CS 2429 - Foundations of Communication Complexity

Lecture #5: 12 February 2014

Lecturer: Toniann Pitassi

Scribe Notes by: Toniann Pitassi

This lecture is about communication complexity approaches to proving complexity theoretic separations.

We will start with the connection between proving lower bounds for ACC and proving lower bounds for the "number-on-forehead" (NOF) model of communication complexity.

1 ACC^0 Lower Bounds and NOF Communication Complexity

In the k-player NOF communication complexity model, there are k players, trying to compute a Boolean function on nk boolean inputs. The nk boolean inputs are partitioned into k pieces each of size n, $x_1, \ldots, x_k$. For each $i \in [k]$, the i^{th} player can see all inputs *except* for input x_i. As in the two-party case, they communicate via an agreed-upon protocol, sending bits to a shared blackboard that are seen by all players. When it is player i's turn to speak, his/her message is a function of the inputs that he/she sees, and the contents of the blackboard so far. The cost of the protocol is the maximum number of bits that are sent, over all inputs of size nk.

For p a prime, an $AC^0[p]$ circuit of depth d is an AC circuit of depth d where besides AND, OR and negation, the mod p gate can be used, where $Mod_p(x_1, \ldots, x_n)$ is 1 if and only if $x_1 + \ldots, +x_n = 1(modp)$. The class ACC^0 is the union of classes $AC^0[p]$ for $p = 2, 3, 4, \ldots$.

An important characterization of ACC^0 proven by Yao [1990], and with a full proof provided by Beigel-Tarui [1994] is the following.

Theorem 1 *Any function computed by an ACC^0 circuit can also be represented by a polylog degree symmetric circuit. That is, by a depth 2 circuit in $x_1, \ldots, x_n$ where the top gate is a symmetric function, and the inputs to the top gate are r-conjunctions (of literals), where r is polylogarithmic in n.*

Using the above characterization it is easy to see that lower bounds on the NOF communication complexity of a Boolean function over n inputs, over some partition of the variables into k pieces, each piece of size n/k where k is polylogarithmic in n, would imply that this function is not in ACC^0. To see this, notice that if $f \in ACC^0$, then f can be written as a depth-2 polylog degree symmetric circuit, where the bottom gates are r-conjunctions of literals. Now fix any partition of the inputs into $k = r + 1$ pieces. By the pigeonhole principle, for each conjunction, there is at least one of the k players who can see all of the underlying literals of the conjunction. Thus we can partition the set of all r-conjunctions into k sets, $S_1, \ldots, S_k$ where the i^{th} set consists of conjunctions that the i^{th} player can evaluate by herself with no communcation. The protocol then

proceeds as follows. Player 1 first evaluates all of the conjunctions in S_1 and sends the number of these conjunctions that evaluate to true; then player 1 does the same thing for S_2 and so on. After all players have spoken (and notice that they can even speak in parallel), all players know the number of 1's feeding into the SYM gate, so they can all evalute the top gate, and therefore evaluate the function.

The best known lower bounds in the NOF model of communication complexity are due to Babai, Nisan and Szegedy. Unfortunately the lower bounds become trivial as soon as the number of players exceeds $\log n$. It is an important open problem to obtain lower bounds for more than $\log n$ players, even in the case where all players talk simultaneously, and after speaking the answer is known to all players. We note that using entirely different techniques, Ryan Williams recently proved a major result, that there is a function in $NEXP$ that is not in ACC^0. Still, proving the NOF lower bound for polylog n players is very important since it has the potential to separate ACC^0 from smaller complexity classes such as P, or NC.

1.1 Other Applications of NOF Communication Complexity

An important problem in data structures is to prove strong lower bounds for dynamic data structures. Mihai Patrascu formulated the following problem, whose resolution would yield a breakthrough polynomial time/space lower bound for many important dynamic data structures problems.

In this game there are three players, Alice, Bob and Charlie. Initially Bob holds m vectors $x_1, \ldots, x_m$, each of length n, and an index $i \in m$. Charlie holds one vector y, of length n, and the same index $i \in m$. They want to compute the set disjointness function on (x_i, y). At the start of the game, Alice sees $x_1, \ldots, x_m$ and y but not i, and is allowed to send one message privately to Alice. The length of Alice's message is $n^{1+\epsilon}$. The problem is to prove that for $m = poly(n)$ (say, $m = n^10$), that Bob and Charlie must communicate n^ϵ many bits in order to determine $DISJ(x_i, y)$.

