$ never moves its head more than 100 cells to the right of $\vdash$. Then there are only finitely many configurations possible for the TM corresponding to a combination of 101 possible positions of tape head, $|Q|$ possible states and $|\Gamma \setminus \{\vdash\}|$ many possibilities for each of the 100 cells to the right of $\vdash$. So the total number of configurations possible is $N = |Q| \times 101 \times (|\Gamma| - 1)^{100}$. Now simulate $\mathcal{M}$ for $N + 1$ steps. There are 3 possibilities: $\mathcal{M}$ halts before moving past 100-th cell; one of the configurations repeats in which case the TM loops; or $\mathcal{M}$ moves past 100-th cell. So we can decide in at most $N + 1$ steps whether $\mathcal{M}$ moves past the 100-th cell or not.

   (b) $\{\mathcal{M} \mid L(\mathcal{M})$ is recursive$\}$.

   Solution: *Not r.e.*

   Define a property $\mathcal{P}$ on *r.e.* sets as follows.

$$\mathcal{P}(A) = \begin{cases} \top & \text{if } A \text{ is recursive} \\ \bot & \text{otherwise} \end{cases}$$

   Consider the set $L = \{(\mathcal{M}, x) \mid \mathcal{M}$ halts on input x in less than 100 steps.$\}$. We can design a TM $\mathcal{N}$ that takes as input $(\mathcal{M}, x)$, simulates $\mathcal{M}$ on input x for 100 steps; accepts if $\mathcal{M}$ halts and rejects otherwise. $\mathcal{N}$ is total and hence L is recursive. In other words, $\mathcal{P}(L) = \top$. Now, $L \subseteq \mathsf{HP}$. But HP is *r.e.* but not recursive i.e., $\mathcal{P}(\mathsf{HP}) = \bot$. Hence $\mathcal{P}$ is a non-monotone property and by part II of Rice's theorem it is not semi-decidabble or equivalanetly not *r.e.* .

   (c) $\{\mathcal{M} \mid L(\mathcal{M})$ has atleast 100 states$\}$.

   Solution: *Recursive.*

   Check the description of the machine to see if the number of states is ≥ 100.

   (d) $\{\mathcal{M} \mid \mathcal{M}$ halts on all inputs of length less than 100$\}$.

   Solution: *r.e. but not recursive.*

   Denote the language by T_{100}. Define a TM $\mathcal{N}$, that on input a TM $\mathcal{M}$, does the following:

   - For each string w with $|w| < 100$, run $\mathcal{M}$ on w
   - If $\mathcal{M}$ halts on all such strings w, then accept and halt

   If $\mathcal{M} \in \mathsf{T}_{100}$, then it halts on all string of length less than 100 and so $\mathcal{N}$ accepts $\mathcal{M}$. Otherwise, $\mathcal{M}$ loops on some string w thus making $\mathcal{N}$ loop on $\mathcal{M}$. Hence $\mathcal{L}(\mathcal{N}) = \mathsf{T}_{100}$ establishing that T_{100} is recursively enumerable.

   To show that T_{100} is not recursive, we provide a reduction from HP. Let $(\mathcal{M}, x)$ be an instance of HP. Define a TM $\mathcal{N}$, that does the following on input y:

   - If $|w| >= 100$, then reject and halt (or accept or enter a trivial loop).
   - Otherwise, run cM on x.
   - Accept and halt if $\mathcal{M}$ halts on x.

   If $\mathcal{M}$ halts on x, then $\mathcal{N}$ halts on all inputs of length < 100. If $\mathcal{M}$ does not halt on x, then $\mathcal{N}$ loops on all inputs of length < 100. That is, $(\mathcal{M}, x) \in \mathsf{HP} \Leftrightarrow \mathcal{N} \in \mathsf{T}_{100}$.

