Homework 9 Morally DUE May 5 at 11:00 AM
THIS HW IS TWO PAGES LONG!!!!!!!!!!!!!!!!!!

1. (35 points) Give the reduction $3COL \le CNF - SAT$.

   SOLUTION

   We are given a graph $G = (V, E)$.

   We assume $V = \{1, \ldots, n\}$.

   For every vertex i we have 3 Boolean variables. We list them and what they mean

   x_{iR}: T if $COL(i) = R$.

   x_{iB}: T if $COL(i) = B$.

   x_{iB}: T if $COL(i) = G$.

   We now write down a formula in two parts

   PART ONE: Making sure that a satisfying assignment really is a (not necessarily proper) coloring

   Every vertex has at least one color:

   $$\wedge_{i=1}^{n}(x_{iR} \vee x_{iB} \vee x_{iG})$$

   Every vertex has at most one color:

   $$\wedge_{i=1}^{n}\neg(x_{iR} \wedge x_{iB}) \wedge \neg(x_{iR} \wedge x_{iG}) \wedge \neg(x_{iB} \wedge x_{iG})$$

   PART TWO: Make sure it's a proper coloring

   $$\wedge_{(i,j)\in E}\neg(x_{iR} \wedge x_{iR}) \wedge \neg(x_{iB} \wedge x_{iB}) \wedge \neg(x_{iG} \wedge x_{iG})$$

   END OF SOLUTION

2. (35 points) We assume all Turing Machines have $\Sigma = \{1, 2, 3\}$ and 3 is the # symbol. and the state set Q is an initial segment of $\mathsf{N} - \{0\}$ (that is, it will be something like $\{1, 2, 3, 4\}$).

   (a) (20 points) Describe a procedure to code Turing Machines into N such that the following holds:

- Two different Turing Machines map to different numbers. (Though it is okay if some numbers do not get mapped to.)
- The following should be computable:
  Input: $x, y \in \mathsf{N}$
  Output:
  If x does not code a TM than output THATSBSMAN.
  If x does code a TM than let it be M_x. Run $M_x(y)$ (this might diverge, and that's fine.)

HINT- do not over think this. Any way you code a TM into numbers should work.

(b) (15 points) Let M be the TM: $Q = \{1, 2, 3\}$, $\Sigma = \{1, 2, 3\}$, $s = 1$, $h = 3$,
$\delta(1, 1) = (1, L)$.
$\delta(1, 2) = (1, 2)$.
$\delta(1, 3) = (2, R)$.
$\delta(2, 1) = (1, 1)$.
$\delta(2, 2) = (3, 3)$.
$\delta(2, 3) = (3, L)$.
Use your procedure to encode this into a number. Show your work and give us your number. (If your number involves the product of numbers, you need not multiply them together. For example, if the above codes to $2^{7^6} \times 3^{4^5}$ then you can leave it in that form and not do the multiplication.)

SOLUTION ON NEXT PAGE

SOLUTION

THE CODING:

Let $M = (Q, \{a, b, \#\}, \delta, s, h)$

The number will be the product of the following numbers

(a) $2^{|Q|}$.

(b) 3^s (Recall that s, the start state, is a number)

(c) 5^h (Recall that h, the halt state, is a number)

(d) there will be $n = (Q-1) \times \Sigma$ rules. Let $p_1, \ldots, p_n$ be the n primes after 5 (so $p_1 = 7$). (It's $Q-1$ since there are no transitions out of h.) Order the rules lexicographically by $Q \times \Sigma$, so

$\delta(1,1)$

$\delta(1,2)$

$\delta(1,3)$

$\delta(2,1)$

$\delta(2,2)$

$\delta(2,3)$

$\vdots$

$\delta(|Q|-1, 3)$.

For $1 \le i \le n$ take rule i and form the following number.

i. $\delta(p, \sigma) = (q, \sigma')$ maps to $2^p \times 3^\sigma \times 5^q \times 7^{\sigma'}$. Note that $\sigma' \in \{1, 2, 3\}$.

ii. $\delta(p, \sigma) = (q, L)$ maps to $2^p \times 3^\sigma \times 5^q \times 7^4$. Note that $4 \notin \{1, 2, 3\}$ so it won't be confused with a symbol.

iii. $\delta(p, \sigma) = (q, R)$ maps to $2^p \times 3^\sigma \times 5^q \times 7^5$. Note that $5 \notin \{1, 2, 3\}$ so it won't be confused with a symbol or with the number that encodes L.

