Ali Muhammad Asad
aa07190

17.5

[Exam 2] Name: **ID:**

Fall 2024: CS 313: Computational Complexity Theory

Due: 12:45 pm, Monday, November 18, 2024. Total Marks: 24

This exam copy contains 3 pages, including this one.

Question 1 4 **[4 points]**

Show that **NL** is closed under intersection.

4 Let $L_1, L_2 \in NL$. Then there exists a non deterministic Turing Machine M_1 & M_2 for L_1 & L_2 that decides L_1 & L_2. We construct another non deterministic Turing Machine M for $L_1 \cap L_2$ as follows:

M_2 "On input x:
1) Simulate the machines M_1 & M_2 on x sequentially.
2) Accept if both machines accept.
3) Else reject". ⇒ Simulation can be done in NL space complexity by first using the tape for M_1 & then reusing the tape for M_2.

Since M_1 & M_2 decide L_1 & L_2 in NL, hence $M \in NL$. Therefore NL is closed under intersection.

Question 2 4 **[6 points]**

For each of the three complexity classes **NL**, **NP**, and **coNP**, give an example of a problem that is complete for the complexity class under logarithmic-space reductions, and provide a very brief explanation of the reduction idea. Try to fit your explanation within the lines provided.

- **NL:** PATH is NL-Complete. PATH can be solved in ~~log space non deterministically by guessing a path from s to t, reusing the same space. Consider an arbitrary~~ log space. → PATH ∈ NL. For a language A in NL, we can construct $\langle G, s, t \rangle$ in log space where the nodes of G are configurations of the machine for A. ~~Language A~~, (C_1, C_2) are an edge iff it is possible to go from configs C_1 to C_2 in M. Hence it reduces. 2
- **NP:** SAT is NP-Complete. We can basically use SAT → its variables & truth assignments to construct any given problem instance such as a graph etc. At any given time, we need to only store the current node/literal which is logarithmic to the size of the input. 1
- **coNP:** TAUT is coNP-complete, & again for any given assignment of truth values, we can evaluate each variable truth assignment on the same space. We only need to know the current truth assignment, which is logarithmic with respect to the input. So it follows a similar argument as with NP. 1

2b: Reducing SAT to another problem will not show that SAT is NP-Hard under log-red.

2c: Same as 2b

Question 3 5.5 [6 points]

For a complexity class C, let co-C $= \{L \mid \overline{L} \in C\}$ and say that C is closed under complementation whenever C = co-C.

Argue as to whether the following statements are true, false, or unknown:

(a) (3 points) All deterministic space complexity classes are closed under complementation.

If a deterministic Turing Machine decides a language L in a given ~~time~~ space, we can construct another deterministic machine to accept the states of $\overline{L}$ in the same ~~time~~ space by swapping the accept & reject states. This doesn't ~~alter~~ change the ~~states~~ transitions, & essentially runs in the same time & space as well. Therefore, all deterministic ~~time~~ space complexity classes are closed under complementation

3

(b) (3 points) All non-deterministic space complexity classes are closed under complementation.

~~We know that NL = coNL by the Immerman Szelepcsényi theorem, but we can't say the same whether this is the same for all non-deterministic space complexity classes is unknown.~~

2.5

By the Immerman Szelepcsinyi theorem, yes.

NL = ~~co~~ coNL & NPSPACE = coNPSPACE

S(n) > log n

& equalities b/w classes scale up.

Question 4 4 [8 points]

Consider the language TQBF = True Quantified Boolean Formula, studied in class.

(a) (2 points) Briefly argue that TQBF is **NP-Hard**, by describing a reduction idea, but not the complete reduction. We know that 2-Player Puzzles/Games exist in NP, & are NP-Hard. A winning strategy for ~~Finding a~~ ~~winning~~ a puzzle essentially reduces to finding a valid truth assignment between quantified boolean formulas, where alternating Quantifiers represent alternate Player moves. So essentially we can reduce TQBF to any such configuration / instance / language of a 2-Player game, therefore TQBF is NP-Hard. → Eg is Generalized Geography, a reduction from TQBF to GG (→ NP-Hard).

1

(b) (2 points) Briefly argue that TQBF is **coNP-Hard**. Alternately we can argue that the complement of ~~solving~~ finding a winning strategy for a puzzle would be if ~~there~~ ~~is~~ ~~the strategy~~ ~~is~~ there is no winning strategy. Then again we can reduce TQBF to that puzzle's language. ~~So~~ If TQBF ~~is~~ is always False, then there is no winning strategy. ~~So~~ Since the complement of the puzzle would be coNP-Hard, TQBF is coNP-Hard.

1

TQBF : $\psi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n (\varphi)$

(c) (2 points) Briefly argue that TQBF is in **PSPACE**.

We can write a recursive algorithm that peels off one quantifier at a time starting with Q_1:

2

1) If $Q_i = \forall$, then $\psi_i = T$ with $x_i = T$ & includes quantifiers from $i+1 \to n$.

2) If $Q_i = \exists$, then $\psi_i = T$ with $x_i = T$ or $\psi_i = T$ with $x_i = F$.

Base Case : ψ_n. And we evaluate.

Since for each instance of the formula & recursion, we store at most 'n' truth assignments of the literals, utilizing the same space, then the space required is proportional to n. So TQBF $\in$ PSPACE.

(d) (2 points) Is TQBF **NP-Complete**? Why or why not?

×

Yes TQBF is NP complete.

We know its NP-Hard by the previous part.

TQBF $\in$ NP.

We can non deterministically guess ~~all~~ a truth assignment for a given literal, & then non deterministically guess the next & so on. One complete assignment / ~~Since one complete assign~~ branch would take upto at most n literals, therefore it can be done in non-deterministic poly time.

Alternatively we can construct a verifier for TQBF that given a certificate 'c' containing the truth assignments for a TQBF instance φ checks if the literals evaluate φ to True. Since there at 'n' literals, the verification can be done in polynomial time.

Hence TQBF $\in$ NP.

Since TQBF $\in$ NP & TQBF is NP-Hard, TQBF is NP-Complete

We know TQBF is PSPACE-Complete!