$4 \times 5 = 20$

3. (a) Let ϕ be a Boolean formula in CNF. An assignment to the variables of ϕ is called *not-all-equal* if in each clause, at least one literal is assigned 1 and at least one literal is 0. Show that the following language is **NP**-Complete.

$$\mathsf{NESAT} = \{\phi \mid \phi \text{ is a CNF formula having a satisfying } \textit{not-all-equal} \text{ assignment}\}$$

   Assume no constants are allowed in the formulae. 10

   Solution: Clearly NESAT is in **NP**. We show that $\mathsf{SAT} \leq_p \mathsf{NESAT}$. Let $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ be an instance of SAT. Construct a formula ψ as follows: introduce $m+1$ new variables $w_1, \cdots, w_m, z$ and for each clause $c_i = x_1 \vee \cdots \vee x_k$ of ϕ (where x_i's are literals), include two clauses $c_{i,1} = x_1 \vee \cdots \vee x_{k-1} \vee w_i$ and $c_{i,2} = x_k \vee \neg w_i \vee z$ in ψ.

Note that if $f : \{u_1, \ldots, u_n\} \to \{0, 1\}$ is a not-all-equal satisfying assignment for a CNF formula ϕ over variables $u_1, \ldots, u_n$, then so is $\neg f$ ($\neg f$ just flips the assignment to each variable).

$\phi \in \mathsf{SAT} \implies \psi \in \mathsf{NESAT}$: If $\phi \in \mathsf{SAT}$, there is an assignment of truth values to $u_1, \ldots, u_n$ such that ϕ is true. We show that there exists a not-all-equal satisfying assignment for ψ. The assignment to $u_1, \ldots, u_n$ remains unchanged. Since each clause of ϕ is satisfied, so is $c_i = x_1 \vee \cdots \vee x_{k-1} \vee x_k$. If $x_1 \vee \cdots \vee x_{k-1}$ true, then $c_{i,1}$ is true; setting $w_i = 0, z = 0$ ensures that $c_{i,2}$ is also satisfied and hence ψ has a not-all-equal assignment. Otherwise, it must be the case that $x_k = 1$. Then setting $w_i = 1, z = 0$ leads to a not-all-equal assignment of ψ.

$\psi \in \mathsf{NESAT} \implies \phi \in \mathsf{SAT}$: If $\psi \in \mathsf{NESAT}$, i.e., there exists a satisfying not-all-equal assignment of variables $u_1, \ldots, u_n, w_1, \ldots, w_m, z$. We show that $\phi \in \mathsf{SAT}$ by exhibiting a satisfying assignment for ψ. Suppose that the not-all-equal assignment sets $z = 0$. Then the same assignemnt to $u_1, u_2, \ldots, u_n$ will satisfy ϕ – consider clause c_i; if $x_k = 1$ then c_i is satisfied; otherwise since $c_{i,2}$ has atleast one literal assigned 1, it must be the case that $\neg w_i = 1$, i.e., $w_i = 0$ and since $c_{i,1}$ is satisfied, $x_1 \vee \cdots \vee x_{k-1}$ is true and so c_i must be true. Now suppose that the not-all-equal assignment f for ψ sets $z = 1$, then $\neg f$ would still satisfy ψ and ensure $z = 0$. Use the same reasoning as above to show a satisfying assignment for ϕ.

□

(b) Let S be a finite set and $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ be a collection of subsets of S, for some $k > 0$. We say S is *2-colourable with respect to* $\mathcal{C}$ if we can colour each element of S in red or blue, such that every C_i contains at least one red element and at least one blue element. Show that the language

$$\mathsf{2COLOUR} = \{\langle S, \mathcal{C}\rangle \mid S \text{ is 2-colourable with respect to } \mathcal{C}\}$$

is **NP**-Complete. 10

Solution: Let $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, a CNF formula over n variables $u_1, \ldots, u_n$, be an instance of NESAT. We will show how to transform ϕ to a set S and a collection of its subsets $\mathcal{C}$ such that $\phi \in \mathsf{NESAT}$ iff $\langle S, \mathcal{C}\rangle \in \mathsf{2COLOUR}$. Define S to contain elements x_i, x_i' corresponding to literals $u_i, \neg u_i$ for $i \in [1, n]$. Add the subsets $B_i = \{x_i, x_i'\}$ (for $i \in [1, n]$) to $\mathcal{C}$. Further, include subsets C_j for $j = 1, \ldots, m$ constructed as follows: for every $i = 1, \ldots, n$, if clause c_j contains u_i, then $C_j \leftarrow C_j \cup \{x_j\}$; otherwise, if c_j contains $\neg u_i$, then $C_j \leftarrow C_j \cup \{x_j'\}$.

