CSE431: Complexity Theory

Homework 2

Anup Rao Due:

1. Suppose we can compute $h(\alpha)$ for every α, we show that we can compute the Halting Problem to obtain a contradiction.

   Let (α, x) be the input of the Halting Problem, we want to decide if $M_\alpha(x)$ halts or not. To do this, we construct a Turing Machine $M'_{\alpha,x}$ that ignores its input and simulates M_α on x. Whenever the simulation ends, it outputs 1. Let β be the code of $M'_{\alpha,x}$. Since we can compute h, we can test if $h(\beta) = 0$ or $h(\beta) = 1$. Since $M'_{\alpha,x}$ ignores its input, it halts with output 1 for all inputs if and only if M_α halts on x. Therefore, when $h(\beta) = 0$ we know that M_α does not halt on x and when $h(\beta) = 1$ we know that M_α does halt on x. Thus, we can decide whether $M_\alpha(x)$ halts or not by computing $h(\beta)$ - a contradiction.

2. Since every Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit of size $O(2^n)$, we know that in particular, the function h from Problem 1 can be computed by circuits of size $O(2^n)$. We will next show that a slight modification of this function that can be computed by much smaller circuit even though the function is not computable by Turing machines.

   Let $g(x, y)$ be a function that partitions its input of length into x, y where x is of length $\log n$, and y is of length $n - \log n$, and $g(x, y) = h(x)$. We first note that g is also not computable by any Turing machine, for otherwise, any Turing Machine M computing g could be used to compute $h(\alpha)$. Now we show a polynomial sized circuit computing g. Assume the input to the circuit is n bits. Since g is only a function of the first $\log n$ bits, g can be computed by a circuit of size $2^{O(\log n)} = n^{O(1)}$.

3. We prove this by giving a reduction from $3SAT$ to IP. We need to convert the clauses of the given $3SAT$ formula to a matrix A and vector b.

   Given a clause like $x_1 \vee \neq x_2 \vee x_3$, we can express it using the inequality:

   $$x_1 + (1 - x_2) + x_3 \geq 1$$

   which is equivalent to

   $$x_1 - x_2 + x_3 \geq 0$$

   We see that the inequality is satisfied if and only if one of the three literals in the clause is set to true. In this way, every clauses can be converted into a linear inequality involving 3 variables in polynomial time.

4. Let $V(x, w)$ be the verifier of an NP problem. Then in polynomial time, we can compute the code of a machine M that does the following: $M(x)$ runs over all choices for w, and computes $V(x, w)$. If $V(x, w)$ outputs 1, then M halts. If $V(x, w)$ outputs 0 for all w, then M enters into an infinite loop.

   If α is the code of M, then we see that $HALT(\alpha, x) = V(x, w)$. So we have shown that every NP problem can be reduced to the halting problem.

HALT cannot be NP-complete, because every NP-complete problem is in EXP, and HALT cannot be computed by a Turing machine.