How to Compress Interactive Communication

Boaz Barak[*] Mark Braverman[†] Xi Chen[‡] Anup Rao[§]

November 10, 2009

Abstract

We describe new ways to simulate 2-party communication protocols to get protocols with potentially smaller communication. We show that every communication protocol that communicates C bits and reveals I bits of information to the participating parties can be simulated by a new protocol involving at most $\tilde{O}(\sqrt{CI})$ bits of communication. In the case that the parties have inputs that are independent of each other, we get much better results, showing how to carry out the simulation with $\tilde{O}(I)$ bits of communication.

These results lead to a direct sum theorem for randomized communication complexity. Ignoring polylogarithmic factors, we show that for worst case computation, computing n copies of a function requires $\sqrt{n}$ times the communication required for computing on copy of the function. For average case complexity, given any distribution μ on inputs, computing n copies of the function on n independent inputs sampled according to μ requires $\sqrt{n}$ times the communication for computing one copy. If μ is a product distribution, computing n copies on n independent inputs sampled according to μ requires n times the communication required for computing the function. We also study the complexity of computing the sum (or parity) of n evaluations of f, and obtain results analogous to those above.

As far as we know, our results give the first compression schemes for general randomized protocols and the first direct sum results in the general setting. Previous results applied only when the protocols were restricted to running in a constant number of rounds, where each message can be compressed in turn, and only applied when the parties are given independent inputs.

[*]Department of Computer Science, Princeton University, boaz@cs.princeton.edu. Supported by NSF grants CNS-0627526, CCF-0426582 and CCF-0832797, US-Israel BSF grant 2004288 and Packard and Sloan fellowships.

[†]Microsoft Research New England, mbraverm@cs.toronto.edu.

[‡]Department of Computer Science, Princeton University, csxichen@gmail.com. Supported by NSF Grants CCF-0832797 and DMS-0635607.

[§]Center for Computational Intractability, Princeton University, anuprao@cs.washington.edu. Supported by NSF Grant CCF 0832797.

Contents

1 Introduction

In this work, we address two questions: (1) Can we compress the communication of an interactive protocol so it is close to the information conveyed between the parties? (2) Is it harder to compute a function on n independent inputs than to compute it on a single input? In the context of communication complexity, the two questions are related, and our answer to the former will yield an answer to the latter.

Techniques for message compression, first considered by Shannon [Sha48], have had a big impact on computer science, especially with the rise of the Internet and data intensive applications. Today we know how to encode messages so that their length is essentially the same as the amount of information that they carry (see for example the text [CT91]). Can we get a similar savings in an interactive setting? A first attempt might be to simply compress each message of the interaction in turn. However, this compression involves at least 1 bit of communication for every message of the interaction, which can be much larger than the total information conveyed between the parties. In this paper, we show how compress interactive communication protocols in a way that is independent of the number of rounds of communication, and in some settings, give compressed protocols with communication that has an almost linear dependence on the information conveyed in the original protocol.

The second question is one of the most basic questions of theoretical computer science, called the *direct sum* question, and is closely related to the *direct product* question. A direct product theorem in a particular computational model asserts that the probability of success of performing n independent computational task decreases in n. Famous examples of such theorems include Yao's XOR Lemma [Yao82] and Raz's Parallel Repetition Theorem [Raz95]. In the context of communication complexity, Shaltiel [Sha03] gave a direct product theorem for the discrepancy of a function, but it remains open to give such a theorem for the success probability of communication tasks. A direct sum theorem asserts that the amount of resources needed to perform n independent tasks grows with n. While the direct sum question for general models such as Boolean circuits has a long history (cf [Uhl74, Pau76, GF81]), no general results are known, and indeed they cannot be achieved by the standard reductions used in complexity theory, as a black-box reduction mapping a circuit C performing n tasks into a circuit C' performing a single task will necessarily make C' *larger* than C, rather than making it smaller. Indeed it is known that at least the most straightforward/optimistic formulation of a direct sum theorem for Boolean circuits is *false*.[1]

Nevertheless, direct sum theorems are known to hold in other computational models. For example, an optimal direct sum theorem is easy to prove for decision tree depth. A more interesting model is *communication complexity*, where this question was first raised by Karchmer, Raz, and Wigderson [KRW91] who conjectured a certain direct sum result for deterministic communication complexity of relations, and showed that it would imply that $\mathbf{P} \not\subseteq \mathbf{NC}^1$. Feder, Kushilevitz, Naor, and Nisan [FKNN91] gave a direct sum theorem for non-deterministic communication complexity, and deduced from it a somewhat weaker result for deterministic communication complexity— if a single copy of a function f requires C bits of communications, then n copies require $\Omega(\sqrt{C}n)$ bits. Feder et al also considered the direct sum question for *randomized* communication complexity (see also Open Problem 4.6 in [KN97]) and showed that the dependence of the communication on the

[1]The example comes from fast matrix multiplication. By a counting argument, there exists an $n \times n$ matrix A over GF(2) such that the map $x \mapsto Ax$ requires a circuit of $\Omega(n^2/\log n)$ size. But the map $(x_1, \dots, x_n) \mapsto (Ax_1, \dots, Ax_n)$ is just the product of the matrices A and X (whose columns are $x_1, \dots, x_n$) and hence can be carried out by a circuit of $O(n^{2.38}) \ll n \cdot (n^2/\log n)$. See Shaltiel's paper [Sha03] for more on this question.

error of the protocol for many copies can be better than that obtained by the naive protocol for many copies.

Chakrabarti et al [CSWY01] gave a direct sum theorem in the case that the communication involves one simultaneous round of communication, while Jain et al [JRS03] (improved upon by [HJMR07]) gave a direct sum theorem for the distributional complexity of constant round randomized protocols when the inputs are assumed to be independent of each other. These results only apply when the number of rounds in the protocol is fixed to some constant, and so give no guarantees in the standard model, with unbounded number of rounds.

All of the works mentioned above, had a common outline that we follow in this work as well. They began by measuring the information that an observer learns about the inputs of the parties by watching the messages and public randomness of the protocol, a quantity that they called the *information cost* of the protocol. Formally, the information cost was defined to be the mutual information $I(XY;\pi)$ between the inputs (XY), and the messages sent and the public randomness in the protocol (π).

The information cost is always smaller than the communication complexity, and if the inputs to the parties are independent of each other (i.e. X is independent of Y), an optimal direct sum theorem can be proved for this measure of complexity. This means that from a protocol computing n copies of f with communication C, one can obtain a protocol computing f with information cost C/n, as long as the inputs to f are independent of each other. Thus the problem of proving direct sum theorems for independent inputs reduces to the problem of simulating a protocol τ with small information cost with a protocol ρ that has small communication. That is, the direct sum question reduces to the problem of protocol compression. Previous works carried out the compression by compressing every message of the protocol individually, hence the dependency on the number of rounds. Our stronger method of compression allows us to get new direct sum theorems that are independent of the number of rounds of communication.

1.1 Our Results

In our work we define a different measure of the information complexity of a communication protocol, that we call the *information content* of the protocol. The information content of the protocol is the information that the *parties* in the protocol learn by watching the messages and public randomness of the protocol, that they did not already know. Formally,

Definition 1.1. Given a distribution μ on inputs X, Y, and protocol π, denoting by $\pi(X,Y)$ the public randomness and messages exchanged during the protocol, we call the quantity

$$\mathsf{IC}_\mu(\pi) \stackrel{def}{=} I(X;\pi(X,Y)|Y) + I(Y;\pi(X,Y)|X)$$

the *information content* of π.

Since each party knows her own input, the protocol can only reveal less information to her than to an independent observer. Thus the information content is never larger than the information cost. It can be shown that the information content of a protocol is the same as the information cost, if the inputs are independent of each other. However, in the case that the inputs are dependent, the information content may be significantly smaller — for example, if μ is a distribution where $X = Y$ always, then the information content is always 0, though the mutual information between the messages and the inputs (i.e. the information cost) can be arbitrarily large. It is also easy to check that if π is deterministic, then the information content is simply the sum of the entropies $\mathsf{IC}_\mu(\pi) = H(\pi(X,Y)|Y) + H(\pi(X,Y)|X)$, which is the same as $H(\pi(X,Y))$ if X, Y are independent.

The notion of information content was used implicitly by Bar-Yossef et al [BYJKS04], and a direct sum theorem for this notion (using the techniques originating from Razborov [Raz92] and Raz [Raz95]) is implicit in their work. This direct sum theorem holds whether or not the inputs to the parties are independent of each other, unlike the analogous result for information cost. We can convert any protocol computing n copies of f with communication C into one that computes f with information content C/n and communication complexity C.

Our most important contributions are two new *protocol compression* methods that reduce the communication of protocols in terms of their information content. The first method works even for non-product distributions over the inputs and can simulate a protocol of information content I and communication complexity C using an expected number of $\tilde{O}(\sqrt{IC})$ communication bits. The second method works only for product distributions but can simulate any protocol of information content I with expected $\tilde{O}(I)$ communication. Note that in both cases the simulation cost is *independent* of the number of rounds. Indeed, these are the first compression schemes that do true *protocol* compression, as opposed to compressing each round at a time. The first result is also the first such compression scheme for non-product distribution over the inputs.

As a result, we obtain the first non-trivial direct sum theorem for randomized communication complexity. Loosely speaking, letting f^n be the function that outputs the concatenation of n invocations of f on independent inputs, and letting f^{+n} be the function that outputs the XOR of n such invocations, we show that **(a)** the randomized communication complexity of both f^n and f^{+n} is up to logarithmic factors $\sqrt{n}$ times the communication complexity of f, and **(b)** the distributional complexity of both f^n and f^{+n} over the distribution μ^n, where μ is a product distribution over individual input pairs, is n times the distributional complexity of f.[2]

1.1.1 Compressing Communication Protocols

We give two new *protocol compression* algorithms, that take a protocol π whose information content is small and transforms it into a protocol τ of small *communication complexity*.[3] Below we denote the communication complexity of a protocol τ by $\mathsf{CC}(\tau)$.

Theorem 1.2. *There is a universal constant c such that for every distribution μ, every protocol π, and every $\epsilon > 0$, there exists functions π_x, π_y, and a protocol τ such that $|\pi_x(X, \tau(X,Y)) - \pi(X,Y)| < \epsilon$, $\Pr[\pi_x(X, \tau(X,Y)) \neq \pi_y(Y, \tau(X,Y))] < \epsilon$ and*

$$\mathsf{CC}(\tau) \leq c\sqrt{\mathsf{CC}(\pi) \cdot \mathsf{IC}_\mu(\pi)} \frac{\log(\mathsf{CC}(\pi)/\epsilon)}{\epsilon} .$$

If the players want to obtain the results of running the protocol π, they can run τ instead and then use the functions π_x, π_y to reconstruct the effects of running π. The condition $|\pi_x(X, \tau(X,Y)) - \pi(X,Y)| < \epsilon$ ensures that the transcript of τ specifies a unique leaf in the protocol tree for π in such a way that this leaf is ϵ-close in statistical distance to the leaf sampled by π. The condition

[2]In both **(a)** and **(b)**, there is a loss of a constant additive factor in the actual statement of the result for f^{+n}. This accounts for the fact that if, say, f is the XOR function itself then clearly there is no direct sum theorem. See Remark 1.11.