Suppose that ϕ has a not-all-equal satisfying assignment. Then we show a 2-colouring of S such that $\langle S, \mathcal{C}\rangle \in \mathsf{2COLOUR}$. If $u_i = 1$, then colour x_i red and x_i' blue. Otherwise, colour x_i blue and x_i' red. This would ensure that each B_i contains at least one blue element and one red element. Further, since each clause has at least one literal assigned 1 and at least one literal assigned 0, the red (resp. blue) elements in C_j would exactly correspond to the literals assigned 1 (resp. 0). Hence each C_j would contain at least one red element and at least one blue element.

For the other direction, suppose that S is 2-colourable with respect to $\mathcal{C}$. Fix a 2-colouring of S. For each $i \in [1, n]$, if x_i is coloured red, then it must be the case that x_i' is coloured blue (and vice-versa), since otherwise B_i would have both its entries coloured in red (resp. blue). If x_i is coloured red, then set $u_i = 1$; otherwise, set $u_i = 0$. To see that this is a not-all-equal satisfying assignment for ϕ, observe that C_j contains at least one element coloured in red and at least one element coloured in blue. The literals in clause c_j corresponding to red and blue elements would be assigned 1 and 0 respectively. This holds for each $j \in [1, m]$ and hence this is a satisfying not-all-equal assignment.

We have $\phi \in \mathsf{NESAT}$ iff $\langle S, \mathcal{C}\rangle \in \mathsf{2COLOUR}$. Further, NESAT is **NP**-complete and the above reduction runs in polynomial time. Therefore, 2COLOUR is **NP**-complete.

4. Answer true or false. Justify.

(a) If L_1, L_2 are **NP**-complete, then so is $L_1 \cup L_2$.

Solution: False.

We construct a counter example using the following two **NP**-Complete sets.

$$\mathsf{HAMPATH} = \{G \mid G \text{ is an undirected graph with a Hamiltonian path}\}$$

$$\mathsf{SAT} = \{\phi \mid \phi \text{ is a satisfiable CNF formula}\}$$

Let

$$L_1 = \{(G, \phi) \mid G \in \mathsf{HAMPATH}, \phi \text{ is any CNF formula}\},$$

and

$$L_2 = \{(G, \phi) \mid G \text{ is an undirected graph}, \phi \in \mathsf{SAT}\}.$$

Clearly L_1, L_2 are both **NP**-Complete, whereas

$$L_1 \cup L_2 = \{(G, \phi) \mid G \text{ is an undirected graph}, \phi \text{ is a CNF formula}\}$$

is not. It can be decided in poly-time.

(b) All **NP**-Hard problems are decidable.

Solution: False.

Halting problem is undecidable but is also **NP**-hard.

(c) $\mathbf{NSPACE}(2^n) \subseteq \mathbf{DSPACE}(4^n)$.

Solution: True.

Follows from Savitch's theorem.

(d) **PSPACE** is closed under Kleene star (i.e., if $L \in \mathbf{PSPACE}$, then $L^* \in \mathbf{PSPACE}$).

Solution: True.

Let $L \in \mathbf{PSPACE}$ and let $\mathcal{M}$ be a deterministic polynomial space machine deciding L. Observe that $x \in L^*$ iff one of the following holds: $x = \varepsilon$, $x \in L$, $x = wz$ such that $w \in L^*$ and $z \in L^*$. Let $x = x_1 x_2 \cdots x_n$ and let $x_{i,j}$ (for $1 \le i \le j \le n$) denote the substring $x_i x_{i+1} \cdots x_j$ of x. We will use dynamic programming to build a matrix $T = (t_{i,j})$ with $t_{i,j} = 1$ if $x_{i,j} \in L^*$ and 0 otherwise. Essentially we consider all substrings of x, starting with substrings of length 1 and ending with all substrings of length n. Below is the pseudocode:

```
Input: x = x_1 ··· x_n

  Initialise t_{i,j} to 0.
  If x = ε, then accept;
  For k = 1,...,n:
    For i = 1,...,n−k+1:
      j = i+k−1;
      Run M on x_{i,j};
      If M accepts, then set t_{i,j} = 1;
      Else
        For ℓ = i,...,j−1:
          If t_{i,ℓ} = 1, and t_{ℓ+1,j} = 1, then set t_{i,j} = 1;
  If t_{1,n} = 1, then accept; Else reject;
```

