IITM-CS6014 : Advanced Theory of Computation Author: Karthik Abinav

Problem Set #4

- Rules of the game remain the same. Submissions must be single file in LaTeX format at the upload link set up in cse moodle page of the course.

1. (15 points) Argue whether the following functions qualify to be called as a *resource* according to Blum's resource axioms.

   (a) (3 points) $\mathsf{VALUE}(M, x) = \begin{cases} 0 & \text{if } M \text{ rejects } x, \\ 1 & \text{if } M \text{ accepts } x, \\ \text{undefined} & \text{if } M \text{ does not halts on } x \end{cases}$

   Answer

   The above resource function satisfies Blums first axiom i.e. it is defined only when M halts on x. However, given a M,x,k , it is not decidable to check if VALUE(M,x) = k. Hence it is not a resource.

   Consider $R = \{(M, x, k) : VALUE(M, x) = k\}$

   Suppose R was decidable via a total TM N, then MP can be decided by giving $(M, x, 1)$ as input to N. But, MP is not decidable and hence R is not decidable.

   (b) (5 points) A *turn* of tape head is defined as movement of tape head from $L \to R$ or $R \to L$.

   $\mathsf{TURN}(M, x) = \begin{cases} \text{No. of turns that tape head makes when } M \text{ runs on } x & \text{if } M \text{ halts on } x \\ \text{undefined} & \text{otherwise} \end{cases}$

   Answer

   The above resource function satisfies Blums first axiom i.e. it is defined only when M halts on x. The above resource also satisfies Blum's second axiom, i.e. $R = \{(M, x, k) : TURN(M, x) = k\}$ is a decidable set.

   Consider a total TM N corresponding to R. Let the description of this machine be as follows:

   - Run M on x.
   - If the machine makes more than k turns, reject and halt.
   - Else, machine M will halt on x and hence if the number of turns made by the machine is k, accept. Else , reject.

   Claim

   Language accepted by a machine which makes finite number of turns is regular.

   This is because, the crossing sequence at every index for this machine will be bounded by a fixed constant s , and from class we saw such machines accept only regular languages.

From the claim above, the machine N will always halt and hence is total.

(c) (5 points) $\mathsf{COUNT}(M, x) = \begin{cases} \text{No. of times } M \text{ vists a state } q & \text{if } M \text{ halts on } x \\ \text{undefined} & \text{otherwise} \end{cases}$

Answer

The above resource function satisfies Blums first axiom i.e. it is defined only when M halts on x. However, given a M,x,k , it is not decidable to check if COUNT(M,x) = k. Hence it is not a resource.

Consider $R = \{(M, x, k) : COUNT(M, x) = k\}$

Suppose R was decidable, then we can use it as a sub routine to decide the HP as follows :

Construct a machine N as follows:

- Simulate M on x.
- If M halts on x, then have a trasition to a special accept state.

This gives the reduction from R to HP.

2. (5 points) Show that if $\mathsf{NTIME}(n) = \mathsf{DTIME}(n)$ then $\mathsf{P} = \mathsf{NP}$. (Padding !!)

Answer

Let $NTIME(n) = DTIME(n)$.
Let $L \in NP$ via a non-deterministic machine M.
Consider $L_{pad} = \{x\#1^{|x|^c} : x \in L\}$

Consider a N which does the following on input y:

- Check if $y = x\#1^{|x|^c}$.
- Extract x
- Run machine M on x, to check if $x \in L$. If yes, accept. Else, reject.

The running time of the above machine is linear in its input size. Therefore, $L_{pad} \in NTIME(n)$.
From assumption, $L_{pad} \in DTIME(n)$.Let the deterministic machine accpeting this be N'.

Consider a machine M' which does the following:

- Construct $y = x\#1^{|x|^c}$
- Check if $y \in L_{pad}$. If yes, accept . Else , reject.

$L(M') = L$. The time taken by this deterministic machine is polynomial in its input length. Hence, $L \in P$.

3. (10 points) Space Hierarchy theorem implies the following: For any $k > 0$, There is a language in $\mathsf{DSPACE}(n^{k+1})$ that is not in $\mathsf{DSPACE}(n^k)$. Use this and a padding argument to show that: $\mathsf{P} \neq \mathsf{DSPACE}(n)$. (6pts)
(Note that we do not know the containment in either direction.)
You can do this in two steps:
   (a) For every language L define, $L_{pad} = \{x01^{|x|^2} \ : \ x \in L\}$.
   Argue that L_{pad} is in $\mathsf{P} \implies L \in \mathsf{P}$.
   (b) Show an L_{pad} which is in $\mathsf{DSPACE}(n)$ but whose corresponding L is not in $\mathsf{DSPACE}(n)$.

