Problem Set 4
Oracles & Circuits
CSCI 6114 Fall 2026

Joshua A. Grochow

Released: September 17, 2026
Due: September 24, 2026

Resources

- Ker-I Ko has an expository survey giving many examples of the connection between circuit lower bounds and oracles.
- Jukna's book on Boolean functions is an excellent resource on constant-depth circuits (Chapters 11 and 12) and decision trees (Chapter 14), among *many* other things!

Exercises

1. An N-variable (Boolean) *decision tree* is a binary rooted tree T, with each vertex labeled by an index $i \in \{1, \ldots, N\}$, and the two children edges of a node labeled 0 and 1. Each leaf is labeled with an element of $\{0, 1\}$. T compute a Boolean function $\{0,1\}^N \to \{0,1\}$ as follows. On input x, if an internal vertex is labeled i, then the function reads the variable x_i; if $x_i = 0$ it proceeds down the edge labeled 0, and if $x_i = 1$ it proceeds down the edge labeled 1. When it reaches a leaf, it outputs the label of that leaf. The *depth* of a decision tree is the longest length of a path from the root to a leaf.

   (a) Show that if f is a Boolean function computed by a decision tree of depth d, then it can be written as a DNF (OR of ANDs of variables and their negations) where the terms are ANDs of at most d literals (literal=variable or negated variable).

(b) Show that if f is a Boolean function computed by a decision tree of depth d, then it can be written as a CNF (AND of ORs) where the clauses are ORs of at most d literals.

(Foreshadowing: the combination of the two parts of this exercise is analogous to the statement $\mathsf{P} \subseteq \mathsf{NP} \cap \mathsf{coNP}$. We'll make some of this precise in subsequent exercises.)

2. (a) Suppose $M^{\square}$ is a polynomial-time oracle Turing machine. Consider $M^A(1^n)$ as a function $\{0,1\}^{2^{n^q+1}-1} \to \{0,1\}$. If $M^A(1^n)$ queries strings of length at most n^q and runs in time n^t, show that this function is computed by a decision tree of depth n^t. Let $N \sim 2^{n^q}$; show the function we were considering is an N-bit function computed by a decision tree of depth $poly(\log N)$ (thus say that "Relativized P is the exponentially blown-up version of log-depth decision trees").

   (b) Suppose $M^{\square}$ is a *non-deterministic* polynomial-time oracle Turing machine. Consider $M^A(1^n)$ as a Boolean function as in the previous part. Suppose $M^A(1^n)$ uses n^k nondeterministic bits and runs in $O(n^t)$ time. Show that this Boolean function can be computed as an OR of $2^{O(n^k)}$-many decision trees, each of height at most $O(n^t)$. Let $N = 2^{n^t}$; show the Boolean function we're considering is computable as an OR of N decision trees each of height at most $O(\log N)$ (note that we must have $k \leq t$).

3. (a) Prove that the n-bit OR function cannot computed by a decision tree of depth $poly(\log n)$.

   (b) Use part (a) in combination with the previous exercise to give an alternative construction of an oracle A such that $\mathsf{P}^A \neq \mathsf{NP}^A$. (Well, really it is the same construction, but it is an alternative proof.)

4. (a) Prove that the N-bit PARITY function (=0 iff the number of input bits that are 1 is even) cannot be computed as an OR of $N^{poly(\log N)}$-many decision trees, each of height at most $poly(\log N)$.

   (b) Use part (a) in combination with Exercise 3 to give a(n alternative proof of the same) construction of an oracle B such that $\mathsf{PSPACE}^B \neq \mathsf{NP}^B$.

(The construction of an oracle separating NP from coNP at the start of Section 3 of Ko's survey uses a similar trick to the trick needed in this problem—look at the two cases near the bottom of p. 10.)

5. Build an oracle B such that $\mathsf{PSPACE}^B \neq \mathsf{NP}^B \neq \mathsf{P}^B$. *Hint:* Combine the construction from the previous problem with the construction we did in class of an oracle separating P from NP. Interleave the stages of the two constructions.

6. (More challenging) Build an oracle C such that $\mathsf{P}^C \neq \mathsf{NP}^C \cap \mathsf{coNP}^C$. *Hint:* One way to do this is to build C such that $\mathsf{P}^C \neq \mathsf{NP}^C = \mathsf{coNP}^C$ (which is in Du & Ko Theorem 4.20(b)), but that is far from the only way! If you want to try this route, try to first build an oracle "directly" that makes $\mathsf{NP} = \mathsf{coNP}$ (that is, don't just use a PSPACE oracle and make them both equal to PSPACE, but build the oracle inductively by stages.) In Du & Ko, you can use SAT^C instead of the "resource-bounded halting language" K_C that they use. Relativized Boolean Satisfiability can be defined this way:

   > Boolean Satisfiability with oracle C
   > *Input:* A Boolean formula built from variables $x_1, \ldots, x_n$, and OR, AND, NOT, and ORACLE gates. An ORACLE gate can take any number of inputs, and $\mathrm{ORACLE}(q_1, \ldots, q_m) = C(q_1 q_2 \ldots q_m)$. That is, the oracle gate treats its input as a string q, and outputs 1 or 0 according to whether q is in the oracle C or not.
   > *Decide:* Is there an assignment to the variables that makes the Boolean formula true?