We now analyze the space complexity of the above algorithm. In each inner loop, we run $\mathcal{M}$ once which takes polynomial space (since $L \in \mathbf{PSPACE}$). Storing T requires $O(n^2)$-space and each execution of the loop requires constant number of cells to store the indices. Hence the algorithm runs in polynomial space.

Alternatively, using the fact that $\mathbf{PSPACE} = \mathbf{NPSPACE}$, we can define a poly-space NDTM that decides L^* that guesses an index i in x; checks whether $x_{1,i} \in L$ using $\mathcal{M}$ and then recursively checks whether $x_{i+1,n} \in L^*$. It is easy to verify that the first three run in polynomial space.

(e) $\mathbf{polyL} \neq \mathbf{polyNL}$, where $\mathbf{polyL} = \cup_{c>0}\mathbf{DSPACE}(\log^c n)$, $\mathbf{polyNL} = \cup_{c>0}\mathbf{NSPACE}(\log^c n)$.

Solution: False.

The two classes are equal. Follows from Savitch's theorem.

$5 \times 4 = 20$

5. A graph $G = (V, E)$ is *bipartite* if V can be partitioned into two sets V_1, V_2 such that every edge in E has one end-point in V_1 and the other in V_2. Let BIPARTITE be the set of all undirected bipartite graphs. Show that $\mathsf{BIPARTITE} \in \mathbf{NL}$.

Hint: Use the fact that $\mathbf{coNL} = \mathbf{NL}$.

20

Solution: To show BIPARTITE is in **NL**, we use the fact that $\mathbf{coNL} = \mathbf{NL}$ and show $\neg\mathsf{BIPARTITE} \in \mathbf{NL}$. .

1. The certificate definition of **NL** is as follows: A language L is in **NL** if there is a polynomial q and a log-space Turing machine $\mathcal{M}$ such that $x \in L \iff \exists u \in \{0,1\}^{q(|x|)}, \mathcal{M}(x, u) = 1$, with the added restriction that u is given

on a special "read-once" tape i.e., the reading head on this tape moves only towards the right. So, x is placed on the input tape, u is placed on the special read-once tape (both of which could have polynomial length) and $\mathcal{M}$ uses at most $O(\log |x|)$ space on its work tape. The reason for the read-once restriction is that in a non-deterministic machine running in log-space, you cannot remember the choices you make.

¬BIPARTITE has short read-once certificates since a graph is not bipartite iff it contains odd cycles. The certificate is an odd cycle that is, a sequence of vertices $v_2, v_2, \ldots, v_k$ that can be verified to form a cycle using $\log n$ space (where $n = |V|$) – remember v_1 (takes $\log n$), for each $i = 1, \ldots, k-1$, check that $\{v_i, v_{i+1}\} \in E$ and lastly, check if $\{v_1, v_k\} \in E$. You do not need to traverse through the certificate backwards at any point and so it is read once. In addition to remembering the first vertex, it is required to maintain a counter to test that the length of the cycle is odd, which again requires $\log n$ space.

2. The above certificate definition can be used to describe a non-deterministic TM $\mathcal{M}$ that will decide ¬BIPARTITE. Let $G = (V, E)$ be an instance of ¬BIPARTITE. Non-deterministically choose a starting vertex $s \in V$; set $u = s$ and $k = 0$. Repeat the following while $i <= n$: if u has degree 0, reject; otherwise, choose an edge $\{u, v\} \in E$, set $u = v$. If $u = s$ and $k > 1$ is odd, then stop and output 1; otherwise increment k and continue execution of the loop. If $i > n$, output 0. Note that this algorithm outputs 1 iff G contains an odd cycle thus deciding ¬BIPARTITE. Since the algorithm need to remember only s, the current vertex u and maintain a counter k that is at most n at any point, the space required is $O(\log n)$.