Answer
(a) For every language L , let $L_{pad} = \{x01^{|x|^2} \ : \ x \in L\}$.
Let $L_{pad} \in \mathsf{P}$ via a deterministic machine M running in polynomial time.
Given an input y, M does the following

- Checks if y is of the form $x01^{|x|^2}$.
- Extracts x out of it, and checks if $x \in L$.

Since, M runs in polynomial time, it does steps 1 and 2 in polynomial time each. Given an x, checking if $x \in L$ can be done in polynomial time by a deterministic machine. Hence, $L \in \mathsf{P}$.

(b) From space hierarchy theorem, $\exists$ A such that , $\mathsf{A} \in \mathsf{DSPACE}(n^2)$ but $\mathsf{A} \notin \mathsf{DSPACE}(n)$.
Consider $\mathsf{A}_{pad} = \{x01^{|x|^2} \ : \ x \in A\}$
Consider a machine N accepting A_{pad}. It does the following

- Checks if the input y is of the form $x01^{|x|^2}$.
- It extracts x from the input and checks if $x \in \mathsf{A}$.

This machine takes space $|x|^2$. That is $\mathsf{DSPACE}(n)$. (Since input size is $|x|^2$). Hence, (b) is also true.

Now, it remains to show that (a) and (b) $\implies$ $\mathsf{P} \neq \mathsf{DSPACE}(n)$.

This can be shown by contradiction. Suppose $\mathsf{P} = \mathsf{DSPACE}(n)$. Then $\mathsf{A}_{pad} \in \mathsf{DSPACE}(n) \implies \mathsf{A}_{pad} \in \mathsf{P} \implies \mathsf{A} \in \mathsf{P} \implies \mathsf{A} \in \mathsf{DSPACE}(n)$.
But this is a contradiction, because A belongs to $\mathsf{DSPACE}(n^2)$ and not $\mathsf{DSPACE}(n)$.

Therefore, (a) and (b) $\implies$ $\mathsf{P} \neq \mathsf{DSPACE}(n)$.

4. (5 points) Show that if $\mathsf{SAT} \in \mathsf{NP} \cap \mathsf{coNP}$ then $\mathsf{NP} = \mathsf{coNP}$. (Definitions !)
Answer
SAT is $\mathsf{NP} - Complete$. Hence, for every language L in NP, there is a polynomial time reduction to SAT. And, $\mathsf{SAT} \in \mathsf{NP} \cap \mathsf{coNP} \implies L \in \mathsf{NP} \cap \mathsf{coNP} \implies$ Every Language in

NP belongs to $\mathsf{NP} \cap \mathsf{coNP} \implies$ Every Language in NP belongs to coNP ...(1)

Consider a language L' in $\mathsf{coNP} \implies \overline{L'} \in \mathsf{NP} \implies \overline{L'} \in \mathsf{NP} \cap \mathsf{coNP} \implies \overline{L'} \in \mathsf{coNP} \implies L' \in \mathsf{NP}$ .. (2)

From (1) and (2), $\mathsf{NP} = \mathsf{coNP}$.

5. (5 points) If L, L' are in NP, then show that $L \cup L'$, $L \cap L'$ are in NP. (Definitions !)

Answer

From defn, $L \in \mathsf{NP} \Leftrightarrow$ there exists a polynomial length certificate c_1 which can be verified in polynomial time via a machine M_1.
Similarly, $L' \in \mathsf{NP} \Leftrightarrow$ there exists a polynomial length certificate c_2 which can be verified in polynomial time via a machine M_2.

$L \cap L'$, construct a new certificate $c_1\#c_2$. This is of polynomial size. To verify if $x \in L \cap L'$, we need to construct a polynomial time running machine which verifies c_1 and c_2. Construct a new machine N which first simulates M_1 on input c_1 and verifies if its the correct certificate of L. If yes, it proceeds to simulating M_2 on c_2. Else rejects. Similarly if c_2 is verified as the correct certificate of L' by simulation of M_2 then it will accept. Else reject. Since simulation of both M_1 and M_2 takes polynomial time in the input length, hence N also runs in polynomial time.