[3] We note that this is in the communication complexity model, and hence these compression schemes are not necessarily computationally efficient. Even for singly message compression there are distributions with small entropy that cannot be efficiently compressed (e.g. pseudorandom distributions).

that $\Pr[\pi_x(X,\tau(X,Y)) \neq \pi_y(Y,\tau(X,Y))] < \epsilon$ guarantees that with high probability both players achieve a consensus on what the sampled leaf was. Thus, the triple τ, π_x, π_y specify a new protocol that is a compression of π.

In the case that the distribution μ over the inputs is a *product* distribution, $\mu = \mu_x \times \mu_y$, we get a stronger result that is tight up to polylogarithmic terms:

Theorem 1.3. *For every product distribution μ, every protocol π, and every $\epsilon > 0$, there exists functions π_x, π_y, and a protocol τ such that $|\pi_x(X,\tau(X,Y)) - \pi(X,Y)| < \epsilon$, $\Pr[\pi_x(X,\tau(X,Y)) \neq \pi_y(Y,\tau(X,Y))] < \epsilon$ and*

$$\mathsf{CC}(\tau) \leq \mathsf{IC}_\mu(\pi)\frac{\mathrm{polylog}(\mathsf{CC}(\pi)/\epsilon)}{\epsilon} .$$

Our results can be viewed as some kind of generalization of the traditional notion of string compression, a notion that applies only to the more restricted case of deterministic one way protocols. In the above theorems, our compressed protocols may use public randomness that can be large (though still bounded in terms of the communication complexity of the original protocol). However, we note that by the results of Newman [New91], any protocol that achieves some functionality can be converted into another protocol that achieves the same functionality and uses few public random bits. Thus our compression schemes are useful even when public randomness is expensive.

1.1.2 Direct sum theorems

Given a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, we define the function $f^n : \mathcal{X}^n \times \mathcal{Y}^n \to \mathcal{Z}^n$ to be the concatenation of the evaluations:

$$f^n(x_1,\ldots,x_n,y_1,\ldots,y_n) \overset{def}{=} (f(x_1,y_1), f(x_2,y_2),\ldots,f(x_n,y_n)).$$

Denote by $R_\rho(f)$ the communication complexity of the best randomized public coin protocol for computing f that errs with probability at most ρ. In this paper we show:

Theorem 1.4 (Direct Sum for Randomized Communication Complexity)**.** *For every $\alpha > 0$,*

$$\mathsf{R}_\rho(f^n) \cdot \log\left(\mathsf{R}_\rho(f^n)/\alpha\right) \geq \Omega\left(\mathsf{R}_{\rho+\alpha}(f)\alpha\sqrt{n}\right)$$

Theorem 1.4 is obtained using Yao's min-max principle from an analogous theorem for *distributional* communication complexity. For a distribution μ on the inputs $\mathcal{X} \times \mathcal{Y}$, we write $D^\mu_\rho(f)$ to denote the communication complexity of the best protocol (randomized or deterministic) that computes f with probability of error at most ρ when the inputs are sampled according to μ. We write μ^n to denote the distribution on n inputs, where each is sampled according to μ independently.

We first state the direct sum theorem for information content that is implicit in the work of [BYJKS04].

Theorem 1.5. *For every μ, f, ρ there exists a protocol τ computing f on inputs drawn from μ with probability of error at most ρ and communication at most $\mathsf{D}^{\mu^n}_\rho(f^n)$ such that $\mathsf{IC}_\mu(\tau) \leq \frac{2D^{\mu^n}_\rho(f^n)}{n}$.*

Compressing protocol τ above using Theorem 1.2 reduces the communication of this protocol to $\tilde{O}\left(\sqrt{\mathsf{IC}_\mu(\tau)\mathsf{D}^{\mu^n}_\rho(f^n)}\right) = \tilde{O}(\mathsf{D}^{\mu^n}_\rho(f^n)\sqrt{n})$. Formally, we prove:

Theorem 1.6 (Direct Sum for Distributional Communication Complexity)**.** *For every $\alpha > 0$,*

$$\mathsf{D}^{\mu^n}_{\rho}(f^n) \cdot \log\left(\mathsf{D}^{\mu^n}_{\rho}(f^n)/\alpha\right) \geq \Omega\left(\mathsf{D}^{\mu}_{\rho+\alpha}(f)\alpha\sqrt{n}\right)$$

The communication complexity bound of Theorem 1.6 only grows as the square root of the number of repetitions. However, in the case that the distribution on inputs is a product distribution, we use our stronger compression (Theorem 1.3) to obtain a direct sum theorem that is optimal up to logarithmic factor:

Theorem 1.7 (Direct Sum for Product Distributions)**.** *If μ is a product distribution, then for every $\alpha > 0$*

$$\mathsf{D}^{\mu^n}_{\rho}(f^n) \cdot \operatorname{polylog}\left(\mathsf{D}^{\mu^n}_{\rho}(f^n)/\alpha\right) \geq \Omega\left(\mathsf{D}^{\mu}_{\rho+\alpha}(f)\alpha n\right)$$

1.1.3 XOR Lemmas for communication complexity

When n is very large in terms of the other quantities, the above theorems can be superseded by trivial arguments, since f^n must require at least n bits of communication just to describe the output. Our next set of theorems show that almost the same bounds apply to the complexity of the XOR (or more generally sum modulo K) of n copies of f, where the trivial arguments do not hold. Assume that the output of the function f is in the group $\mathbb{Z}_K$ for some integer K, and define

$$f^{+n}(x_1, \ldots, x_n, y_1, \ldots, y_n) \overset{def}{=} \sum_{i=1}^{n} f(x_i, y_i).$$

We have the following results for the complexity of f^{+n}:

Theorem 1.8 (XOR Lemma for Randomized Communication Complexity)**.** *For every $\alpha > 0$,*

$$\mathsf{R}_{\rho}(f^{+n}) \cdot \log\left(\mathsf{R}_{\rho}(f^{+n})/\alpha\right) \geq \Omega\left((\mathsf{R}_{\rho+\alpha}(f) - 2\log K)\,\alpha\sqrt{n}\right)$$

Theorem 1.9 (XOR Lemma for Distributional Communication Complexity)**.** *For every $\alpha > 0$,*

$$\mathsf{D}^{\mu^n}_{\rho}(f^{+n}) \cdot \log\left(\mathsf{D}^{\mu^n}_{\rho}(f^{+n})/\alpha\right) \geq \Omega\left(\left(\mathsf{D}^{\mu}_{\rho+\alpha}(f) - 2\log K\right)\alpha\sqrt{n}\right)$$

Theorem 1.10 (XOR Lemma for Product Distributions)**.** *If μ is a product distribution, then for every $\alpha > 0$,*

$$\mathsf{D}^{\mu^n}_{\rho}(f^{+n}) \cdot \operatorname{polylog}\left(\mathsf{D}^{\mu^n}_{\rho}(f^{+n})/\alpha\right) \geq \Omega\left(\left(\mathsf{D}^{\mu}_{\rho+\alpha}(f) - 2\log K\right)\alpha n\right)$$

Remark 1.11. If $f : \mathbb{Z}_K \times \mathbb{Z}_K \to \mathbb{Z}_K$ is itself the sum function, then the communication complexity of f^{+n} does not grow at all, since there is a simple protocol to compute $\sum_i (x_i + y_i) = \sum_i x_i + \sum_j y_j$ using $2\log K$ bits. This suggests that some kind of additive loss (like the $2\log K$ term above) is necessary in the above theorems.

2 Our Techniques

We now give an informal overview of our compression algorithms. Our direct sum results are obtained in Section 4 by combining these with the direct sum for information content proven in Section 5. Full description of the compression algorithms are given in section 6 (for the general case) and 7 (for the product distribution case).

The goal of our compression algorithms is to take a protocol that uses large amounts of communication and conveys little information, and convert it into a protocol that makes better use of the communication to achieve better communication complexity. (Such algorithms need not be necessarily computationally efficient, see Footnote 3.)

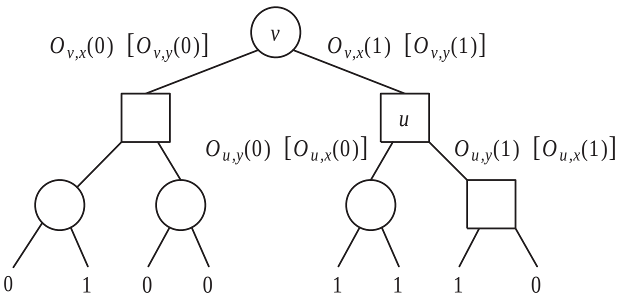
Note that generic message compression can be fit into this context by considering a deterministic one-way protocol, where player X needs to send a message to player Y. In this classical setting it is well known that protocol compression (i.e. simple data compression) can be achieved. In principle, one could try to apply round-by-round message compression to compress entire protocols. This approach suffers from the following fatal flaw: individual messages may (and are even likely) to contain $\ll 1$ bits of information. The communication cost of ≥ 1 bit per round would thus be $\gg$ information content of the round. Thus any attempt to implement the compression on a round-by-round basis, as opposed to an entire-protocol basis may work when the number of rounds is bounded, but is doomed to fail in general.

An instructive example on conveying a subconstant amount of information that we will use later in this exposition is the following. Suppose that player X gets n independent random bits $x_1, \dots, x_n$ and Y has no information about them. X then computes the majority $m = MAJ(x_1, \dots, x_n)$ and sends it to Y. With a perfectly random prior, the bit m is perfectly balanced, and thus in total X conveys one bit of information to Y. Suppose that in the protocol Y only really cared about the value of x_5. How much information did X convey about the input x_5? By symmetry and independence of the inputs, X conveys $1/n$ bits of information about x_5. After the bit m (suppose $m = 1$) is received by Y, her estimate of $P[x_5 = 1]$ changes from $1/2$ to $1/2 + \Theta(1/\sqrt{n})$. The fact that changing the probability from $1/2$ to $1/2 + \epsilon$ only costs ϵ^2 bits of information is the cause for the suboptimality of our general compression algorithm.

There are several challenges that need to be overcome to compress an arbitrary protocol. An interesting case to consider is a protocol where the players alternate sending each other messages, and each transmitted message is just a bit with information content $\epsilon \ll 1$. In this case, we cannot afford to even transmit one bit to simulate each of the messages, since that would incur an overhead of $1/\epsilon$, which would be too large for our application. This barrier was one of the big stumbling blocks for earlier works, which is why their results applied only when the number of rounds in the protocols was forced to be small.

We give two simulation protocols to solve this problem. The first solution works for all distributions, achieving sub-optimal parameters, while the second works only for product input distributions and achieves optimal parameters up to poly-logarithmic factors. In both solutions, the players simulate the original protocol π using shared randomness. The intuition is that if a message contains a small amount of information, then we do not need to communicate it, and can sample it using shared randomness instead.

