CS 151 Complexity Theory **Spring 2025**

Problem Set 3

Out: April 17 *Due:* **April 24**

Reminder: you are encouraged to work in groups of two or three; however you must turn in your own write-up and note with whom you worked. You may consult the course notes and the optional text (Papadimitriou). The full honor code guidelines and collaboration policy can be found in the course syllabus.

Please attempt all problems. **Please turn in your solutions via Gradescope, by 1pm on the due date.**

1. Barrington's Theorem. We begin with a few words of background before the actual problem. A *branching program* is a directed acyclic graph in which each node is labelled by a variable x_i, one of these is designated as the *start node* and there are two special nodes labelled "accept" and "reject." All of the nodes labelled with variables have exactly two outgoing edges, one labelled "0" and the other labelled "1". An input $x = x_1x_2 \dots x_n$ defines a path from the start node to the accept or reject node as follows: at every node labelled x_i, we follow the outgoing edge whose label coincides with the value of x_i in the input. If we reach the accept node, the input is accepted; if we reach the reject node, the input is rejected. Polynomial-size branching programs capture **L/poly** in the same way that polynomial-size circuits capture **P/poly**.

   In this problem we will consider a *very* restricted subclass of polynomial-size branching programs. With the exception of the accept and reject nodes, all of the nodes will be divided into levels $\ell_1, \ell_2, \dots \ell_m$, with each level containing *at most 5 nodes*; the only permitted edges are directed from a node in level ℓ_i to a node in level ℓ_{i+1}, or a node in level ℓ_m to either the accept or reject nodes. Before Barrington's result in 1986, people were pretty sure these *width 5 branching programs* were similar in power to finite automata – it was thought that they could not even maintain counters during their computation. In this problem you will prove, in stark contrast to this intuition, that width 5 branching programs contain non-uniform $\mathbf{NC}_1$, and that in fact they exactly characterize non-uniform $\mathbf{NC}_1$. (Non-uniform $\mathbf{NC}_1$ is the class of languages decided by polynomial-size, $O(\log n)$-depth Boolean circuits).

   (a) Recall that S_5 is the group of permutations on the elements $\{1, 2, 3, 4, 5\}$. We will specify a sequence of m *instruction* triples (i_j, σ_j, τ_j), $j = 1 \dots m$, where $\sigma_j, \tau_j \in S_5$ and $1 \leq i_j \leq n$. On an input $x = x_1x_2 \dots x_n$ the instructions *yield* the permutation $\pi_1\pi_2 \dots \pi_m$, where $\pi_j = \sigma_j$ if $x_{i_j} = 0$ and $\pi_j = \tau_j$ if $x_{i_j} = 1$. We say that the sequence of instructions *π-accepts* a set $A \subseteq \{0,1\}^n$ if every $x \in A$ yields π and every $x \notin A$ yields the identity permutation e (and $\pi \neq e$). Verify that if there is a sequence of m instructions that π-accepts A, then there is a width-5 branching program with m levels that accepts A.

(b) Recall that every permutation can be written as the product of disjoint cycles. We will be concerned with elements of S_5 that are 5-cycles. Examples of these elements are $\sigma = (1\ 2\ 3\ 4\ 5)$ or its inverse $\sigma^{-1} = (1\ 5\ 4\ 3\ 2)$. Show that if π is a 5-cycle, and a sequence of m instructions π-accepts A, then for any 5-cycle $\pi' \in S_5$, there is a sequence of m instructions that π'-accepts A.

(c) Show that for any 5-cycle π, if there is a sequence of m instructions that π-accepts A then there is a sequence of m instructions that π-accepts the complement of A.

(d) Show that if π and π' are 5-cycles, and there is a sequence of m instructions that π-accepts A, and a sequence of m' instructions that π'-accepts B, then there is a sequence of $2(m+m')$ instructions that π''-accepts $(A \cap B)$, for some 5-cycle π''. You may use the fact that there exist 5-cycles σ and τ whose commutator $\sigma\tau\sigma^{-1}\tau^{-1}$ is a 5-cycle. (You may wish to verify this fact for yourself).

(e) Show that if π and π' are 5-cycles, and there is a sequence of m instructions that π-accepts A, and a sequence of m' instructions that π'-accepts B, then there is a sequence of $2(m + m')$ instructions that π''-accepts $(A \cup B)$, for some 5-cycle π''. Hint: write $A \cup B$ as an expression involving only complements and intersection, and use parts (c) and (d).

(f) Show that if A is decided by a fan-in 2, depth d Boolean circuit with $\wedge, \vee$ and $\neg$ gates, then there is a sequence of at most 4^d instructions that π-accepts A, for some 5-cycle π. Hint: induction on d. Conclude that every language in non-uniform $\mathbf{NC}_1$ has a polynomial-size width-5 branching program.

(g) Show that every language decided by polynomial-size width-5 branching programs is in non-uniform $\mathbf{NC}_1$. Conclude that the languages decided by polynomial-size width-5 branching programs are *exactly* non-uniform $\mathbf{NC}_1$.

2. The class $\mathbf{P/log}$ is the class of languages decidable by a Turing Machines running in polynomial time that take $O(\log n)$ bits of advice. Show that SAT$\in \mathbf{P/log}$ implies $\mathbf{P} = \mathbf{NP}$.

3. The parity function on n inputs is

$$\bigoplus_n (x_1, x_2, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

i.e., it is 1 if and only if there are an odd number of 1's among its n inputs. Recall that for a Boolean function f, $L(f)$ denotes the leaf-size of the smallest $(\wedge, \vee, \neg)$-formula that computes it.

(a) Show that $L(\bigoplus_n) \leq n^2$ when n is a power of 2.

(b) A *formal complexity measure* FC is a function mapping Boolean functions on n variables to the natural numbers, and satisfying the following properties:

i. $FC(x_i) = 1$ for $1 \leq i \leq n$

ii. $FC(f) = FC(\neg f)$ for all f

iii. $FC(f \vee g) \leq FC(f) + FC(g)$ for all f, g

Show that for *every* formal complexity measure FC we have $FC(f) \leq L(f)$. Thus $FC(f)$ is a lower bound on the formula complexity of f. Hint: pick an optimal formula for f, and use induction on $L(f)$.

(c) We will define a formal complexity measure (due to Krapchenko) that will allow us to prove a lower bound on $L(\bigoplus_n)$. Let A and B be subsets of $\{0,1\}^n$, and define

$$\begin{aligned} H(A,B) &= \{(a,b) : a \in A, b \in B, a \text{ and } b \text{ differ in exactly 1 coordinate}\} \\ K(f) &= \max_{A \subseteq f^{-1}(1),\, B \subseteq f^{-1}(0)} \frac{|H(A,B)|^2}{|A||B|}. \end{aligned}$$

Show that K is a formal complexity measure. The most difficult part is showing that property (iii) holds. For this part, let A and B be subsets maximizing the expression that defines $K(f \vee g)$, and partition A into $A_f \subseteq f^{-1}(1)$ and $A_g \subseteq g^{-1}(1)$; observe that $H(A_f, B) \cup H(A_g, B) = H(A, B)$...

(d) Show that $L(\bigoplus_n) \geq n^2$.