$L \cup L'$, construct a new certificate $c_1\#c_2$. This is of polynomial size. To verify if $x \in L \cup L'$, we need to construct a polynomial time running machine which verifies c_1 and c_2. Construct a new machine N which first simulates M_1 on input c_1 and verifies if its the correct certificate of L. If no , it proceeds to simulating M_2 on c_2. Else accepts. If c_2 is verified as the correct certificate of L' by simulation of M_2 then it will accept. Else reject. Since simulation of both M_1 and M_2 takes polynomial time in the input length, hence N also runs in polynomial time.

6. ([illegible] points) If L, L' are in $\mathsf{NP} \cap \mathsf{coNP}$, then show that $L \oplus L'$ defined as

$$L \oplus L' = \{x : x \text{ is in one of } L \text{ or } L' \text{ but not both.}\}$$

is in $\mathsf{NP} \cap \mathsf{coNP}$. (Definitions !)

Answer

If $L \in \mathsf{NP} \cap \mathsf{coNP}$ implies $\overline{L} \in \mathsf{NP}$. Similarly, $\overline{L'} \in \mathsf{NP}$.
Let c_1, c_2, c_3, c_4 be the polynomial size certificates verifiable in polynomial time corresponding to language $L, \overline{L}, L', \overline{L'}$ respectively.

Similar to the construction above we have to check if c_1 *and* c_3 are the correct certificates or c_2 *and* c_4 are correct certificates for the respectively languages $L \cap \overline{L'}$ and $\overline{L} \cap L'$. Hence, $L \oplus L' \in \mathsf{NP}$.

Similarly, we have to check if c_1 *and* c_4 are the correct certificates or c_2 *and* c_3 are correct certificates for the respectively languages $\overline{L} \cap \overline{L'}$ and $L \cap L'$. Hence, $L \oplus L' \in \mathsf{coNP}$.

Therefore, $L \oplus L' \in \mathsf{NP} \cap \mathsf{coNP}$

7. (15 points) Consider the following language: PRIMES $= \{n \mid n \text{ is a prime }\}$ where the input n is in binary. Without using the known result that PRIMES is in P, solve the following:

(a) (5 points) Show that PRIMES is in coNP.

Answer

To show that PRIMES $\in \mathsf{coNP}$, we have to show that $\overline{PRIMES} \in \mathsf{NP}$.
$\overline{PRIMES} = \{n \mid \exists p, q \ s.t. \ p, q \neq 1 \ and \ p * q = n\}$
Given p,q as a certificate (each of them need log n bits respectively and hence polynomial in the size of input n), we can verify if p $*$ q = n in polynomial time as follows:
Calculate n/p using long-division. Since each step of long division eliminates on digit from the dividend(and every step is polynomial in dividend length), hence it runs for polynomial time in the length of dividend. Now check if the obtained quotient is equal to q and the obtained remainder is equal to 0 (This can be done in linear time with respect to length of q).

Therefore,$\overline{PRIMES} \in \mathsf{NP}$ and hence PRIMES $\in \mathsf{coNP}$.

(b) (10 points) Here is Lucas test for primality (you dont need prove it) : n is prime if and only if there is an integer $a \in \{2, \ldots, n-1\}$ with $a^{n-1} \equiv 1 \mod n$, and for every prime factor q of $n-1$: $a^{\frac{n-1}{q}} \not\equiv 1 \mod n$. Use this test to show that PRIMES is in NP.

Answer

Given a and the prime factorization of n-1 as a certificate, it is easy to verify the above condition of Lucas in O(log^3(n)) time. Modular exponentiation takes O(log n) time and at each stage multiplication takes O(log^2(n)) steps. Now we have to show that $|a|$ + |prime factorization of n-1| is a polynomial in log n.

Clearly, since $a < n$, therefore it can be represented using log(n) bits. Now we have to check
1) prime factorization of n-1 can be represented in poly in log(n) bits.
2) Each of the prime factors should be further verified as a prime.

Let n-1 be factorized as $p_1^{a_1} * p_2^{a_2} * ..p_k^{a_k}$.
This product is greater than $2^{a_1} * 2^{a_2} * ...2^{a_k}$. Therefore, $a_1 + a_2 ... a_k < log(n)$. Implies the number of prime factors of n-1 is utmost log(n).

For each of $p_1, p_2..p_k$, there will be another set of certificates. Each of these certificates is a polynomial in log(n) and verifiable in polynomial in log(n)(From strong

induction, because each of $p_1, p_2..p_k$ is strictly smaller than n-1). Hence, the total time for verifying the prime factorization for n-1 is (polynomial in log(n)) $*$ (polynomial in log(n)), which is polynomial in log(n).
And since each of the prime factors can be log(n) length and there are log(n) of them, the total length is a polynomial in log(n).