It will be convenient to think of a protocol in terms of its protocol tree, after fixing the shared randomness (there may still be private randomness that is not fixed). This is a binary tree where every node v belongs to one of parties in the protocol, and specifies the probability of sending 1 or 0 as the next bit. We then define the tree of probabilities illustrated in Figure 1 as follows. For each node v_x of the protocol tree that is owned by the player X (i.e. it is his turn to speak), player X knows the "correct" probabilities $O_{v_x,x}(0)$ and $O_{v_x,x}(1)$ of the bit that she is about to send. Player Y does not know these probabilities, but she has estimates $O_{v_x,y}(0)$ and $O_{v_x,y}(1)$ for them based on her input Y (formally these estimates are simply the probability of seeing a 0 or 1 conditioned on the protocol reaching v_x and conditioned on y). In the case where the input distribution μ is



v
$O_{v,x}(0)$ $[O_{v,y}(0)]$
$O_{v,x}(1)$ $[O_{v,y}(1)]$
u
$O_{u,y}(0)$ $[O_{u,x}(0)]$
$O_{u,y}(1)$ $[O_{u,x}(1)]$
0 1 0 0 1 1 1 0

Figure 1: An illustration of the protocol tree for π. The round nodes are owned by X and the square nodes are owned by Y. On each edge the "correct" probability is indicated. The "approximate" probability that is estimated by the player who does not own the node is shown in the brackets.

a product distribution $\mu = \mu_x \times \mu_y$, the X player can also compute the estimates $O_{v_x,y}(0)$ and $O_{v_x,y}(1)$, since they are independent of the input y given the node v_x. The goal is to simulate the protocol according to the "correct" distributions.

2.1 Compression in the general case

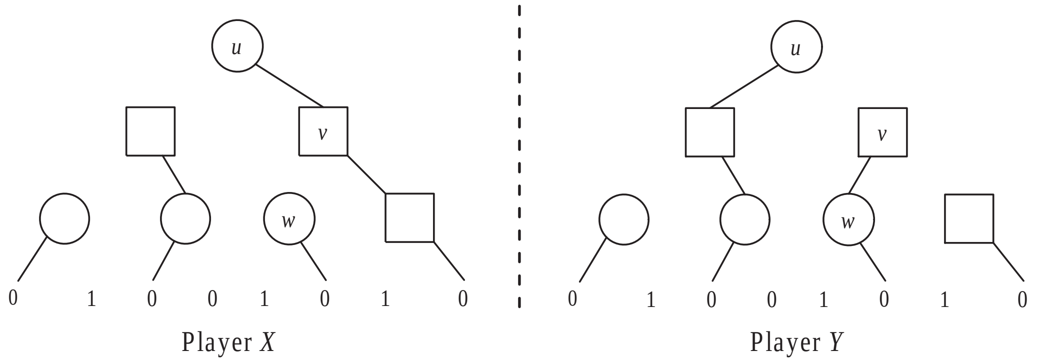


Figure 2: An illustration of the compression protocol for non-product distributions. The circle nodes are owned by player X and the square nodes are owned by Y. The figure illustrates the states of the protocol trees after all the bits have been sampled. The players then proceed to resolve their disagreements. The disagreement at node u is resolved in favor of X since he owns the node. The protocol proceeds to node v where the disagreement is resolved in favor of Y. The final computation path in this case is $u - v - w$, the output is 0, and the total number of disagreements along the path is 2.

In our first compression protocol, the players use shared randomness to sample the bit at every node of the protocol tree for $\pi(x, y)$. In other words, for every prefix v of messages, each player samples the next bit of the interaction according to the best guess that they have for how this bit is distributed, even if the next bit is actually transmitted by the other player in the original protocol. The players do this using shared randomness, in a way that guarantees that if their guesses are close to the correct distribution, then the probability that they sample the same bit is high. More precisely, the players share a random number $\kappa_v \in [0,1]$ for every node v in the tree, and each player guesses the next bit following v to be 1, if the player's estimated probability for the message being 1 is at least κ_v. Note that the player that owns v samples the next bit with the correct probability. It's not hard to see that the probability of getting inconsistent samples at the node v is at most $|O_{v,x} - O_{v,y}| \overset{def}{=} |O_{v,x}(0) - O_{v,y}(0)| + |O_{v,x}(1) - O_{v,y}(1)|$. Once they have each sampled from the possible interactions, we shall argue that there is a *correct leaf* in the protocol tree, whose distribution is exactly the same as the leaf in the original protocol. This is the leaf that is obtained by starting at the root and repeatedly taking the edge that was sampled by the owner of the node. We then show how the players can use hashing and binary search to communicate a polylogarithmic number of bits with each other to resolve the inconsistencies in their samples and find this correct path with high probability. In this way, the final outcome will be statistically close to the distribution of the original protocol. An example run for this protocol is illustrated on Figure 2. The additional interaction cost scales according to the expected number of inconsistencies on the path to the correct leaf, which we show can be bounded by $\sqrt{I \cdot C}$, where I is the information content and C is the communication cost of the original protocol.

Recall from the Majority example above that ϵ information can mean that $|O_{v,x} - O_{v,y}| \approx \sqrt{\epsilon}$. In fact, the "worst case" example for us is when in each round I/C information is conveyed, leading to a per-round error of $\sqrt{I/C}$ and a total expected number of mistakes of $\sqrt{I/C} \cdot C = \sqrt{I \cdot C}$.

2.2 Compression when the inputs are independent

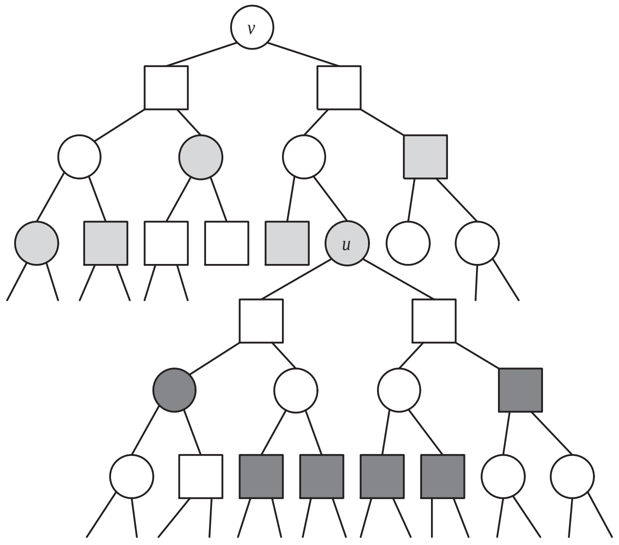


Figure 3: An illustration of the compression protocol for product distributions. The gray layer represents the "frontier" of nodes where some fixed amount of information is conveyed in the original protocol, and which is simulated in one iteration of the compressed protocol. Once the players agreeon a node u, they compute a new frontier, illustrated here by the black layer.

Our more efficient solution, which gives a protocol with communication complexity within polylogarithmic factors of the information content, only applies when the input distribution $\mu = \mu_x \times \mu_y$ is a product distribution. It is illustrated on Figure 3. The idea in this case is not to simulate the protocol round-per-round at all. Rather, we simulate chunks of the protocol that convey a constant amount of information each. If we can simulate a portion of the protocol that conveys a constant (or even 1/poly-log) amount of information using poly-logarithmic number of bits of communication, then we can simulate the entire protocol using the optimal $\tilde{O}(I)$ bits of communication.

The advantage the players have in the product case is that for each node in the tree, the player who owns that node knows not only the correct distribution for the next bit, but also knows what the distribution that the other party has in mind is. They can use this shared knowledge to sample entire paths according to the distribution that is common knowledge at every step. In general, the distribution of the sampled path can deviate quite a bit from the correct distribution. However, we argue that if the information conveyed on a path is small (1/ polylog bit), then the difference between the correct and the approximate probability is constant. After sampling the approximate bits for the appropriate number of steps so as to cover 1/ polylog information, the players can communicate to estimate the correct probability with which this node was supposed to occur. The players can then either accept the sequence or resample a new sequence in order to get a final sample that behaves in a way that is close to the distribution of the original protocol.

There are several technical challenges involved in getting this to work. The fact that the inputs of the players are independent is important for the players to decide how many messages the players should try to sample at once to get to the frontier where 1/ polylog bits of information have been revealed. When the players' inputs are dependent, they cannot estimate how many messages they should sample before the information content becomes too high, and we are unable to make this approach work.

3 Preliminaries

Notation. We reserve capital letters for random variables and distributions, calligraphic letters for sets, and small letters for elements of sets. Throughout this paper, we often use the notation $|b$ to denote conditioning on the event $B = b$. Thus $A|b$ is shorthand for $A|B = b$. Given a sequence of symbols $A = A_1, A_2, \ldots, A_k$, we use $A_{\leq j}$ denote the prefix of length j.

We use the standard notion of *statistical/total variation* distance between two distributions.

Definition 3.1. Let D and F be two random variables taking values in a set $\mathcal{S}$. Their *statistical distance* is

$$|D - F| \overset{def}{=} \max_{\mathcal{T} \subseteq \mathcal{S}} (|\Pr[D \in \mathcal{T}] - \Pr[F \in \mathcal{T}]|) = \frac{1}{2} \sum_{s \in \mathcal{S}} |\Pr[D = s] - \Pr[F = s]|$$

If $|D - F| \leq \epsilon$ we shall say that D is ϵ-*close* to F. We shall also use the notation $D \overset{\epsilon}{\approx} F$ to mean D is ϵ-close to F.

3.1 Information Theory

Definition 3.2 (Entropy)**.** The *entropy* of a random variable X is $H(X) \overset{def}{=} \sum_x \Pr[X = x] \log(1/\Pr[X = x])$. The *conditional entropy* $H(X|Y)$ is defined to be $\mathbb{E}_{y \in_{\mathrm{R}} Y} [H(X|Y = y)]$.

Fact 3.3. *$H(AB) = H(A) + H(B|A)$.*

Definition 3.4 (Mutual Information)**.** The *mutual information* between two random variables A, B, denoted $I(A;B)$ is defined to be the quantity $H(A) - H(A|B) = H(B) - H(B|A)$. The *conditional mutual information* $I(A;B|C)$ is $H(A|C) - H(A|BC)$.

In analogy with the fact that $H(AB) = H(A) + H(B|A)$,

Proposition 3.5. *Let C_1, C_2, D, B be random variables. Then*

$$I(C_1C_2;B|D) = I(C_1;B|D) + I(C_2;B|C_1D).$$

The previous proposition immediately implies the following:

Proposition 3.6 (Super-Additivity of Mutual Information)**.** *Let C_1, C_2, D, B be random variables such that for every fixing of D, C_1 and C_2 are independent. Then*

$$I(C_1;B|D) + I(C_2;B|D) \quad \leq \quad I(C_1C_2;B|D).$$

We also use the notion of *divergence*, which is a different way to measure the distance between two distributions:

Definition 3.7 (Divergence)**.** The informational divergence between two distributions is $\mathbb{D}\left(A||B\right) \stackrel{def}{=} \sum_x A(x)\log(A(x)/B(x))$.

For example, if B is the uniform distribution on $\{0,1\}^n$ then $\mathbb{D}\left(A||B\right) = n - H(A)$.

Proposition 3.8. $\mathbb{D}\left(A||B\right) \geq |A - B|^2$.

