CS 278 Complexity Theory
Problem Set 4

Due: Apr 16, 2020, 11:59 PM PST

Submission on Gradescope. You are encouraged to discuss the problems and solve them in groups. However, the solutions are to be written up alone, **listing all the collaborators**.

1. **SAT Algorithms for $\mathbf{ACC}^0$ – Part One:** (18 points) In this exercise, we will give a faster-than-brute-force SAT algorithm for ACC0 circuits, assuming a structural result by Yao, Beigel and Tarui.

   Definition 1. *A SYM+ circuit C' on n variables of weight w is a pair (Ψ, P), where $\Psi : \{0, 1, \ldots, w\} \to \{0, 1\}$ and $P : \{0, 1\}^n \to \mathbb{Z}$ is a multilinear polynomial that can be written as $P(x) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i$ with non-negative integer coefficients a_S that sum up to w. The value of C' on input bits $x_1, \ldots, x_n$ is simply $\Psi(P(x_1, \ldots, x_n))$.*

   Theorem 2. *Let C be an $\mathbf{ACC}^0$ circuit of depth $d = O(1)$, size s and n input bits. Then, there exists a SYM+ circuit C' of weight $2^{\log(s)^{O(1)}}$ that is equivalent to C. Furthermore, there's an algorithm that runs in time $2^{\log(s)^{O(1)}}$ that output $C' = (\Psi, P)$.*

   Our goal will be to show that there exists an algorithm that checks whether an $\mathbf{ACC}^0$ circuit of size s on n input bits is satisfiable or not, in time $O(2^n \cdot n)$, as long as $2^{\log(s)^{O(1)}} \leq 2^n$.[1]. Note that the naive algorithm that evaluate the circuit on all 2^n assignments takes $O(2^n \cdot s)$ time (so we improve when $n \ll s$).

   The algorithm first converts C to a the SYM+ representation (P, Ψ) using the above theorem. Suppose even further that $P(x) = \sum_{S \subseteq [n]} a_S \cdot \prod_{i \in S} x_i$ is written as a list of 2^n integer coefficients $\{a_S : S \subseteq [n]\}$, and that Ψ is written as a list of $w + 1$ values $\Psi(0), \ldots, \Psi(w)$. Using divide-and-conquer approach, we show that there's an algorithm that computes the values of P on all 2^n inputs in $\{0, 1\}^n$ in time $O(2^n \cdot n)$.

   The idea is as follows: Write $P(x_1, \ldots, x_n) = P_1(x_1, \ldots, x_{n-1}) \cdot x_n + P_0(x_1, \ldots, x_n)$ where both P_1 and P_0 are multilinear polynomials with integer coefficients.

   (a) Describe how to get a list of coefficients for P_1 and P_0 from the list of coefficients for P.

   (b) Suppose you evaluated P_0 and P_1 on all 2^{n-1} assignments for $x_1, \ldots, x_{n-1}$. Derive a list of all 2^n values of P in $O(2^n)$ time (you can assume integer addition takes $O(1)$ time).

   (c) Using recursion, what would be the time complexity of your entire algorithm? (Hint: write a recurrence relation involving n and solve for n.)

   (d) Explain how to check if C' is satisfiable in $O(2^n \cdot n)$ time.

[1] Equivalently, $\log(s)^{O(1)} \leq n$ which certainly holds for any $s = \text{poly}(n)$ and n large enough.

2. **SAT Algorithms – Faster than 2^n Time.** (12 points) In the previous question we gave a SAT algorithm for $\mathbf{ACC}^0$ circuits that runs in $O(2^n \cdot n)$ time for any circuits of poly(n) size and constant depth. Next, we want to give a SAT algorithm for $\mathbf{ACC}^0$ circuits of size $s = \text{poly}(n)$ that runs in time $O(2^n/n^{100})$.

   (a) Let $t = 101 \cdot \log n$. Let C be a $\mathbf{ACC}^0$ circuit on n input bits of size $s = \text{poly}(n)$ and depth $d = O(1)$. For any fixed assignment $z \in \{0,1\}^t$ to the first t bits, let $C_z(x_{t+1}, \ldots, x_n) = C(z_1, \ldots, z_t, x_{t+1}, \ldots, x_n)$. Note that C_z is attained from C by hardwiring the first t input bits to $(z_1, \ldots, z_t)$, and thus C_z is a circuit of $n - t$ input bits of size at most s. Show that C is satisfiable if and only if $C^*(x_{t+1}, \ldots, x_n) = \bigvee_{z \in \{0,1\}^t} C_z(x_{t+1}, \ldots, x_n)$ is satisfiable.

   (b) What is the circuit size of C^*? On how many input bits it depends? What is its depth?

   (c) Apply the SAT algorithm from Question 1 on C^*. Show that it checks whether C^* is satisfiable in time at most $O(2^n/n^{100})$.

3. **Parallel Repetition of AM.** (15 points) Recall that an $\mathbf{AM}[2]$ protocol is a protocol that on input x, Arthur sends a random string r, Merlin responds with a string y and then Arthur applies a deterministic predicate $A(x, y, r)$. Suppose Π is a $\mathbf{AM}[2]$ protocol with

   - (**Perfect Completeness**): If $x \in L$, $\mathbf{Pr}_r[\exists y = y(x, r) : A(x, y, r) = 1] = 1$
   - (**Soundness error at most** $1/3$): If $x \notin L$, $\mathbf{Pr}_r[\exists y = y(x, r) : A(x, y, r) = 1] \leq 1/3$

   We consider the parallel repetition of this protocol. First, Arthur sends k messages in parallel as its first message, $(r_1, \ldots, r_k)$. Then, Merlin sends k messages in parallel, $(y_1, \ldots, y_k)$ as the second message of the protocol. Finally, Arthur applies the predicates $A(x, y_1, r_1), \ldots, A(x, y_k, r_k)$ and accepts if and only if all $A(x, y_i, r_i) = 1$ for all $i \in [k]$. Show that

   - If $x \in L$, then there exists a strategy for Merlin such that Arthur always accepts.
   - If $x \notin L$, no matter how Merlin answers, Arthur would accept with probability at most $(1/3)^k$.

   Note: in the case $x \notin L$, Merlin could "cheat" and have its answer y_i depending on $(r_1, \ldots, r_k)$ and not only r_i.

4. **MAM=AM.** (Extra Credit - 10 points) An **MAM** protocol is one where Merlin speaks first, then Arthur sends a random string, then Merlin speaks again, and finally Arthur applies a predicate on the input x and all messages. We denote by **MAM** the class of languages that have **MAM** protocols with perfect completeness and soundness error $1/3$. Show that $\mathbf{MAM} \subseteq \mathbf{AM}$. (Hint: Generalize Q3 to **MAM** protocols to reduce the soundness error. What happens when Arthur sends its message before Merlin's first message? Is the protocol still sound? Show that it is assuming the original soundness error is extremely small.)