15-455: UCT **K. Sutner**

Assignment 10 Due: **April 21**, 2024.

1. Boolean Polynomials (30)

Background

It is often helpful to express Boolean logic in terms of arithmetic. To this end, define a Boolean polynomial to be a multivariate polynomial $p(\boldsymbol{x}) \in \mathbb{Z}[\boldsymbol{x}]$ such that $p(\boldsymbol{a}) \in \mathbf{2}$ as long as $\boldsymbol{a} \in \mathbf{2}^n$. Thus p represents a Boolean function $\widehat{p} : \mathbf{2}^n \to \mathbf{2}$.

Note that for a Boolean polynomial we can replace x^k by x without changing the corresponding Boolean function. Let's call the operation of expanding a polynomial and reducing the powers in monomials linearization; the result is a multilinear polynomial.

As usual, when we refer to a polynomial we all implicit expressions such as $p_1 \cdot p_2$; if we need to use the explicit coefficient list form, we will say so.

Task

A. Determine short Boolean polynomials for n-ary conjunctions and disjunctions.

B. Determine smashed Boolean polynomials for n-ary disjunctions, exclusive disjunctions and the counting formula "exactly one out of."

C. Show that for every Boolean function $f : \mathbf{2}^n \to \mathbf{2}$ there exists exactly one multilinear Boolean polynomial that represents it.

D. Devise a fast test to check whether a polynomial in coefficient form is Boolean. Explain the running time of your test.

E. Let p be a Boolean polynomial. Observe that $\widehat{p}$ is constant 1 iff $2(1 - p)$ is a Boolean polynomial.
Did we just prove that co-$\mathbb{NP} = \mathbb{P}$?

2. Wurzelbrunft SAT (30)

Background

Prof. Dr. Alois Wurzelbrunft is currently fascinated by randomized algorithms. He thinks that checking satisfiability of a CNF formula is best done by picking values for the variables at random. Say, $\varphi(x_1, x_2, \ldots, x_n)$ is the given formula. The algorithm is beautifully simple:

> Pick a random vector $\boldsymbol{b} \in \mathbf{2}^n$ and return $\varphi(\boldsymbol{b})$.

Wurzelbrunft is vaguely aware that there is a minor problem: the algorithm will produce false negatives: the formula is satisfiable, but we get No instead. He thinks that can be handled as usual by repeating the test an appropriate number r of times.

Task

A. Show that the probability of a false negative may be as high as $1 - 2^{-n}$.

B. Suppose we repeat the algorithm r times, independently. Estimate the improved probability of false negatives.

C. What professional advice can you give Wurzelbrunft? Explain.

Comment

For part (B), use the fact that for small δ we have $(1 - \delta)^r \approx e^{-r\delta}$.

3. ZPP (20)

Background

Ordinarily Turing machines for decision problems are required to return yes or no. Let's generalize slightly to allow for an additional output ¿, meaning "don't know." We'll call these machines ambiguous, think of them as having three different halting states. We are interested in probabilistic ambiguous machines $\mathcal{M}$ that run in polynomial time and satisfy

$$\Pr[\mathcal{M}(x) \in \{L(x), ¿\}] = 1$$
$$\Pr[\mathcal{M}(x) = ¿] \leq 1/2$$

So they never give a wrong answer but may return ¿. As usual we identify a language L with its characteristic function.

Task

A. Show that ZPP is the set of languages accepted by a ¿-machine.

B. Show that $\mathsf{ZPP} = \mathsf{RP} \cap \text{co-}\mathsf{RP}$.

Comment

You can use part (A) for (B), but a direct proof is also possible.

4. Closure (30)

Background

As usual in the study of complexity classes, one tries to establish closure properties of probabilistic classes with respect to Boolean operations. For example, we know that BPP is a Boolean algebra. Here are some more exotic closure properties.

Task

A. Is BPP closed under polynomial time reductions? Explain.

B. Is BPP closed under Kleene star? Explain.

5. What If? (30)

Background

The possibility of derandomization suggests that $\mathbb{P} = \mathsf{BPP}$ is a reasonable conjecture. In this case, case we would expect $\mathbb{NP}$ to be strictly larger. Here is a corresponding "what if" scenario that explores a consequence of the assumption that $\mathbb{NP}$ is actually contained in BPP.

Task

A. Sketch and encoding of Boolean formulae in which the size of a formula does not change when a variable is replaced by a truth value.

B. Show that $\mathbb{NP} \subseteq \mathsf{BPP}$ implies that $\mathbb{NP} = \mathsf{RP}$.

C. How plausible is this consequence?

Comment Part (A) makes the argument in part (B) a little easier.

6. BPP and PH (30)

Background

In this question we will show that $\mathsf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$. To this end, we need a few combinatorial facts.

Call $S \subseteq \mathbf{2}^m$ yuge if $|S| \geq (1 - 1/(2m))\, 2^m$, and teeny if $|S| \leq 1/(2m)\, 2^m$. Given $z \in \mathbf{2}^m$, define the shift of S by z:

$$z \oplus S = \{\, z \oplus x \mid x \in S \,\}.$$

where $\oplus$ denotes bit-wise xor. Given $\boldsymbol{z} = (z_1, \ldots, z_m) \in (\mathbf{2}^m)^m$, write $\boldsymbol{z} \oplus S = \bigcup_i z_i \oplus S$ for the union of all the individual shifts. We will see that for yuge S, there exists $\boldsymbol{z}$ such that $\boldsymbol{z} \oplus S$ is all of $\mathbf{2}^m$.

Task

A. Suppose S is teeny. Show that for all $\boldsymbol{z}$ we have $\boldsymbol{z} \oplus S \neq \mathbf{2}^m$.

B. Suppose S is yuge. Show that there exists $\boldsymbol{z}$ such that $\boldsymbol{z} \oplus S = \mathbf{2}^m$.

C. Show that $\mathsf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$.

D. Conclude that $\mathbb{P} = \mathbb{NP}$ implies that $\mathbb{P} = \mathsf{BPP}$.

Comment

(A) is just counting, for (B) use a probabilistic existence proof. For (C), use the previous results and translate membership in L into yugeness (associate each instance x with a suitable set S_x).