Proposition 3.9. *Let A, B, C be random variables in the same probability space. For every a in the support of A and c in the support of C, let B_a denote $B|A = a$ and B_{ac} denote $B|A = a, C = c$. Then $I(A;B|C) = \mathbb{E}_{a,c \in_R A,C}\left[\mathbb{D}\left(B_{ac}||B_c\right)\right]$*

The above facts imply the following easy proposition:

Proposition 3.10. *With notation as in Proposition 3.9, for any random variables A, B,*

$$\mathop{\mathbb{E}}_{a \in_R A}\left[|(B_a) - B|\right] \leq \sqrt{I(A;B)}.$$

Proof.

$$\begin{aligned}
\mathop{\mathbb{E}}_{a \in_R A}\left[|(B_a) - B|\right] &\leq \mathop{\mathbb{E}}_{a \in_R A}\left[\sqrt{\mathbb{D}\left(B_a||B\right)}\right] \\
&\leq \sqrt{\mathop{\mathbb{E}}_{a \in_R A}\left[\mathbb{D}\left(B_a||B\right)\right]} && \text{by convexity} \\
&= \sqrt{I(A;B)} && \text{by Proposition 3.9}
\end{aligned}$$

□

3.2 Communication Complexity

Let $\mathcal{X}, \mathcal{Y}$ denote the set of possible inputs to the two players, who we name P_x, P_y. In this paper[4], we view a *private coins protocol* for computing a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{Z}_K$ as a binary tree with the following structure:

- Each node is *owned* by P_x or by P_y
- For every $x \in \mathcal{X}$, each internal node v owned by P_x is associated with a distribution $O_{v,x}$ supported on the children of v. Similarly, for every $y \in \mathcal{Y}$, each internal node v owned by P_y is associated with a distribution $O_{v,y}$ supported on the children of v.
- The leaves of the protocol are labeled by output values from $\mathbb{Z}_K$.

On input x, y, the protocol π is executed as in Figure 4.

Generic Communication Protocol

1. Set v to be the root of the protocol tree.
2. If v is a leaf, the protocol ends and outputs the value in the label of v. Otherwise, the player owning v samples a child of v according to the distribution associated with her input for v and sends a bit to the other player to indicate which child was sampled.
3. Set v to be the newly sampled node and return to the previous step.

Figure 4: A communication protocol.

A public coin protocol is a distribution on private coins protocols, run by first using shared randomness to sample an index r and then running the corresponding private coin protocol π_r. Every private coin protocol is thus a public coin protocol. The protocol is called deterministic if all distributions labeling the nodes have support size 1.

Definition 3.11. The *communication complexity* of a public coin protocol π, denoted $\mathsf{CC}(\pi)$, is the maximum depth of the protocol trees in the support of π.

Given a protocol π, $\pi(x, y)$ denotes the concatenation of the public randomness with all the messages that are sent during the execution of π. We call this the *transcript* of the protocol. We shall use the notation $\pi(x, y)_j$ to refer to the j'th transmitted bit in the protocol. We write $\pi(x, y)_{\leq j}$ to denote the concatenation of the public randomness in the protocol with the first j message bits that were transmitted in the protocol. Given a transcript, or a prefix of the transcript, v, we write $\mathsf{CC}(v)$ to denote the number of message bits in v (i.e. the length of the communication).

We often assume that every leaf in the protocol is at the same depth. We can do this since if some leaf is at depth less than the maximum, we can modify the protocol by adding dummy nodes which are always picked with probability 1, until all leaves are at the same depth. This does not change the communication complexity.

[4]The definitions we present here are equivalent to the classical definitions and are more convenient for our proofs.

Definition 3.12 (Communication Complexity notation)**.** For a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{Z}_K$, a distribution μ supported on $\mathcal{X} \times \mathcal{Y}$, and a parameter $\rho > 0$, $D^{\mu}_{\rho}(f)$ denotes the communication complexity of the cheapest deterministic protocol for computing f on inputs sampled according to μ with error ρ. $R_\rho(f)$ denotes the cost of the best randomized public coin protocol for computing f with error at most ρ on *every* input.

We shall use the following simple fact, first observed by Yao:

Fact 3.13 (Yao's Min-Max)**.** $R_\rho(f) = \max_\mu D^{\mu}_{\rho}(f)$.

Recall that the information content $\mathsf{IC}_\mu(\pi)$ of a protocol π is defined to be $I(\pi(X,Y);X|Y) + I(\pi(X,Y);Y|X)$.

Remark 3.14 (Information content of private vs. public coins protocols.)**.** Another way to view the difference between public coins and private coins protocols is that the public randomness is considered part of the protocol's transcript. But even if the randomness is short compared to the overall communication complexity, making it public can have a dramatic effect on the information content of the protocol. (As an example, consider a protocol where one party sends a message of $x \oplus r$ where x is its input and r is random. If the randomness r is private then this message has zero information content. If the randomness is public then the message completely reveals the input. This protocol may seem trivial since its communication complexity is larger than the input length, but in fact we will be dealing with exactly such protocols, as our goal will be to "compress" communication of protocols that have very large communication complexity, but very small information content.)

3.3 Finding differences in inputs

We use the following lemma of Feige et al. [FPRU94]:

Lemma 3.15 ([FPRU94])**.** *There is a randomized public coin protocol τ with communication complexity $O(\log(k/\epsilon))$ such that on input two k-bit strings x, y, it outputs the first index $i \in [k]$ such that $x_i \neq y_i$ with probability at least $1 - \epsilon$, if such an i exists.*

For completeness, we include the proof (based on hashing) in Appendix C.

4 Proof of the direct sum theorem

In this section, we prove Theorem 1.4, showing a direct sum for distributional communication complexity even in the case where the input distribution is not necessarily a product distribution. By Yao's minimax principle, for every function f, $R_\rho(f) = \max_\mu D^{\mu}_{\rho}(f)$. Thus Theorem 1.6 implies Theorem 1.4 and Theorem 1.9 implies Theorem 1.8. So we shall focus on proving Theorem 1.6 and its XOR Lemma analaog Theorem 1.9.

By Theorem 1.5, the main step to establish Theorem 1.6 is to give an efficient simulation of a protocol with small information content by a protocol with small communication complexity. We shall thus prove

Theorem 1.2 (Restated)**.** *There is a universal constant c such that for every distribution μ, every protocol π, and every $\epsilon > 0$, there exists functions π_x, π_y, and a protocol τ such that $|\pi_x(X, \tau(X,Y)) - \pi(X,Y)| < \epsilon$, $\Pr[\pi_x(X, \tau(X,Y)) \neq \pi_y(Y, \tau(X,Y))] < \epsilon$ and*

$$\mathsf{CC}(\tau) \leq c\sqrt{\mathsf{CC}(\pi) \cdot \mathsf{IC}_\mu(\pi)} \frac{\log(\mathsf{CC}(\pi)/\epsilon)}{\epsilon} .$$

Proof of direct sum theorem from Theorem 1.2. Before proving Theorem 1.2, let's see how we can use it to get our main result (Theorem 1.6). Let π be any protocol computing f^n on inputs drawn from μ^n with probability of error less than ρ. Then by Theorem 1.5, there exists a protocol τ_1 computing f on inputs drawn from μ with error at most ρ with $\mathsf{CC}(\tau_1) \leq \mathsf{CC}(\pi)$ and $\mathsf{IC}_\mu(\tau_1) \leq 2\mathsf{CC}(\pi)/n$. Next, applying Theorem 1.2 to the protocol τ_1 gives that there must exist a protocol τ_2 computing f on inputs drawn from μ with error at most $\rho + \alpha$ and

$$\begin{aligned}
\mathsf{CC}(\tau_2) &\leq O\left(\sqrt{\mathsf{CC}(\tau_1)\mathsf{IC}_\mu(\tau_1)}\log(\mathsf{CC}(\tau_1)/\alpha)/\alpha\right) \\
&= O\left(\sqrt{\mathsf{CC}(\pi)\mathsf{CC}(\pi)/n}\log(\mathsf{CC}(\pi)/\alpha)/\alpha\right) \\
&= O\left(\frac{\mathsf{CC}(\pi)\log(\mathsf{CC}(\pi)/\alpha)/\alpha}{\sqrt{n}}\right)
\end{aligned}$$

This proves Theorem 1.6. □

Proof of the XOR Lemma. The proof for Theorem 1.9 (XOR Lemma for distributional complexity) is very similar. First, we show an XOR-analog of Theorem 1.5:

Theorem 4.1. *For every distribution μ, there exists a protocol τ computing f with probability of error ρ over the distribution μ with $\mathsf{CC}(\tau) \leq \mathsf{D}_\rho^{\mu^n}(f^{+n}) + 2\log K$ such that if τ' is the protocol that is the same as τ but stops running after $\mathsf{D}_\rho^{\mu^n}(f^{+n})$ message bits have been sent, then $\mathsf{IC}_\mu(\tau') \leq \frac{2D_\rho^{\mu^n}(f^{n+})}{n}$.*

Now let π be any protocol computing f^{+n} on inputs drawn from μ^n with probability of error less than ρ. Then by Theorem 4.1, there exists a protocol τ_1 computing f on inputs drawn from μ with error at most ρ with $\mathsf{CC}(\tau_1) \leq \mathsf{CC}(\pi) + 2\log K$ and such that if τ_1' denotes the first $\mathsf{CC}(\pi)$ bits of the message part of the transcript, $\mathsf{IC}_\mu(\tau_1') \leq 2\mathsf{CC}(\pi)/n$. Next, applying Theorem 1.2 to the protocol τ_1' gives that there must exist a protocol τ_2' simulating τ_1' on inputs drawn from μ with error at most $\rho + \alpha$ and

$$\begin{aligned}
\mathsf{CC}(\tau_2') &\leq O\left(\sqrt{\mathsf{CC}(\tau_1')\mathsf{IC}_\mu(\tau_1')}\log(\mathsf{CC}(\tau_1')/\alpha)/\alpha\right) \\
&= O\left(\sqrt{\mathsf{CC}(\pi)\mathsf{CC}(\pi)/n}\log(\mathsf{CC}(\pi)/\alpha)/\alpha\right) \\
&= O\left(\frac{\mathsf{CC}(\pi)\log(\mathsf{CC}(\pi)/\alpha)/\alpha}{\sqrt{n}}\right)
\end{aligned}$$

Finally we get a protocol for computing f by first running τ_2' and then running the last $2\log K$ bits of π. Thus we must have that $O\left(\frac{\mathsf{CC}(\pi)\log(\mathsf{CC}(\pi)/\alpha)/\alpha}{\sqrt{n}}\right) + 2\log K \leq D_{\rho+\alpha}^\mu(f)$, as in the theorem. □

5 Reduction to Small Information Content

We now prove Theorems 1.5 and 4.1, showing that the existence of a protocol with communication complexity C for f^n (or f^{+n}) implies a protocol for f with information content roughly C/n.

Theorem 1.5 (Restated)**.** *For every μ, f, ρ there exists a protocol τ computing f on inputs drawn from μ with probability of error at most ρ and communication at most $\mathsf{D}_\rho^{\mu^n}(f^n)$ such that $\mathsf{IC}_\mu(\tau) \leq \frac{2D_\rho^{\mu^n}(f^n)}{n}$.*

Theorem 4.1 (Restated)**.** *For every distribution μ, there exists a protocol τ computing f with probability of error ρ over the distribution μ with $\mathsf{CC}(\tau) \leq \mathsf{D}_\rho^{\mu^n}(f^{+n}) + 2\log K$ such that if τ' is the protocol that is the same as τ but stops running after $\mathsf{D}_\rho^{\mu^n}(f^{+n})$ message bits have been sent, then $\mathsf{IC}_\mu(\tau') \leq \frac{2D_\rho^{\mu^n}(f^{n+})}{n}$.*

The key idea involved in proving the above theorems is a way to split dependencies between the inputs that arose in the study of lowerbounds for the communication complexity of disjointness and in the study of parallel repetition [KS92, Raz92, Raz95].