Another application of NOF communication complexity is in proof complexity. For any proof system where the lines in the proof are expressible as degree d polyomials (over variables $x_1, \ldots, x_n$), NOF lower bounds for $d+1$ players implies tree-size lower bounds for the proof system.

2 An Approach toward Circuit Depth and Formula Size Lower Bounds via Communication Complexity

In this section we discuss Karchmer-Wigderson's approach to proving circuit depth lower bounds via communication complexity lower bounds. Because formulas can be balanced, circuit depth lower bounds implies formula *size* lower bounds.

Lemma 2 *Let F be a formula over the deMorgan basis of size s. Then there is another formula F' computing the same function as F and of depth $O(\log s)$.*

Proof Assume that our formulas have negations pushed to the leaves, and let the size of a formula be the number of leaves. Since F has size s, it is possible to show that F will contain a subformula F' of size t where such that $s/3 \leq t \leq 2s/3$. To see this, start at the top and think of a tree with two subtrees: left and right. If either left or right subtree is of size between 1/3 and 2/3

of the size of the whole tree, then we are done. Otherwise, pick the subtree that is bigger than 2/3 of the size of the original tree, and continue with that subtree. Sooner or later, we will come across a first subtree whose size is in the required range, since the parent which was passed up had size more than $2s/3$, so the larger of the two subtrees has size at least $s/3$. Let F' be this subformula, and let F'' denote the formula F but with a new variable z in place of F'. Now notice that $F = (F''|_{z=0} \wedge \neg F') \vee (F''_{z=1} \wedge F')$. This formula computes the same function as F. Now we recursively re-balance the subformulas $F|_{z=0}$, $F|_{z=1}$, and F''. Then we plug the resulting balanced formulas into the right-hand side of the expression for F given above. Each recurisve call adds at most 3 to the depth of the formula. On the other hand, since after each recursive call the size of the formula shrinks by a factor 2/3, there can be at most $\log_{3/2} |F|$ nested recursive calls (i.e., the depth of the recursion is at most $O(\log n)$). Thus in total the depth of the formula obtained at the end of this recursive re-balancing will be $O(\log |F|)$.

Let $g(x, y)$ be a boolean function, $x, y \in \{0, 1\}^n$. In a communication protocol for g, there are two players, Alice and Bob. Alice receives x and Bob receives y and they communicate via an agreed-upon protocol in order to compute the function $g(x, y)$. (The protocol proceeds in rounds, alternating who speaks. In each round, Alice's message sent is a function of her input x and the communication history thus far, and similarly Bob's next message is a function of his input y and the communication history so far.) We assume without loss of generality that the last bit of the communication history is the value of $g(x, y)$. The communication complexity of a protocol for f is the maximum number of bits exchanged by the protocol, over all inputs x, y, $|x| = |y| = n$. The communication complexity of f, $CC(f)$, is the minimum over all protcols Π for f, of the communication complexity of Π.

Similarly, we can define the communication complexity of a search problem $S(x, y)$. For each x, y, the players communicate via an agreed-upon protocol, and the last message sent is some z such that $z \in S(x, y)$. Again the communication complexity of a protocol is the maximum number of bits exchanged over all x, y, $|x| = |y| = n$, and the communication complexity of S, $CC(S)$, is the minimum over all protocols Π for S.

Karchmer and Wigderson proved that there is a tight relationship between the depth of a circuit for computing a boolean function, and the communication complexity of a related search problem.

Let $f(x)$ be a Boolean function, $|x| = n$. The search problem associated with f, $S_f(x, y)$ is defined as follows. If $f(x) \neq f(y)$ then $S_f(x, y)$ is equal to all indices $i \in [n]$ such that $x_i \neq y_i$. Otherwise, if $f(x) = f(y)$, then $S_f(x, y)$ contains all $i \in [n]$.

We can also define a more restricted search problem in the case where f is monotone. (A Boolean function is monotone if flipping any of the coordinates from 0 to 1 does not decrease the value of the function.) For monotone f, the monotone search problem associated with f, $S_f^{mono}(x, y)$ is defined as follows. If $f(x) = 1$ and $f(y) = 0$ then $S_f^{mono}(x, y)$ contains all $i \in [n]$ such that $x_i = 1$ and $y_i = 0$. Otherwise $S_f^{mono}(x, y)$ contains all $i \in [n]$.

