Oracles & Circuits

Ben Little

Oct ??, 2023

Problem 1

Using the technique from Hartmanis, Swelson, and Immerman (1983).

Given a lanuguage $A \in \mathbf{E}$, we can create a new language A' whose strings are very-inefficiently coded strings in A. So inefficient, in fact, that A' is actually in $\mathbf{P}$.

In this case, use the $TALLY(A)$ language. Encode each string of A into 1-prefixed binary representation, which only adds a $n \log(n)$ factor to the lengths of strings and so $BINARY(A) \in E$. Then re-encode this binary representation into unary, which increases the length of strings exponentially. Then $TALLY(A) \in \mathbf{P}$.

If $\mathbf{E} \neq \mathbf{NE}$, then take any language in $\mathbf{NE} - \mathbf{E}$, and use it to produce a $TALLY$ language that is in $\mathbf{NP} - \mathbf{P}$

Problem 2

Part a

By induction. Base case is trivial: a depth 1 decision tree has at most one literal, so its corresponding DNF has at most one term.

For the inductive case, by hypothesis the subtrees corresponding to 0 and 1 branches of the root have DNF formulas $\bigvee \phi_i$ and $\bigvee \psi_i$ where each ϕ_i and ψ_i are conjunctions. When the root is considered, the formula is $\neg x \wedge \bigvee \phi_i \vee x \wedge \bigvee \psi_i$, which is $\bigvee \neg x \wedge \phi_i \vee \bigvee x \wedge \psi_i$, a DNF of width d.

Part b

Converting a DNF to CNF does not change the width of the formula because converting ANDs to ORs and vice versa does not change the number of boolean operator alternations.

Problem 3

Part a

Let D be a decision tree that takes all oracle responses in the trace of $M^a(1^n)$ as input. Since $M^A(1^n)$ runs in time n^t, it makes at most n^t queries and therefore the depth of the decision tree is at most n^t.

If instead we use binary input, the depth of the tree is at most $\log(N)^t$ or $poly(\log(N))$.

Part b

Let the subtrees of the OR be the decision trees for each determinisitic execution. From part a these have height $O(n^t)$. Because there are n^k non-deterministic bits, there are 2^{n^k} possible assignments and as many subtrees.

Problem 4

Part a

The n-bit OR has exactly one input that outputs 0, namely the input with all zeros. Any decision tree of depth less than n must interogate fewer than n bits and therefore can't confirm that all bits are zero. So there exist inputs to this decision tree that disagree with the OR function.

Part b

In short, a poly-depth circuit family can branch enough to decide any NP problem, but it can't OR the the branches back together to see if any witness accepts.

Consider the decision tree for NP^A from problem 3. Since there are $2^{O(n)}$ subtrees in the latter and there can be no poly-log-depth circuit that computes the OR function, there exists an oracle such that no circuit that computes the function in $poly(O(n))$ depth. Since every poly-time TM can be simulated by a poly-depth circuit family, there exists no such machine that computes the function.

Problem 5

No attempt yet

Problem 6

No attempt yet

Problem 7

No attempt yet