Proof. Fix μ, f, n, ρ as in the statement of the theorems. We shall prove Theorem 1.5 first. Theorem 4.1 will easily follow by the nature of our proof. To prove Theorem 1.5, we show how to use the best protocol for computing f^n to get a protocol with small information content computing f. Let π be a deterministic protocol with communication complexity $\mathsf{D}_\rho^{\mu^n}(f^n)$ computing f^n with probability of error at most ρ.

Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ denote random variables distributed according to μ^n. Let $\pi(X^n, Y^n)$ denote the random variable of the transcript (which is just the concatenation of all messages, since this is a deterministic protocol) that is obtained by running the protocol π on inputs $(X_1, Y_1), \ldots,$ (X_n, Y_n). We define random variables $W = W_1, \ldots, W_n$ where each W_j takes value in the disjoint union $\mathcal{X} \uplus \mathcal{Y}$ so that each $W_j = X_j$ with probability $1/2$ and $W_j = Y_j$ with probability $1/2$. Let W^{-j} denote $W_1, \ldots, W_{j-1}, W_{j+1}, \ldots, W_n$.

Our new protocol τ shall operate as in Figure 5. Note the distinction between *public* and *private* randomness. This distinction make a crucial difference in the definition of information content, as making more of the randomness public reduces the information content of a protocol.

The probability that the protocol τ makes an error on inputs sampled from μ is at most the probability that the protocol π makes an error on inputs sampled from μ^n. It is also immediate that $\mathsf{CC}(\tau) = \mathsf{CC}(\pi)$. All that remains is to bound the information content $\mathsf{IC}_\mu(\tau)$. We do this by relating it to the communication complexity of π.

To simplify notation, below we will use π to denote $\pi(X, Y)$ when convenient.

$$\mathsf{D}_\rho^{\mu^n}(f^n) \geq \mathsf{CC}(\pi) \geq I(X_1 \cdots X_n Y_1 \cdots Y_n; \pi | W) \geq \sum_{j=1}^{n} I(X_j Y_j; \pi | W) = n I(X_J Y_J; \pi | W J),$$

where the last inequality follows from Proposition 3.6. Next observe that the variables JW^{-J} are

Protocol τ

Public Randomness Phase :

1. The players sample $j, w^{-j} \in_{\mathrm{R}} J, W^{-J}$ using public randomness.

Private Randomness Phase :

1. P_x sets $x_j = x$, P_y sets $y_j = y$.
2. For every $i \neq j$, P_x samples X_i conditioned on the value of w^{-j}.
3. For every $i \neq j$, P_y samples Y_i conditioned on the value of w^{-j}.
4. The players simulate π on the inputs $x_1, \ldots, x_n, y_1, \ldots, y_n$ and output the j'th output of π.

Figure 5: A protocol simulating π

independent of X_J, Y_J, W_J. Thus we can write

$$\begin{aligned} I(X_J Y_J; \pi | JW) &= I(X_J Y_J; \pi | J W_J W^{-J}) + I(X_J Y_J; J W^{-J} | W_J) \\ &= I(X_J Y_J; J W^{-J} \pi | W_J) \\ &= I(XY; J W^{-J} \pi | W_J) \\ &= \frac{I(XY; J W^{-J} \pi | X_J) + I(XY; J W^{-J} \pi | Y_J)}{2} \\ &= \frac{I(Y; J W^{-J} \pi | X_J) + I(X; J W^{-J} \pi | Y_J)}{2}, \end{aligned}$$

where the last equality follows from the fact that X_J determines X and Y_J determines Y. This last quantity is simply the information content of τ. Thus we have shown that $\mathsf{CC}(\pi) \geq (n/2)\mathsf{IC}_\mu(\tau)$ as required.

Remark 5.1. The analysis above can be easily improved to get the bound $\mathsf{IC}_\mu(\tau) \leq \mathsf{CC}(\tau)/n$ by taking advantage of the fact that each bit of the transcript gives information about at most one of the players' inputs, but for simplicity we do not prove this here.

This completes the proof for Theorem 1.5. The proof for Theorem 4.1 is very similar. As above, we let π denote the best protocol for computing f^{+n} on inputs sampled according to μ^n. Analogous to τ as above, we define the simulation γ as in Figure 6.

As before, the probability that the protocol γ makes an error on inputs sampled from μ is at most the probability that the protocol π makes an error on inputs sampled from μ^n, since there is an error in γ if and only if there is an error in the computation of z. It is also immediate that $\mathsf{CC}(\gamma) = \mathsf{CC}(\pi) + 2\log K$.

Let $\gamma'(X, Y)$ denote the concatenation of the public randomness and the messages of γ upto the computation of z. Then, exactly as in the previous case, we have the bound:

$$\mathsf{IC}_\mu(\gamma') \leq 2\mathsf{CC}(\gamma)/n$$

Protocol γ

Public Randomness Phase :

1. The players sample $j, w^{-j} \in_{\mathrm{R}} J, W^{-J}$ using public randomness.

Private Randomness Phase :

1. P_x sets $x_j = x$, P_y sets $y_j = y$.
2. For every $i \neq j$, P_x samples X_i conditioned on the value of w^{-j}.
3. For every $i \neq j$, P_y samples Y_i conditioned on the value of w^{-j}.
4. The players simulate π on the inputs $x_1, \ldots, x_n, y_1, \ldots, y_n$ to compute $z \in \mathbb{Z}_K$.
5. P_x computes $\sum_{i \neq j, w_i = y_i} f(x_i, w_i)$ and sends this sum to P_y
6. P_y outputs the value of the function as $z - \sum_{i \neq j, w_i = y_i} f(x_i, w_i) - \sum_{i \neq j, w_i = x_i} f(w_i, y_i)$.

Figure 6: A protocol simulating π

This completes the proof.

□

6 Protocol compression: the non product case

We now prove our main technical theorem, Theorem 1.2:

Theorem 1.2 (Restated)**.** *There is a universal constant c such that for every distribution μ, every protocol π, and every $\epsilon > 0$, there exists functions π_x, π_y, and a protocol τ such that $|\pi_x(X, \tau(X,Y)) - \pi(X,Y)| < \epsilon$, $\Pr[\pi_x(X, \tau(X,Y)) \neq \pi_y(Y, \tau(X,Y))] < \epsilon$ and*

$$\mathsf{CC}(\tau) \leq c\sqrt{\mathsf{CC}(\pi) \cdot \mathsf{IC}_\mu(\pi)} \frac{\log(\mathsf{CC}(\pi)/\epsilon)}{\epsilon} .$$

6.1 A proof sketch

Here is a high level sketch of the proof. Let μ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Let π be a public coin protocol that does some computation using the inputs X, Y drawn according to μ. Our goal is to give a protocol τ that simulates π on μ such that[5]

$$\mathsf{CC}(\tau) = O\left(\sqrt{\mathsf{CC}(\pi) \cdot \mathsf{IC}_\mu(\pi)} \log(\mathsf{CC}(\pi)\right) .$$

For the sake of simplicity, here we assume that the protocol π has no public randomness. π then specifies a protocol tree which is a binary tree of depth $\mathsf{CC}(\pi)$ where every non-leaf node w is owned by one of the players, whose turn is to speak in this node. Each non leaf node has a "0 child" and

[5]We identify the communication complexity of the protocols π, τ with their expected communication under μ, as by adding a small error, the two can be related using an easy Markov argument.

a "1 child". For every such node w in the tree and every possible message $b \in \{0,1\}$, the X player gets input x and uses this to define $O_{w,x}(b)$ as the probability in π that conditioned on reaching the node w and the input being x, the next bit will be b. The Y player defines $O_{w,y}(b)$ analogously. Note that if w is owned by the X player, then $O_{w,x}(b)$ is exactly the correct probability with which b is transmitted in the real protocol.

For every such node w, the players use public randomness to sample a shared random number $\kappa_w \in [0,1]$ for every non-leaf node w in the tree. The X player uses these numbers to define the child $C_x(w)$ for every node w as follows: if $O_{w,x}(1) < \kappa_w$, $C_x(w)$ is set to the 0 child of w, and is set to the 1 child otherwise. The Y Player does the same using the values $O_{w,y}(1)$ (but the same κ_w) instead.

Now let $v_0, \ldots, v_{\mathsf{CC}(\pi)}$ be the correct path in the tree. This is the path where every subsequent node was sampled by the player that owned the previous node: for every i,

$$v_{i+1} = \begin{cases} C_x(v_i) & \text{if X player owns } v_i \\ C_y(v_i) & \text{if Y player owns } v_i \end{cases}$$

$v_{\mathsf{CC}(\pi)}$ has the same distribution as a leaf in π was supposed to have, and the goal of the players will be to identify $v_{\mathsf{CC}(\pi)}$ with small communication.

In order to do this, the X player will compute the sequence of nodes $v_0^x, \ldots, v_{\mathsf{CC}(\pi)}^x$ by setting $v_{i+1}^x = C_x(v_i^x)$. Similarly, the Y player computes the path $v_0^y, \ldots, v_{\mathsf{CC}(\pi)}^y$ by setting $v_{i+1}^y = C_y(v_i^y)$. Observe that if these two paths agree on the first k nodes, then they must be equal to the correct path upto the first k nodes.

So far, we have not communicated at all. Now the parties communicate to find the first index i for which $v_i^x \neq v_i^y$. If $v_{i-1} = v_{i-1}^x = v_{i-1}^y$ was owned by the X player, the parties reset the i'th node in their paths to v_i^x. Similarly, if v_{i-1} was owned by the Y player, the parties reset their i'th node to be v_i^y. In this way, they keep fixing their paths until they have computed the correct path.

Thus the communication complexity of the new protocol is bounded by the number of mistakes times the communication complexity of finding a single mistake. Every path in the tree is specified by a $\mathsf{CC}(\pi)$-bit string, and finding the first inconsistency reduces to the problem of finding the first difference in two $\mathsf{CC}(\pi)$-bit strings. A simple protocol of Feige et al [FPRU94] (based on hashing and binary search) gives protocol for finding this first inconsistency, with communication only $O(\log \mathsf{CC}(\pi))$. We describe an analyze this protocol in Appendix C. In Appendix 6 we show how to bound the expected number of mistakes on the correct path in terms of the information content of the protocol. We show that if we are node v_i in the protocol and the next bit has ϵ information, then the probability that $\Pr[C_x(v_i) \neq C_y(v_i)] \leq \sqrt{\epsilon}$. Since the total information content is $\mathsf{IC}_\mu(\pi)$, we can use the Cauchy-Schwartz inequality to bound the expected number of mistakes by $\sqrt{\mathsf{CC}(\pi)\mathsf{IC}_\mu(\pi)}$.