Theorem 3 *The circuit depth of f is equal to the communication complexity of S_f. That is, $depth(f) = \Theta(CC(S_f))$. Furthermore, if f is monotone, then the monotone circuit depth of f is equal to the communicatiaon complexity of S_f^{mono}. That is, $monotone-depth(f) = \Theta(CC(S_f^{mono}))$.*

Proof We will first show that $CC(S_f) \leq depth(f)$ by constructing a protocol for S_f. Let C be a depth d circuit computing f, and assume without loss of generality that the top gate of the circuit is an AND gate. (The argument is very similar if the top gate is an OR gate.) Assume that Alice

has x and Bob has y, where $f(x) = 1$ and $f(y) = 0$, and they want to find an input i such that $x_i \neq y_i$. Because $f(x) = 1$ both inputs to the top AND gate must have value 1 on input x, and because $f(y) = 0$, at least one input to the top AND gate has value 0 on input y. Bob speaks first, and sends a 0 to Alice if the left input to the AND gate evaluates to 0 on y, and otherwise sends a 1 if the right input to the AND gate evaluates to 0 in y. In the next round, we have recursed to subcircuit of the original circuit, of depth one less than the original depth, and they begin this round again knowing that the subcircuit on x evaluates to 1, and the subcircuit on y evaluates to 0. Since the next gate is an OR gate, this time Alice speaks, sending a 0 if the left subcircuit of the current subcircuit evaluates to 1 on x, and otherwise sending a 0 to Bob, indicating that the right subcircuit evaluates to 1 on x. Alice and Bob continue in this manner until eventually they reach an input. By induction, it is clear that they will reach an input and on this bit, x_i and y_i will be different. The number of bits communicated by Alice and Bob is equal to the depth of the circuit.

In the case where f is monotone, and the circuit for f of depth d is also monotone, they follow the same protocol, but because there are no negations, we are guaranteed that when they arrive at a leaf node, that $x_i = 1$ and $y_i = 0$, as desired.

To show the other direction we want to convert a protocol for $S_f(x, y)$ into a Boolean circuit for f. To prove this, we define a more general communication game. For any two disjoint sets $A, B \subseteq \{0,1\}^n$, denote by $G_{A,B}$ the following game. Alice gets $x \in A$, Bob gets $y \in B$ and the goal is to find a coordinate i such that $x_i \neq y_i$. We prove the following claim:

Claim: If $CC(G_{A,B}) = d$, then there is a Boolean function f such that: (i) $f(x) = 1$ for all $x \in A$; (ii) $f(y) = 0$ for all $y \in B$; (iii) $Depth(f) \leq d$.

That is, the function f is 1 on any input from A, 0 on any input from B and the circuit depth of f is at most d. The proof of the claim is by induction on $d = CC(G_{A,B})$. The base case is $d = 0$. That is, the two players know the answer without any communication, and therefore there is some coordinate i such that for every $x \in A$, and every $y \in B$, $x_i \neq y_i$. Thus the function $f(z) = z_i$ or $f(z) = \neg z_i$ satisfies the requirements of the game.

For the induction step, assume that we have a protocol of communication complexity d for the game $G_{A,B}$. Assume without loss of generality that Alice sends the first bit in the protocol. This bit partitions A into two disjoint sets $A = A_0 \cup A_1$. If the first bit is 0, the rest of the protocol is a protocol for the game $G_{A_0,B}$. If the first bit is 1, the rest of the protocol is a protocol for the game $G_{A_1,B}$. Hence, for both games, we have protocols with communication complexity at most $d - 1$. By the inductive hypothesis, we have two functions f_0 and f_1 that satisfy: (i) $f_0(x) = 1$ for all $x \in A_0$; (ii) $f_1(x) = 1$ for all $x \in A_1$; (iii) $f_0(y) = f_1(y) = 0$ for every $y \in B$; (iv) $Depth(f_0), Depth(f_1) \leq d - 1$. Define $f = f_0 \vee f_1$. Then it is clear that for every $x \in A$, $f(x) = f_0(x) \vee f_1(x) = 1$, and for every $y \in B$, $f(y) = f_0(y) \vee f_1(y) = 0$, and the depth of f is d.

Similarly if Bob sends the first bit, then B is partitioned into two disjoint sets $B = B_0 \cup B_1$. The argument is very similar to as before, except that now we have two functions g_0, g_1 corresponding to the two games G_{A,B_0}, G_{A,B_1}, and we take $g = g_0 \wedge g_1$.

This completes the proof by taking A to be $f^{-1}(1)$ and B to be $f^{-1}(0)$.