GOTO NEXT PAGE FOR THE CODING OF THE TM

$Q = \{1, 2, 3\}$ (so the number has 2^3),

$\Sigma = \{1, 2, 3\}$,

$s = 1$ (so the number has 3^1),

$h = 3$ (so the number has 5^3).

$\delta(1, 1) = (1, L)$. This is coded by $7^{2^1 3^1 5^1 7^4}$

$\delta(1, 2) = (1, 2)$. This is coded by $11^{2^1 3^2 5^1 7^2}$

$\delta(1, 3) = (2, R)$. This is coded by $13^{2^1 3^3 5^2 7^5}$

$\delta(2, 1) = (4, 1)$. This is coded by $17^{2^2 3^1 5^4 7^1}$

$\delta(2, 2) = (3, 3)$. This is coded by $23^{2^2 3^2 5^3 7^3}$

$\delta(2, 3) = (3, L)$. This is coded by $29^{2^2 3^3 5^3 7^4}$

So the Turing Machine maps to the number

$$2^3 \times 3^1 \times 5^3 \times$$

$$7^{2^1 3^1 5^1 7^4} \times 11^{2^1 3^2 5^1 7^2} \times 13^{2^1 3^3 5^2 7^5} \times 17^{2^2 3^1 5^4 7^1} \times 23^{2^2 3^2 5^3 7^3} \times 29^{2^2 3^3 5^3 7^4}.$$

END OF SOLUTION

GOTO THE NEXT PAGE

3. (30 points) During this problem we will use $M_1, \ldots, M_{100}$ to mean ANY 100 Turing Machines. They are not associated to any numbering.

   HALTON0 is the set of all Turing Machines that halt on input 0.

   (a) (10 points) Bill gives you 100 Turing Machines $M_1, \ldots, M_{100}$. He wants to know if *at least 17 of them are in HALTON0.*

   Come up with a Turing Machine M (by that I mean just write psuedocode that uses $M_1, \ldots, M_{100}$) so that

   $$M \in \text{HALTON0 iff at least 17 of } M_1, \ldots, M_{100} \text{ are in HALTON0.}$$

   (b) (10 points) Bill gives you 100 Turing Machines $M_1, \ldots, M_{100}$. He wants to know HOW MANY are in HALTON0.

   If you could ASK HALTON0 100 questions then you could do this—just ask $M_1 \in$ HALTON0?, $M_2 \in$ HALTON0?,..., $M_{100} \in$ HALTON0? and output the number that returned YES.

   What if you can ask HALTON0 less than 100 questions? Find a number $q < 100$ such that you can determine HOW MANY are in HALTON0 with q questions to HALTON0. Write psuedocode (which will make q queries to HALTON0) that will, on input $M_1, \ldots, M_{100}$, output HOW MANY of them are in HALTON0 (so the output is a number between 0 and 100). Try to make q as small as you can. (HINT: Use part (a).)

   (c) (10 points) Bill gives you 100 Turing Machines $M_1, \ldots, M_{100}$. He wants to know WHICH ONES halt on 0.

   If you could ASK HALTON0 100 questions then you could do this—just ask $M_1 \in$ HALTON0?, $M_2 \in$ HALTON0?,..., $M_{100} \in$ HALTON0? and see see which ones return YES.

   What if you are allowed to ask HALTON0 less than 100 questions? IS there a number $q < 100$ such that you can determine WHICH of $M_1, \ldots, M_{100}$ are in HALTON0 with q questions to HALTON0? Prove your result.

   SOLUTION

(a) Turing Machine M:

   i. Run $M_1(0), \ldots, M_{100}(0)$ all at the same time and wait until 17 of them halt.

   ii. If you see 17 of them halting, then halt.

   Clearly M halts on 0 (actually on any input) IFF ≥ 17 of the M_i's halt.

(b) Here is the procedure:

   i. Create a TM M such that M halts on 0 IFF at least 50 of $M_1, \ldots, M_{100}$ halt on 0. (use the technique in part (a)). If YES then we know ≥ 50 of them halt on 0, if NO then we know that ≤ 49 of them halt on 0.

   ii. Proceed by binary search to find out how many halt on 0.

   The number of queries to HALTON0 is $\lceil \log_2(100) \rceil = 7$.

(c) First apply the technique of part (b) to find out HOW MANY halt on 0. This only took 7 queries. Say the answer is that a of them halt.

   Then *RUN ALL OF THEM UNTIL a HALT!* Once a halt, you know which a halt and you know that NO OTHERS will halt. So we know the a that halt, and that the others DO NOT.

END OF SOLUTION