6.2 The actual proof

In order to prove Theorem 1.2, we consider the protocol tree $\mathcal{T}$ for π_r, for every fixing of the public randomness r. If R is the random variable for the public randomness used in π, we have that

Claim 6.1. $\mathsf{IC}_\mu(\pi) = \mathbb{E}_R\left[\mathsf{IC}_\mu(\pi_R)\right]$

Proof.

$$\begin{aligned}\mathsf{IC}_\mu(\pi) &= I(\pi(X,Y);X|Y) + I(\pi(X,Y);Y|X)\\ &= I(R\pi_R(X,Y);X|Y) + I(R\pi_R(X,Y);Y|X)\\ &= I(R;X|Y) + I(R;Y|X) + I(\pi_R(X,Y);X|YR) + I(\pi_R(X,Y);Y|XR)\\ &= I(\pi_R(X,Y);X|YR) + I(\pi_R(X,Y);Y|XR)\\ &= \mathop{\mathbb{E}}_R[\mathsf{IC}_\mu(\pi_R)]\end{aligned}$$

□

It will be convenient to describe protocol π_r in a non-standard, yet equivalent way in Figure 7.

Protocol π_r

Sampling Phase :

1. For every non-leaf node w in the tree, the player who owns w samples a child according to the distribution given by her input and the public randomness r. This leaves each player with a subtree of the original protocol tree, where each node has out-degree 1 or 0 depending on whether or not it is owned by the player.

Path Finding Phase :

1. Set v to be the root of the tree.
2. If v is a leaf, the computation ends with the value of the node. Else, the player to whom v belongs communicates one bit to the other player to indicate which of the children was sampled.
3. Set v to the sampled child and return to the previous step.

Figure 7: π restated

For some error parameters β, γ, we define a randomized protocol $\tau_{\beta,\gamma}$ that will simulate π and use the same protocol tree. The idea behind the simulation is to avoid communicating by guessing what the other player's samples look like. The players shall make many mistakes in doing this, but they shall then use Lemma 3.15 to correct the mistakes and end up with the correct transcript. Our simulation is described in Figure 8.

Define $\pi_x(x, \tau_{\beta,\gamma}(x,y))$ (resp. $\pi_y(y, \tau_{\beta,\gamma}(x,y))$) to be leaf of the final path computed by P_x (resp. P_y) in the protocol $\tau_{\beta,\gamma}$ (see Figure 8). The definition of the protocol $\tau_{\beta,\gamma}$ implies immediately the following upper bound on its communication complexity

$$\mathsf{CC}(\tau_{\beta,\gamma}) = O(\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi)}\log(\mathsf{CC}(\pi)/\beta)/\gamma)\ . \tag{1}$$

Let $V = V_0, \ldots, V_{\mathsf{CC}(\pi)}$ denote the "right path" in the protocol tree of $\tau_{\beta,\gamma}$. That is, every i, $V_{i+1} = 0$ if the left child of $V_{\le i}$ is sampled by the owner of $V_{\le i}$ and $V_{i+1} = 1$ otherwise. Observe that

Protocol $\tau_{\beta,\gamma}$

Public Sampling Phase :

1. Sample r according to the distribution of the public randomness in π.

Correlated Sampling Phase :

1. For *every* non-leaf node w in the tree, let κ_w be a uniformly random element of $[0,1]$ sampled using public randomness.
2. On input x, y, player P_x (resp. P_y) defines the tree $\mathcal{T}_x$ (resp. $\mathcal{T}_y$) in the following way: for each node w, P_x (resp. P_y) includes the edge to the left child if $\Pr[\pi_r(X,Y)$ reaches the left child$|\pi_r(X,Y)$ reaches w and $X = x] > \kappa_w$ (resp. if $\Pr[\pi_r(X,Y)$ reaches the left child$|\pi_r(X,Y)$ reaches w and $Y = y] > \kappa_w$). Otherwise, the right child is picked.

Path Finding Phase :

1. Each of the players computes the unique path in their trees that leads from the root to a leaf. The players then use Lemma 3.15, communicating $O(\log(n/\beta))$ bits to find the first node at which their respective paths differ, if such a node exists. The player that does not own this node corrects this edge and recomputes his path. They repeatedly correct their paths in this way $\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi)}/\gamma$ times.

Figure 8: The simulation of π

this path has the right distribution, since every child is sampled with exactly the right conditional probability by the corresponding owner. That is, we have the following claim:

Claim 6.2. *For every x, y, r, the distribution of $V|xyr$ as defined above is the same as the distribution of the sampled transcript in the protocol π.*

This implies in particular, that

$$I(X;V|rY) + I(Y;V|rX) = \mathsf{IC}_\mu(\pi_r)\ .$$

Given two fixed trees $\mathcal{T}_x, \mathcal{T}_y$ as in the above protocol, we say there is a *mistake* at level i if the out-edges of V_{i-1} are inconsistent in the trees. We shall first show that the expected number of mistakes that the players make is small.

Lemma 6.3. $\mathbb{E}\,[\#$ *of mistakes in simulating* $\pi_r|r] \le \sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi_r)}$.

Proof. For $i = 1,\ldots,\mathsf{CC}(\pi)$, we denote by C_{ir} the indicator random variable for whether or not a mistake occurs at level i in the protocol tree for π_r, so that the number of mistakes is $\sum_{i=1}^{\mathsf{CC}(\pi)} C_{ir}$.

We shall bound $\mathbb{E}\,[C_{ir}]$ for each i. A mistake occurs at a vertex w at depth i exactly when $\Pr[V_{i+1} = 0|x \wedge V_{\le i} = w] \le \kappa_w < \Pr[V_{i+1} = 0|y \wedge V_{\le i} = w]$ or $\Pr[V_{i+1} = 0|y \wedge V_{\le i} = w] \le \kappa_w < \Pr[V_{i+1} = 0|x \wedge V_{\le i} = w]$. Thus a mistake occurs at $v_{\le i}$ with probability at most $|(V_i|xv_{<i}r) - (V_i|yv_{<i}r)|$.

If $v_{<i}$ is owned by P_x, then $V_i|xv_{<i}r$ has the same distribution as $V_i|xyv_{<i}r$; If $v_{<i}$ is owned by P_y, then $V_i|yv_{<i}r$ has the same distribution as $V_i|xyv_{<i}r$. Using Proposition 3.8 and Proposition 3.9, we have

$$\begin{aligned}
&\mathbb{E}\left[C_{ir}\right] \\
&\leq \mathop{\mathbb{E}}_{xyv_{<i}\in_{\mathrm{R}} XYV_{<i}}\left[\left|(V_i|xv_{<i}r)-(V_i|yv_{<i}r)\right|\right] \\
&\leq \mathop{\mathbb{E}}_{xyv_{<i}\in_{\mathrm{R}} XYV_{<i}}\left[\max\{\left|(V_i|xyv_{<i}r)-(V_i|yv_{<i}r)\right|,\left|(V_i|xyv_{<i}r)-(V_i|xv_{<i}r)\right|\right] \\
&\leq \mathop{\mathbb{E}}_{xyv_{<i}\in_{\mathrm{R}} XYV_{<i}}\left[\sqrt{\mathbb{D}\left(V_i|xyv_{<i}r||V_i|yv_{<i}r\right)+\mathbb{D}\left(V_i|xyv_{<i}r||V_i|xv_{<i}r\right)}\right] && \text{By Proposition 3.8} \\
&\leq \sqrt{\mathop{\mathbb{E}}_{xyv_{<i}\in_{\mathrm{R}} XYV_{<i}}\left[\mathbb{D}\left(V_i|xyv_{<i}r||V_i|yv_{<i}r\right)+\mathbb{D}\left(V_i|xyv_{<i}r||V_i|xv_{<i}r\right)\right]} && \text{by convexity} \\
&= \sqrt{I(X;V_i|YV_{<i}r)+I(Y;V_i|XV_{<i}r)} && \text{by Proposition 3.9}
\end{aligned}$$

Finally we apply the Cauchy Schwartz inequality to conclude that

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{\mathsf{CC}(\pi)} C_{ir}\right] &= \sum_{i=1}^{\mathsf{CC}(\pi)} \mathbb{E}\left[C_{ir}\right] \\
&\leq \sqrt{\mathsf{CC}(\pi)\sum_{i=1}^{\mathsf{CC}(\pi)} \mathbb{E}\left[C_{ir}\right]^2} \\
&\leq \sqrt{\mathsf{CC}(\pi)\sum_{i=1}^{\mathsf{CC}(\pi)} I(X;V_i|YV_{<i}r)+I(Y;V_i|XV_{<i}r)} \\
&= \sqrt{\mathsf{CC}(\pi)\left(I(X;V^{\mathsf{CC}(\pi)}|Yr)+I(Y;V^{\mathsf{CC}(\pi)}|Xr)\right)} \\
&= \sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_{\mu}(\pi_r)}
\end{aligned}$$

□

We then get that overall the expected number of mistakes is small:

Lemma 6.4. $\mathbb{E}\left[\#\text{ of mistakes in simulating }\pi\right] \leq \sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_{\mu}(\pi)}$.

Proof.

$$\begin{aligned}
\mathbb{E}\left[\#\text{ of mistakes in simulating }\pi\right] &= \mathop{\mathbb{E}}_{R}\left[\#\text{ of mistakes in simulating }\pi_R\right] \\
&\leq \mathop{\mathbb{E}}_{R}\left[\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_{\mu}(\pi_R)}\right] \\
&\leq \sqrt{\mathop{\mathbb{E}}_{R}\left[\mathsf{CC}(\pi)\cdot\mathsf{IC}_{\mu}(\pi_R)\right]} \\
&= \sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_{\mu}(\pi)}
\end{aligned}$$

□

Lemma 6.5. *The distribution of the leaf sampled by* $\tau_{\beta,\gamma}$ *is* $\gamma + \beta\frac{\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi)}}{\gamma}$*-close to the distribution of the leaf sampled by* π*.*

Proof. We show that in fact the probability that both players do not finish the protocol with the leaf $V_{\mathsf{CC}(\pi)}$ is bounded by $\gamma + \beta\frac{\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi)}}{\gamma}$. This follows from a simple union bound — the leaf $V_{\mathsf{CC}(\pi)}$ can be missed in two ways: either the number of mistakes on the correct path is larger than $\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi)}/\gamma$ (probability at most γ by Lemma 6.4 and Markov's inequality) or our protocol fails to detect all mistakes (for each mistake this happens with probability β). □

We set $\beta = \gamma^2/\mathsf{CC}(\pi)$. Then, since $\mathsf{CC}(\pi) \geq \mathsf{IC}_\mu(\pi)$, we get that the protocol errs with probability at most $\rho + 2\gamma$. On the other hand, by (1), the communication complexity of the protocol is at most $O(\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi)}\log(\mathsf{CC}(\pi)/\beta)/\gamma) = O(\sqrt{\mathsf{CC}(\pi)\cdot\mathsf{IC}_\mu(\pi)}\log(\mathsf{CC}(\pi)/\gamma)/\gamma)$. Setting $\epsilon = 2\gamma$ proves the theorem. □

7 Proofs for the Product Case

In this section we argue how to get a linear bound in the case that μ is a product distribution. We shall prove Theorem 1.7. Throughout this section we assume that the distribution μ on X, Y is a product distribution.