3 Monotone Depth Bounds

It was a major milestone to prove depth lower bounds for *monotone* circuits for functions in monotone P. We note that depth lower bounds were previously known via Razborov's monotone circuit size lower bounds, but his function (clique) is not believed to be in P. Karchmer and Wigderson used the communication complexity framework to prove $O(\log^2 n)$ depth lower bounds for monotone circuits computing st-connectivity. Later, Raz and McKenzie separated all levels of the monotone circuit hierarchy, proving that $mNC_i \neq mNC_{i+1}$ for all i.

Here we give a new proof of a strengthening of these results due to Goos and Pitassi, via a randomized reduction to the communication complexity of set disjointness!

We exhibit a monotone function on n variables whose monotone circuits require depth $\Omega(n/\log n)$; previously, a bound of $\Omega(\sqrt{n})$ was known (Raz and Wigderson, JACM 1992). Moreover, we prove a tight $\Theta(\sqrt{n})$ monotone depth bound for a function in monotone P.

We obtain new randomised lower bounds on the communication complexity of search problems. Our proofs are relatively simple reductions from the *set-disjointness* function, the canonical NP-complete problem in communication complexity. These results allow us to derive, almost for free, new lower bounds for monotone depth. We construct a monotone function on n variables whose monotone circuits require depth $\Omega(n/\log n)$. Previously, the best bound for an explicit monotone function (perfect matchings) was $\Omega(\sqrt{n})$ due to Raz and Wigderson. Moreover, we prove a tight $\Theta(\sqrt{n})$ monotone depth bound for a function in monotone P.

3.1 Starting point: Critical block sensitivity

We build on the techniques recently introduced by Huynh and Nordström [**?**]. They defined a new complexity measure for search problems called *critical block sensitivity*, which is a generalisation of the usual notion of block sensitivity for functions.

A *search problem* on n variables is a relation $S \subseteq \{0,1\}^n \times Q$ where Q is some set of possible solutions. On input $\alpha \in \{0,1\}^n$ the search problem is to find a solution $q \in Q$ that is *feasible for* α, that is, $(\alpha, q) \in S$. We assume that S is such that all inputs have at least one feasible solution. An input is called *critical* if it has a unique feasible solution.

Definition [Critical block sensitivity] Fix a search problem $S \subseteq \{0,1\}^n \times Q$. Let $f \subseteq S$ be a total function, i.e., for each input $\alpha \in \{0,1\}^n$ the function picks out some feasible solution $f(\alpha)$ for α. We denote by $\mathrm{bs}(f,\alpha)$ the usual block sensitivity of f at α. That is, $\mathrm{bs}(f,\alpha)$ is the maximal number bs such that there are disjoint blocks of coordinates $B_1, \ldots, B_{\mathrm{bs}} \subseteq [n]$ satisfying $f(\alpha) \neq f(\alpha^{B_i})$ for all i; here, α^{B_i} is the same as α except the input bits in coordinates B_i are flipped. The *critical block sensitivity* of S is defined as:

$$bscrit(S) = min_{f \subseteq S} max_{\mathrm{critical}\alpha} bs(f,\alpha).$$

3.2 Composed search problems

In order to study a search problem $S \subseteq \{0,1\}^n \times Q$ in the setting of two-party communication complexity, we need to specify how the n input variables of S are divided between the two players, Alice and Bob.

Unfortunately, for many search problems (and functions) there is often no partition of the variables that would carry the "intrinsic" complexity of S over to communication complexity.

In order to make the function hard for communication complexity, one usually studies *composed* (or *lifted*) variants $S \circ g^n$ of the original problem. In a composed problem, each of the n input bits of S are encoded using a small two-party function $g\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, sometimes called a *gadget*. As input to $S \circ g^n$ Alice gets an $x \in \mathcal{X}^n$ and Bob gets a $y \in \mathcal{Y}^n$. We think of the pair (x, y) as encoding the input

$$\alpha = g^n(x,y) = (\, g(x_1,y_1), \ldots, g(x_n,y_n)\,)$$

of the original problem S. The objective is to find a $q \in Q$ such that $(g^n(x,y), q) \in S$.

3.3 Our communication complexity results

Theorem 4 *There is a two-party gadget $g\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ such that if $S \subseteq \{0,1\}^n \times Q$ is any search problem, then $S \circ g^n$ has randomised bounded-error communication complexity $\Omega(bscrit(S))$.*

Our key idea is to choose g to be *random-self-reducible*. Random-self-reducibility is a notion often studied in cryptography and classical complexity theory, but less often in communication complexity.