Therefore, the certificate a , prime factors of (n-1) is a polynomial size in input length verifiable in polynomial time in input length. Therefore, PRIMES $\in \mathsf{NP}$.

Hence conclude that PRIMES is in $\mathsf{NP} \cap \mathsf{coNP}$.

From (a),(b), PRIMES is in $\mathsf{NP} \cap \mathsf{coNP}$.

8. (10 points) Prove that reachability in undirected forests (a possibly disconnected acyclic undirected graph) can be solved in log-space. That is, given (T, s, t) where T is an undirected forest, it can be tested in log-space whether s is connected to t by a path.

9. (18 points) Let $\mathsf{E} = \bigcup_{c>0} \mathsf{DTIME}(2^{cn})$ and $\mathsf{NE} = \bigcup_{c>0} \mathsf{NTIME}(2^{cn})$. A set A is called *sparse* if there is a polynomial p, such that $|\{x \in A : |x| = n\}| \leq p(n)$. A set A is called *tally set* if $A \subseteq \{1\}^*$. Prove that following are equivalent.

    1. Restricted to tally sets $\mathsf{NP} = \mathsf{P}$. That is all tally sets in NP are in P.
    2. Restricted to sparse sets $\mathsf{NP} = \mathsf{P}$. That is all sparse sets in NP are in P.
    3. $\mathsf{E} = \mathsf{NE}$

Hence conclude that $\mathsf{E} \neq \mathsf{NE} \implies \mathsf{P} \neq \mathsf{NP}$.

Hint : Try for (b) $\implies$ (a) $\implies$ (c) $\implies$ (b). For the second implication : consider the language $L_{tally} = \{1^{2^{c|x|}} : x \in L\}$. This will not work, but a slight modification of this language which includes some more information about x will work !.
For the third implication, consider the language

$$L_{order} = \{(k, i, c) : \textit{the } i^{th} \textit{ bit of the } k^{th} \textit{ string(in lex order) in } L \textit{ is } c\}$$

Answer

From the definition, every tally set A is a sparse set for a polynomial $p(n) = 2$. Because $\mathsf{A} \subseteq \{1\}^*$ implies $\forall n\ |\{|x| = n : x \in A\}|$ is either 0 or 1.
Hence, (b) $\implies$ (a) is trivially true.That is if every sparse set in NP is in P, then every tally set(which is a subset of sparse set) in NP is in P.

`Idea borrowed from akshay`

(a) $\implies$ (c)
Let, $x \in L$(via a non deterministic machine M) such that, $L \in \mathsf{NE}$. Consider the padded

version of L as $L_{tally} = \{1^{c*i}$: i th lexicographic string in $\{0,1\}^*$ is $x.\}$
Clearly, the length of 1^{c*i} is in the order of $2^{|x|}$ (because log n bits are used to represent x) Now a new nondeterministic machine M' can accept the above padded language in NP time because the size of input is exponential compared to size of input for M. From assumption, $\mathsf{NP} = \mathsf{P}$ for tally sets. Hence there is a deterministic machine N which accepts L.

Consider a machine N' accepting L. It does the following

- It pads the input as $y = 1^{c*i}$.
- It checks if $y \in L_{tally}$ in the same fashion as N.

Since N is a deterministic machine, N' is also a deterministic machine. And since it runs in $O(2^n)$ time with respect to its input, hence L belongs to E.

Therefore, (a) $\implies$ (c).

10. (7 points) Imagine a world in which $\mathsf{P} = \mathsf{NP}$. Now show that there is a polynomial time algorithm which given a Boolean formula ϕ produces a satisfying assignment for ϕ if ϕ is satisfiable.(Hint : Use queries to SAT).
Answer
We can get the assignments for $x_1, x_2, ...x_n$ in n+1 queries to SAT as follows.

- Firstly, make a query to SAT to find if ϕ is satisfiable. If no, immediately return.
- If ϕ is satisfiable, construct new expressions, ϕ_1,ϕ_2...ϕ_n, where, $\phi_i = \phi \wedge x_i$. If, ϕ_i is satisfiable(By query on SAT) assign $x_i = 1$. Else, assign $x_i = 0$.

Since, $\mathsf{P} = \mathsf{NP}$, therefore there is a polynomial time algorithm to SAT. And since, the number of queries to SAT is in $O(n)$(and ϕ has atleast n variables), therefore, above algorithm runs in polynomial time.