7.1 A proof sketch

We start with a rough proof sketch. Given function f, product distribution μ and protocol π, we want to come up with a protocol τ simulating π such that $\mathsf{CC}(\tau) = \tilde{O}(\mathsf{IC}_\mu(\pi))$. We assume that π is a private coin protocol for simplicity. This μ is a product distribution, we can prove inductively that if w is owned by the X player, then $O_{w,y}$ is independent of Y. Thus if a node is owned by the X player then he knows $O_{w,y}$ as well as $O_{w,x}$ (since the input y does not help in computing $O_{w,x}$ more than can be learned from the transcript). Hence for every non-leaf node w we denote by O_w the probability estimate by the party that does *not* own w. As a technical condition, we will assume that for every w, $O_{w,x}, O_{w,y} \in 1/2 \pm \beta$ for $\beta = 1/\operatorname{polylog}(\mathsf{CC}(\pi))$. This condition can be achieved for example by re-encoding π so that each party, instead of sending a bit b, sends $\operatorname{polylog}(\mathsf{CC}(\pi))$ random bits such that their majority is b.

For every node w owned by the X player, we define the *divergence at* w, denoted by D_w as $\mathbb{D}(O_{w,x}||O_{w,y})$ where $\mathbb{D}(p||q) = p\log(p/q) + (1-p)\log((1-p)/(1-q))$ equals the divergence (also known as the Kullback Leibler distance) between the p-biased coin and the q-biased coin. Given a node v, we define $\mathcal{B}_v$ to be the set of descendants w of v such that if we sum up $D_{w'}$ for all intermediate nodes w' on the path from v to w we get a total $< \beta$ but adding D_w makes the total at least β or w is a leaf. We define B_v to be the distribution over $\mathcal{B}_v$ that is induced by following the probabilities $O_{w'}$ along the path. Note that this distribution is known to both parties. We also define B_{vx}, B_{vy} to be the corresponding distributions where each player uses the actual probabilities on the nodes he owns.

The protocol proceeds as follows: (initially v is set to the root of the tree, t below is some large constant)

1. Both parties use their shared randomness to obtain a random element w according to the distribution B_v. (This involves first sampling a random leaf and then using binary search to

find the first location in which the divergence surpasses β.)

2. The X player sends a bit a_1 that equals 1 with probability $\min\{1, B_{vx}(w)/(tB_v(w)\}$.

3. The Y player sends a bit a_2 that equals 1 with probability $\min\{1, B_{vy}(w)/(tB_v(w)\}$.

4. If $a_1 = a_2 = 1$ then they set $v = w$. If v is a leaf they end the protocol, otherwise to go back to Step 1.

To get a rough idea why this protocol makes sense, consider the case that all the nodes in $\mathcal{B}_v$ are two levels below v, with the first node (i.e., v) owned by the X player, and the node in the intermediate level owned by the Y player. For a node $w \in \mathcal{B}_v$, let $B(w)$ be the true probability of arriving at w, and let $\tilde{B}(w) = B_v(w)$ be the estimated probability. Fixing w, we write $B(w) = B_1 B_2$ and $\tilde{B}(w) = \tilde{B}_1 \tilde{B}_2$, where B_i denotes the true probability that step i is taken according to w, and $\tilde{B}_i$ denotes this probability as estimated by the party that does not own the node.

The probability that w is output at the end of Step 1 is $\tilde{B}_1 \tilde{B}_2$. Now assume that the threshold t is set high enough so that we can assume that $tB_v(w) > B_{vx}(w), B_{vy}(y)$ with high probability. In this case the probability that w is accepted equals

$$\Pr[a_1 = 1] \Pr[a_2 = 1] = \left(\frac{B_1 \tilde{B}_2}{t\tilde{B}_1 \tilde{B}_2} \right) \left(\frac{\tilde{B}_1 B_2}{t\tilde{B}_1 \tilde{B}_2} \right) = \frac{B_1 B_2}{t^2 \tilde{B}_1 \tilde{B}_2} \tag{2}$$

thus the total probability that w is output is $\tilde{B}_1 \tilde{B}_2$ times (2) which is exactly its correct probability $B_1 B_2$ divided by t^2, and hence we get an overhead of t^2 steps, but output the right distribution over w.

7.2 The actual proof.

Again, we prove Theorem 1.7 via a reduction. We start with a protocol π with $\mathsf{CC}(\pi) = D_\rho^{\mu^n}(f^n)$ such that π computes f^n with probability of error at most ρ on inputs sampled according to μ^n. Our first step shall be to get a protocol that computes f^n but whose messages are *smoothed out* in the sense that every bit in the protocol is relatively close to being unbiased. We define a protocol that is a simulation of π in the following way: for a parameter β that we shall fix later, every time a player wants to send a bit in π, she instead sends $1000\frac{\log(\mathsf{CC}(\pi)/\gamma)}{\beta^2}$ bits which are each independently chosen to be the correct value with probability $1/2 + \beta$. The receiving player takes the majority of the bits sent to reconstruct the intended transmission. By the Chernoff bound, we have that the probability that any transmission is received incorrectly is at most $\gamma/\mathsf{CC}(\pi)$. By the union bound, this means that for every input, the distribution of the simulated transcript is γ-close to the correct distribution. We have thus argued the following claim:

Claim 7.1. *For every f, μ, ρ, γ, there exists a protocol π computing f^n with probability of error $\rho + \gamma$ on the distribution μ such that*

$$\mathsf{CC}(\pi) = O\left(\frac{D_\rho^{\mu^n}(f^n) \log(D_\rho^{\mu^n}(f^n)/\gamma)}{\beta^2} \right)$$

and for every x, y, v, i we have that

$$\Pr[\pi(x,y)_{i+1} = 1 | v^i] \in [1/2 - \beta, 1/2 + \beta].$$

In analogy with Theorem 1.5 and Theorem 4.1, Claim 7.1 leads to the following results:

Theorem 7.2 (Reduction to Small Information Content)**.** *For every $\gamma, \beta, \mu, f, \rho$ there exists a protocol τ computing f on inputs drawn from μ with probability of error at most $\rho + \gamma$, such that*

$$\mathsf{CC}(\tau) \leq O\left(\frac{D_\rho^{\mu^n}(f^n)\log(D_\rho^{\mu^n}(f^n)/\gamma)}{\beta^2}\right)$$

and

$$\mathsf{IC}_\mu(\tau) \leq O\left(\frac{\mathsf{CC}(\tau)\log\mathsf{CC}(\tau)}{n}\right).$$

Further, for every x, y, j, t, we have the j'th bit of the messages satisfies
$\Pr[\tau(x,y)_j = 1 | \tau(x,y)_{\leq j-1} = t] \in [1/2 - \beta, 1/2 + \beta]$.

For f^{+n} we have the following theorem:

Theorem 7.3 (Reduction to Small Information Content)**.** *For every distribution μ, there exists a protocol τ computing f with probability of error $\rho + \gamma$ over the distribution μ with*

$$\mathsf{CC}(\tau) \leq O\left(\frac{D_\rho^{\mu^n}(f^{+n})\log(D_\rho^{\mu^n}(f^{+n})/\gamma)}{\beta^2}\right) + 2\log K$$

such that if τ' is the protocol that simulates all but the last $2\log K$ bits of τ, then

$$\mathsf{IC}_\mu(\tau') \leq O\left(\frac{\mathsf{CC}(\tau')\log\mathsf{CC}(\tau')}{n}\right)$$

Further, for every x, y, j, t, we have the j'th bit of the messages satisfies
$\Pr[\tau'(x,y)_j = 1 | \tau(x,y)_{\leq j-1} = t] \in [1/2 - \beta, 1/2 + \beta]$.

To complete the proof, we need to show how to simulate the protocols τ in the above theorems with small communication complexity. We shall do this by proving the following theorem:

Theorem 7.4. *There exists a constant k such that for every $\epsilon > 0$, if π is a protocol such that for every x, y, v, i we have that*

$$\Pr[\pi(x,y)_{i+1} = 1 | v_{\leq i}] \in \left[\frac{1}{2} - \frac{1}{k\log(\mathsf{CC}(\pi)/\epsilon)}, \frac{1}{2} + \frac{1}{k\log(\mathsf{CC}(\pi)/\epsilon)}\right]$$

Then for every product distribution μ on inputs X, Y there exists a protocol τ and a function p such that for every x, y and every transcript l of π,

$$\frac{\Pr[p(\tau(x,y)) = l]}{\Pr[\pi(x,y) = l]} \leq \exp(O(\epsilon))$$

and the expected communication complexity of τ under the distribution μ is at most

$$\exp(O(\epsilon)) \cdot \mathsf{IC}_\mu(\pi) \cdot k\log(\mathsf{CC}(\pi)/\epsilon).$$

We use the above theorems to prove our final theorem about product distributions:

Proof of Theorem 1.7. Let $\alpha > 0$ be the parameter in the statement of Theorem 1.7. We set $\gamma = \alpha/4$, and set $\epsilon = O(\alpha)$, in which the constant is small enough so that both $\exp O(\epsilon)$'s in Theorem 7.4 are at most $1 + \alpha/4$. Let π be a protocol satisfying the conclusions of Theorem 7.2 with β set to $\beta = \frac{1}{k' \log(\mathsf{D}_\rho^{\mu^n}(f^n)/\epsilon)}$ for some constant k'. k' can be chosen to be large enough so that

$$\beta = \frac{1}{k' \log(\mathsf{D}_\rho^{\mu^n}(f^n)/\epsilon)} \leq \frac{1}{k \log(\mathsf{CC}(\pi)/\epsilon)}$$

as in Theorem 7.4.

By Theorem 7.4, we get a protocol computing f on the distribution μ with error $(\rho+\gamma)\exp(O(\epsilon))$ and expected communication $\exp(O(\epsilon))\mathsf{D}_\rho^{\mu^n}(f^n)\operatorname{polylog}(\mathsf{D}_\rho^{\mu^n}(f^n)/\gamma)/n$. Markov's inequality implies that by interrupting the protocol if it runs for too long, we can get a protocol that errs with probability $(\rho+\gamma)\exp(O(\epsilon))+\lambda$ and has communication complexity $\exp(O(\epsilon))\mathsf{D}_\rho^{\mu^n}(f^n)\operatorname{polylog}(\mathsf{D}_\rho^{\mu^n}(f^n)/\gamma)/\lambda n$. Set $\lambda = \alpha/4$ to prove the theorem.

□

The proof for Theorem 1.10 is almost exactly the same, so we omit it.

7.3 Proof of Theorem 7.4

It only remains to prove Theorem 7.4. Set $\beta = 1/k \log(\mathsf{CC}(\pi)/\epsilon)$. We need the following definition:

Definition 7.5 (Conditional Divergence)**.** Given a protocol π, a prefix v of the transcript and $j \in [\mathsf{CC}(v)]$, we define the j'th step divergence cost as

$$\mathbb{D}_{x,j}^{\pi}(v) \overset{def}{=} \mathbb{D}\left((\pi(x,Y)_j | v_{\leq j-1}) || (\pi(X,Y)_j | v_{\leq j-1})\right)$$

$$\mathbb{D}_{y,j}^{\pi}(v) \overset{def}{=} \mathbb{D}\left((\pi(X,y)_j | v_{\leq j-1}) || (\pi(X,Y)_j | v_{\leq j-1})\right)$$

We define the divergence cost for the whole prefix as the sum of the step divergence costs

$$\mathbb{D}_{x}^{\pi}(v) \overset{def}{=} \sum_{j=1}^{\mathsf{CC}(v)} \mathbb{D}_{x,j}^{\pi}(v), \qquad \mathbb{D}_{y}^{\pi}(v) \overset{def}{=} \sum_{j=1}^{\mathsf{CC}(v)} \mathbb{D}_{y,j}^{\pi}(v)$$

It is easy to check that

$$\mathop{\mathbb{E}}_{X,Y,\pi(X,Y)}\left[\mathbb{D}_{X}^{\pi}(\pi(X,Y)) + \mathbb{D}_{Y}^{\pi}(\pi(X,Y))\right] = \mathsf{IC}_\mu(\pi)$$

Thus the conditional divergence is in some sense a measure of the amount of information revealed by the relevant prefix of the transcript. Observe that $\mathbb{D}_x^\pi(v)$ is a function only of x and v. Further, we have that if the node corresponding to $v_{\leq j-1}$ is owned by x, then $\mathbb{D}_{y,j}^{\pi}(v) = 0$, since conditioned on $v_{\leq j-1}$, Y is independent of V_j.