3.4 CSPs and their canonical search problems

Definition [d-CSPs] A CSP F consists of a set of (boolean) variables $vars(F)$ and a set of constraints $cons(F)$. Each constraint $C \in cons(F)$ is a function that maps a truth assignment $\alpha\colon vars(F) \to \{0,1\}$ to either 0 or 1. If $C(\alpha) = 1$, we say that C is *satisfied* by α, otherwise C is *violated* by α. Let $vars(C)$ denote the smallest subset of $vars(F)$ such that C depends only on the truth values of the variables in $vars(C)$. We say that F is of *degree* d, or F is a d-CSP, if $|vars(C)| \leq d$ for all C. Note that d-CNF formulas are a special case of d-CSPs, and conversely, each d-CSP can be written as an equivalent d-CNF with a factor 2^d blow-up in the number of constraints.

An *unsatisfiable* CSP F has no assignment that satisfies all the constraints. Each such F comes with an associated *canonical search problem* $S(F)$.

Definition [Canonical search problems] Let F be an unsatisfiable CSP. In the search problem $S(F)$ we are given an assignment $\alpha\colon vars(F) \to \{0,1\}$ and the goal is to find a constraint $C \in cons(F)$ that is violated by α.

We give new critical block sensitivity lower bounds for the canonical search problems associated with *Tseitin* and *Pebbling* formulas.

3.5 Sensitivity of Tseitin formulas

Tseitin formulas are well-studied examples of unsatisfiable CSPs that are hard to refute in many proof systems.

Definition [Tseitin formulas] Let $G = (V, E, \ell)$ be a connected labelled graph of maximum degree d where the labelling $\ell: V \to \{0,1\}$ has odd Hamming weight. The *Tseitin formula* Tse_G associated with G is the d-CSP that has the edges $e \in E$ as variables and for each node $v \in V$ there is a constraint C_v defined by

$$C_v(\alpha) = 1 \leftrightarrow \sum_{e:v \in e} \alpha(e) \equiv \ell(v) \pmod 2.$$

It follows from a simple parity argument that Tse_G is unsatisfiable.

Theorem 5 *If G is a sufficiently strong constant degree expander (e.g., a Ramanujan graph [**?**]), then the critical block sensitivity of $S(Tse_G)$ is $\Omega(n/\log n)$.*

3.6 Applications: Monotone depth

Raz and McKenzie [**?**] developed a general framework to prove monotone depth lower bounds for many monotone functions. We borrow the following piece from their machinery. Here we denote by $depth(f)$ the minimum depth of a monotone circuit computing f.

Theorem 6 (Raz–McKenzie transformation) *Let $g: \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a two-party gadget and let F be an unsatisfiable d-CSP on n variables and m constraints. There is an explicit construction of a monotone function $f: \{0,1\}^N \to \{0,1\}$ on $N = m|\mathcal{X}|^d$ inputs such that $depth(f)$ is lower bounded by the (deterministic) communication complexity of $S(F) \circ g^n$.*

Monotone depth from Tseitin. First, let G be a sufficiently strong constant degree d expander graph. Then $S(Tse_G)$ is the canonical search problem associated with a d-CSP on $O(n)$ variables and n constraints. Theorems 4 and 5 tell us that $S(Tse_G) \circ g^n$ has two-party communication complexity $\Omega(n/\log n)$.

Corollary 7 (Monotone depth from Tseitin) *There is an explicit monotone function f on N inputs such that $depth(f) = \Omega(N/\log N)$.*

4 Versatile Gadgets

In this section we introduce *versatile* two-party functions.

4.1 Self-reductions and versatility

The simplest reductions between communication problems are those that can be computed without communication. Let $f_i: \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ for $i = 1, 2$, be two-party functions. We say that f_1 *reduces to* f_2, written $f_1 \leq f_2$, if the communication matrix of f_1 appears as a submatrix of the communication matrix of f_2. Equivalently, $f_1 \leq f_2$ iff there exist one-to-one mappings π_A and π_B such that

$$f_1(x, y) = f_2(\pi_A(x), \pi_B(y))$$

for all $(x, y) \in \mathcal{X}_1 \times \mathcal{Y}_1$.

Our restriction to one-to-one reductions above is merely a technical convenience.

We will be interested in special kinds of reductions that reduce a function to *itself*. Our first flavour of self-reducibility relates a function f and its negation $\neg f$: A function f is called *flippable* if $\neg f \leq f$. Note that since the associated reduction maps z-inputs to $(1-z)$-inputs in a one-to-one fashion, a flippable function must be *balanced*: exactly half of the inputs satisfy $f(x,y) = 1$.

A function f is called *random-self-reducible* if there are mappings π_A and π_B together with a random variable r such that for every z-input $(x,y) \in f^{-1}(z)$ the random pair $(\pi_A(x,r), \pi_B(y,r))$ is uniformly distributed among all the z-inputs of f.

Definition [Versatility] A two-party function g is called *versatile* if (1) $g \geq$ AND, (2) g is flippable, and (3) g is random-self-reducible.

Consider the function VER: $Z_4 \times Z_4 \to \{0,1\}$ defined by

$$\text{VER}(x,y) = 1 \leftrightarrow x + y \in \{2,3\},$$

for all $x, y \in Z_4$, where the arithmetic is that of Z_4.

Lemma 8 VER *is versatile.*

Proof The reduction from AND is simply given by $\text{AND}(x,y) = \text{VER}(x,y)$. Moreover, VER is flippable because $\neg\text{VER}(x,y) = \text{VER}(x+2,y)$. To see that VER is random-self-reducible, start with (x,y) and compute as follows. First, choose (x,y) uniformly at random from the set $\{(x,y),(1-x,-y)\}$ so that $x+y$ is uniformly distributed either in the set $\{0,1\}$ if (x,y) was a 0-input, or in the set $\{2,3\}$ if (x,y) was a 1-input. Finally, choose a random $a \in Z_4$ and output $(x+a, y-a)$.

5 Communication Lower Bound

In this section we prove the communication lower bound (Theorem 4) assuming that g is a versatile gadget.

Setup. Fix any versatile gadget $g: \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. Let Π be a randomised ϵ-error protocol for a composed search problem $S \circ g^n$. Recall that an input (x,y) for the problem $S \circ g^n$ is *critical* if there is exactly one solution q with $((x,y),q) \in S \circ g^n$. In particular, if $g^n(x,y)$ is critical for S, then (x,y) is critical for $S \circ g^n$. The behaviour of the protocol Π on a critical input (x,y) is predictable: the protocol's output $\Pi(x,y)$ is the unique solution with probability at least $1-\epsilon$.

However, noncritical inputs (x,y) are much trickier: not only can the distribution of the output $\Pi(x,y)$ be complex, but the distributions of $\Pi(x,y)$ and $\Pi(x',y')$ can differ even if (x,y) and (x',y') encode the same input $g^n(x,y) = g^n(x',y')$ of S. The latter difficulty is the main technical challenge, and we address it by using random-self-reducible gadgets.

Defining a function $f \subseteq S$. First, we record for each $\alpha \in \{0,1\}^n$ the *most likely feasible output* of Π on inputs (x,y) that encode α. More formally, for each α we define μ_α to be the uniform distribution on the set of preimages of α, i.e., μ_α is uniform on $\{(x,y) : g^n(x,y) = \alpha\}$. Alternatively, this can be viewed as a product distribution $\mu_\alpha = \mu_{\alpha_1} \times \mu_{\alpha_2} \times \cdots \times \mu_{\alpha_n}$, where μ_z, $z \in \{0,1\}$, is the uniform distribution on $g^{-1}(z)$.

The most likely feasible solution output by Π on inputs $(x,y) \sim \mu_\alpha$ is now captured by a total function $f \subseteq S$ defined by

$$f(\alpha) \ = argmax_{q:(\alpha,q)\in S} \Pr_{(x,y)\sim\mu_\alpha}[\,\Pi(x,y) = q\,].$$

Here, ties are broken arbitrarily and the randomness is taken over both $(x, y) \sim \mu_\alpha$ and the random coins of the protocol Π. (Note that, in general, the most likely output of $\Pi(x, y)$ may not be feasible. However, above, we explicitly pick out the most likely *feasible* solution. Thus, f is indeed a subfunction of S.)

The sensitive critical input. We can now use the critical block sensitivity of S: there is a critical input α such that $bs(f, \alpha) \geq bscrit(S)$. Let $B_1, \ldots, B_{\text{bs}} \subseteq [n]$ be the sensitive blocks with $f(\alpha^{B_i}) \neq f(\alpha)$.

Lemma 9 *The protocol* Π *can distinguish between* μ_α *and* $\mu_{\alpha^{B_i}}$ *in the sense that*

$$(x, y) \sim \mu_\alpha \rightarrow \Pr[\,\Pi(x, y) = f(\alpha)\,] \geq 1 - \epsilon,$$

$$(x, y) \sim \mu_{\alpha^{B_i}} \rightarrow \Pr[\,\Pi(x, y) = f(\alpha)\,] \leq 1/2.$$

Proof The consequent in the first property (9) is true even for each individual (x, y) in the support of μ_α since α is critical. To see that the second property (9) is true, suppose for a contradiction that we had $\Pr[\,\Pi(x, y) = f(\alpha)\,] > 1/2$ for $(x, y) \sim \mu_{\alpha^{B_i}}$. By averaging, there is a fixed input (x, y) in the support of $\mu_{\alpha^{B_i}}$ such that $\Pr[\,\Pi(x, y) = f(\alpha)\,] > 1/2$. By the correctness of Π (i.e., $1-\epsilon > 1/2$) this implies that $f(\alpha)$ is feasible for α^{B_i}. Thus, $f(\alpha)$ is the most likely feasible solution output by $\Pi(x, y)$, that is, $f(\alpha^{B_i}) = f(\alpha)$ by the definition of f. But this contradicts the fact that f is sensitive to B_i at α.

The reduction. Given an input (a, b) for $UDISJ_{bs}$ our goal is to describe a randomised reduction $(a, b) \mapsto (x, y)$ such that

(P1) 0-inputs: If $UDISJ_{bs}(a, b) = 0$, then $(x, y) \sim \mu_\alpha$.

(P2) 1-inputs: If $UDISJ_{bs}(a, b) = 1$ with $a_i = b_i = 1$, then $(x, y) \sim \mu_{\alpha^{B_i}}$.

Suppose for a moment that we had a reduction with properties (P1–P2). Let Π' be the protocol that on input (a, b) first applies the reduction $(a, b) \mapsto (x, y)$ with properties (P1–P2), then runs Π on (x, y), and finally outputs 0 if $\Pi(x, y) = f(\alpha)$ and 1 otherwise. Lemma (distinguish) tells us that

- If $UDISJ_{bs}(a, b) = 0$, then $\Pi'(a, b) = 0$ with probability at least $1 - \epsilon$.
- If $UDISJ_{bs}(a, b) = 1$, then $\Pi'(a, b) = 1$ with probability at least $1/2$.

The error probability of Π' can be bounded away from $1/2$ by repeating Π' twice and outputting 0 iff both runs of Π' output 0. (Here we are assuming that ϵ is small enough, say at most $1/4$. If not, we can use some other standard success probability boosting tricks.) This gives a randomised protocol for $UDISJ_{bs}$ with the same communication cost (up to constants) as that of Π. Our main theorem follows.

Indeed, it remains to implement a reduction $(a, b) \mapsto (x, y)$ satisfying (P1–P2). We do it in three steps.

Step 1. On input $(a, b) = (a_1 \dots a_{\text{bs}}, b_1 \dots b_{\text{bs}})$ to $UDISJ_{bs}$ we first take each pair (a_i, b_i) through the reduction AND $\leq g$ to obtain instances $(a'_1, b'_1), \dots, (a'_{\text{bs}}, b'_{\text{bs}})$ of g. Note that

- if $UDISJ_{bs}(a, b) = 0$, then $g(a'_i, b'_i) = 0$ for all i;
- if $UDISJ_{bs}(a, b) = 1$, then there is a unique i with $g(a'_i, b'_i) = 1$.

Step 2. Next, the instances (a'_i, b'_i) are used to populate a vector $(x, y) = (x_1 \dots x_n, y_1 \dots y_n)$ carrying n instances of g, as follows. The instance (a'_i, b'_i) is plugged in for the coordinates $j \in B_i$ with the copies corresponding to $\alpha_j = 1$ flipped. That is, we define for $j \in B_i$:

- if $\alpha_j = 0$, then $(x_j, y_j) := (a'_i, b'_i)$;
- if $\alpha_j = 1$, then $(x_j, y_j) := (\pi_A(a'_i), \pi_B(b'_i))$, where (π_A, π_B) is the reduction $\neg g \leq g$.

For $j \notin \cup_i B_i$ we simply fix an arbitrary $(x_j, y_j) \in g^{-1}(\alpha_j)$. We now have that

- if $UDISJ_{bs}(a, b) = 0$, then $g^n(x, y) = \alpha$;
- if $UDISJ_{bs}(a, b) = 1$ with $a_i = b_i = 1$, then $g^n(x, y) = \alpha^{B_i}$.

Step 3. Finally, we apply a random-self-reduction independently for each component (x_i, y_i) of (x, y): this maps a z-input (x_i, y_i) to a uniformly random z-input $(x_i, y_i) \sim \mu_z$. The result is a random vector (x, y) that has a distribution of the form $\mu_\alpha = \mu_{\alpha_1} \times \mu_{\alpha_2} \times \dots \times \mu_{\alpha_n}$ and matches our requirements (P1–P2), as desired.

This concludes the proof.

6 The Composition Conjecture

Karchmer, Raz and Wigderson suggested the following approach towards proving that NC_1 is not contained in P. Given two boolean functions f and g, the depth complexity of the composed function gof is roughly the sum of the depth complexities of f and g.

KRW Conjecture: Let $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$. Then the depth of gof is equal to the depth of f plus the depth of g.

They showed that proving this conjecture would imply $P \neq NC_1$. The basic idea is that one could apply $O(\log n)$ compositions of a random function $f : \{0,1\}^{\log n} \to \{0,1\}$ thus obtaining a new function over n bits that is computable in polynomial time yet requires depth $\Omega(\log^2 n)$. The key point is that a random function on $\log n$ bits has depth complexity $\log n - o(\log n)$ and can be described explicitly using n bits.

What does the KW relation R_{gof} look like? Alice and Bob each get as inputs m-by-n matrices, X and Y respectively, such that $f(X) \in g^{-1}(1)$ and $f(Y) \in g^{-1}(0)$ and their goal is to find an entry (j, i) such that $X_{j,i} \neq Y_{j,i}$. Clearly $CC(R_{gof}) \leq CC(R_g) + CC(R_f)$. TO see this, Alice and Bob first use the optimal protocol of g on inputs $f(X)$ and $f(Y)$ to find an index $j \in [m]$ such that $f(X_j) \neq f(Y_j)$. Then they use the optimal protocol on f on inputs $f(X_j)$ and $f(Y_j)$ to find a coordinate i on which the j-th rows differ, thus obtaining an entry (j, i) on which X and Y differ.

7 Open Problems

The biggest open problem is to obtain depth lower bounds for nonmonotone circuits, which as we saw above, is equivalent to proving communication lower bounds for a corresponding search problem.

Another open problem is to resolve the depth complexity of monotone circuits for majority. The complexity of monotone formulas/circuits for the majority function is a fascinating topic. Without the monotonicity restriction, majority can be solved with simple linear-size circuits of depth $O(\log n)$, where the best known depth is $4.95 \log n + O(1)$. There are two fundamental algorithms that give log depth, monotone circuits. The first is a beautiful construction by vValiant achieving monotone formulas of depth $5.3 \log n + O(1)$ and size $O(n^{5.3})$. The second is obtained from the celebrated AKS sorting network, giving a completely uniform construction of depth $K \log n$ and size $O(n \log n)$. Unfortunately, K is an enormous constant – about 5000, and the proof is quite complicated. Converting the circuit to a formula yields a monotone formula of size $O(n^K)$ which is roughly n^{5000}!

Getting the best depth-size tradeoffs is perhaps the most sought after goal around the classical question, which achieving uniformity comes next.

The problem can be naturally split into two subproblems, which together solve the original problem. Problem I takes an n-bit vector and outputs an m-bit vector such that if the original vector contained a majority of 1's, then the output vector contains at least 2/3 fraction of 1's, and similarly if the original vector contains less than a majority of 1's, then the output vector has at most a 1/3 fraction of 1's. Problem II is the promise problem of solving majority on the bounded-away-from-1/2 inputs given by the output of problem I. (Note that Problem II is an approximate counting problem.) Subproblem II has a uniform monotone circuit of linear size, and the same depth as Valiant's solution. The monotone search problem associated with majority is as follows. Alice is given a vector x, such that the number of 1's in x is $n/2 + 1$ and Bob is given a vector y with $n/2$ 1's and they want to find an element $i \in [n]$ such that $x_i = 1$ and $y_i = 0$. The monotone communication complexity for a potentially easier promise problem is similar but now Alice is given a vector greater than $2n/3$ 1's and Bob is given a vector with less than $n/3$ 1's and they want to find some element in x but not in y. The promise problem can be solved in depth $2 \log n$. Can the non-promise problem be solved in depth $2 \log n$?

Another problem is to obtain explicit uniform formulas for majority of optimal or near optimal size. Toward this goal, can we come up with a natural (top-down) communication complexity protocol for mMaj that uses $O(\log n)$ many bits?