We use the fact that the bits in our protocol are close to uniform to show that the step divergence is at most $O(\beta)$ for each step:

Proposition 7.6. *For every j, $\mathbb{D}_{x,j}^{\pi}(v)$ and $\mathbb{D}_{y,j}^{\pi}(v)$ are bounded by $O(\beta)$.*

Proof. This follows from the fact that all probabilities for each step lie in $[1/2-\beta, 1/2+\beta]$. The worst the divergence between two distributions that lie in this range can be is clearly $\log\left(\frac{1/2+\beta}{1/2-\beta}\right) = \log(1+O(\beta)) = O(\beta)$. □

Next, for every prefix v of the transcript, and inputs x, y, we define a subset of the prefixes of potential transcripts that start with v, $\mathcal{B}_{vxy}$ in the following way: we include w in $\mathcal{B}_{vxy}$ if and only if for every w' that is a strict prefix of w,

$$\max\left\{\sum_{j=\mathsf{CC}(v)+1}^{\mathsf{CC}(w')} \mathbb{D}^{\pi}_{x,j}(w'), \sum_{j=\mathsf{CC}(v)+1}^{\|w'\|} \mathbb{D}^{\pi}_{y,j}(w')\right\} < \beta,$$

and we have that w itself is either a leaf or satisfies

$$\max\left\{\sum_{j=\mathsf{CC}(v)+1}^{\mathsf{CC}(w)} \mathbb{D}^{\pi}_{x,j}(w), \sum_{j=\mathsf{CC}(v)+1}^{\|w\|} \mathbb{D}^{\pi}_{y,j}(w)\right\} \geq \beta.$$

The set $\mathcal{B}_{vxy}$ has the property that every path from v to a leaf of the protocol tree must intersect exactly one element of $\mathcal{B}_{vxy}$, i.e. if we cut all paths at the point where they intersect $\mathcal{B}_{vxy}$, we get a protocol tree that is a subtree of the original tree. We define the distribution B_{vxy} on the set $\mathcal{B}_{vxy}$ as the distribution on $\mathcal{B}_{vxy}$ induced by the protocol π. Namely we sample from B_{vxy} by sampling from $\pi(x,y)|v$ and then taking the unique vertex of $\mathcal{B}_{vxy}$ that the sampled path intersects. Similarly, we define the distributions B_{vx}, B_{vy}, B_v on $\mathcal{B}_{vxy}$ to be the distributions obtained by first sampling a path according to $\pi(x,Y)|v, \pi(X,y)|v, \pi(X,Y)|v$ and then taking the unique vertices in $\mathcal{B}_{vxy}$ that these paths intersect. For every transcript w, the players can compute the element of $\mathcal{B}_{vxy}$ that intersects the path w by communicating $2\log \mathsf{CC}(\pi)$ bits.

Given, these definitions, we are now ready to describe our simulation protocol. The protocol proceeds in rounds. In each round the players shall use rejection sampling to sample some consecutive part of the transcript.

7.3.1 A single round

The first protocol, shown in Figure 9 assumes that we have already sampled the prefix v. We define the protocol for some constant t that we shall set later.

Note that $B_v(w) = \prod_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(w)} \Pr[\pi(X,Y)_{\leq i+1} = w_{\leq i+1} | \pi(X,Y)_{\leq i} = w_{\leq i}]$. We write $B^x_v(w)$ to denote the part of this product that corresponds to nodes sampled by P_x, and $B^y_v(w)$ to denote the part that corresponds to nodes sampled by P_y. Thus $B_v = B^x_v B^y_v$. We use B^x_{vx}, B^y_{vx} etc to denote the analogous functions. Then note that since X, Y are independent, we have that $B^y_{vx} = B^y_v$. Thus we get

$$\left(\frac{B_{vx}}{B_v}\right)\left(\frac{B_{vy}}{B_v}\right) = \left(\frac{B^y_v B^x_{vxy}}{B^x_v B^y_v}\right)\left(\frac{B^y_{vxy} B^x_v}{B^x_v B^y_v}\right) = \frac{B^x_{vxy} B^y_{vxy}}{B^x_v B^y_v} = \frac{B_{vxy}}{B_v} \tag{3}$$

This suggests that our protocol should pick a transcript distributed according to B_{vxy}. We shall argue that the subsequent prefix of the transcript sampled by the protocol in Figure 9 cannot be sampled with much higher probability than what it is sampled with in the real distribution. Let B'_{vxy} denote the distribution of the accepted prefix of $\tau_{v,t}$.

Protocol $\tau_{v,t}$

1. Both players use public randomness to sample a path according to $\pi(X,Y)|v$ and communicate $2\log \mathsf{CC}(\pi)$ bits to sample an element w of $\mathcal{B}_{vxy}$ according to the distribution B_v.
2. P_x samples a bit a_1 which is 1 with probability
$$\min\left\{\frac{B_{vx}(w)}{tB_v(w)}, 1\right\}.$$
3. P_y samples a bit a_2 which is 1 with probability
$$\min\left\{\frac{B_{vy}(w)}{tB_v(w)}, 1\right\}.$$
4. If both a_1 and a_2 were 1, they accept w. Else they repeat the protocol.

Figure 9: The protocol to sample a subsequent part of the transcript

Claim 7.7 (No sample gets undue attention)**.** *For every prefix w,*
$$B'_{vxy}(w)/B_{vxy}(w) \le 1 + 2\exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right)$$

We shall also show that the expected communication complexity of this protocol is not too high:

Claim 7.8 (Small number of rounds)**.** *The expected communication complexity of τ_v is at most*
$$\frac{O(t^2)}{1 - \exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right)}$$

Claim 7.7 and Claim 7.8 will follow from the following claim:

Claim 7.9.
$$\Pr_{w \in_R B_{vxy}}\left[\frac{B_{vx}(w)}{B_v(w)} \ge t\right] \le \exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right), \quad \Pr_{w \in_R B_{vxy}}\left[\frac{B_{vy}(w)}{B_v(w)} \ge t\right] \le \exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right)$$

Let us first argue that Claim 7.7 follows from Claim 7.9.

Proof of Claim 7.7. Set a to be the function that maps any $w \in \mathcal{B}_{vxy}$ to $\min\left\{(1/t)\frac{B_{vx}(w)}{B_v(w)}, 1\right\} \cdot \min\left\{(1/t)\frac{B_{vy}(w)}{B_v(w)}, 1\right\}$. Set $a' = (1/t)\frac{B_{vx}(w)}{B_v(w)}(1/t)\frac{B_{vy}(w)}{B_v(w)}$. Then clearly $a'(w) \ge a(w)$ for every w. Applying Equation 3, we get

$$a' = (1/t^2)\left(\frac{B_{vx}}{B_v}\right)\left(\frac{B_{vy}}{B_v}\right) = (1/t^2)\frac{B_{vxy}}{B_v}$$

Thus $B_{vxy} = \beta a' \cdot B_v$ for some constant β. By Proposition B.3, applied to a', a and the distributions $D = B_{vxy}, D' = B'_{vxy}$, we have that for every w,

$$\frac{B'_{vxy}(w)}{B_{vxy}(w)} \leq \frac{1}{1 - \Pr_{w \in_{\mathrm{R}} B_{vxy}}[a'(w) > a(w)]}$$

On the other hand, by the union bound and Claim 7.9,

$$\Pr_{w \in_{\mathrm{R}} B_{vxy}}[a'(w) > a(w)] \leq \Pr_{w \in_{\mathrm{R}} B_{vxy}}\left[\frac{B_{vx}(w)}{B_v(w)} > t \vee \frac{B_{vy}(w)}{B_v(w)} > t\right] \leq 2\exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right)$$

Since $1/(1-z) \leq 1 + O(z)$ for $z \in (0, 1/10)$, we get Claim 7.7. □

Now we show Claim 7.8 assuming Claim 7.9.

Proof of Claim 7.8. We shall use Proposition B.4. We need to estimate the probability that the first round of $\tau_{v,t}$ accepts its sample. This probability is exactly

$$\sum_{w \in \mathcal{B}_{vxy}} B_v(w) \min\left\{(1/t)\frac{B_{vx}(w)}{B_v(w)}, 1\right\} \cdot \min\left\{(1/t)\frac{B_{vy}(w)}{B_v(w)}, 1\right\}$$

Let $A \subset \mathcal{B}_{vxy}$ denote the set $\{w : \frac{B_{vx}(w)}{B_v(w)} \leq t \wedge \frac{B_{vy}(w)}{B_v(w)} \leq t\}$. Then we see that the above sum can be lower bounded:

$$\sum_{w \in \mathcal{B}_{vxy}} B_v(w) \min\left\{(1/t)\frac{B_{vx}(w)}{B_v(w)}, 1\right\} \cdot \min\left\{(1/t)\frac{B_{vy}(w)}{B_v(w)}, 1\right\}$$
$$\geq (1/t^2) \sum_{w \in A} B_v(w) \left(\frac{B_{vx}(w)}{B_v(w)}\right)\left(\frac{B_{vy}(w)}{B_v(w)}\right) = (1/t^2) \sum_{w \in A} B_{vxy},$$

where the last equality follows from Equation 3.

Finally, we see that Claim 7.9 implies that $\sum_{w \in A} B_{vxy} \geq 1 - \exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right)$. Proposition B.4 then gives the bound we need.

□

Next we prove Claim 7.9. To do this we shall need to use a simple generalization of Azuma's inequality, which we prove in Appendix A.

Proof of Claim 7.9. Let W be a random variable distributed according to B_{vxy}. Set $Z_{\mathsf{CC}(v)+1}, \ldots, Z_{\mathsf{CC}(\pi)}$ to be real valued random variables such that if $i \leq \mathsf{CC}(W)$,

$$Z_i = \log\left(\frac{\Pr[\pi(x,Y)_i = W_i | vW_{\leq i-1}]}{\Pr[\pi(X,Y)_i = W_{\leq i} | vW_{\leq i-1}]}\right).$$

If $i > \mathsf{CC}(W)$, set $Z_i = 0$. Observe that $\mathbb{E}\left[Z_i | w_{\leq i-1}\right] = \mathbb{D}^{\pi}_{x,i}(w)$. We also have that

$$\sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)} Z_i = \log\left(\frac{\Pr[\pi(x,Y)^{\mathsf{CC}(w)} = w|v]}{\Pr[\pi(X,Y)^{\mathsf{CC}(w)} = w|v]}\right)$$
$$= \log\left(\frac{B_{vx}(w)}{B_v(w)}\right) \quad (4)$$

Next set $T_i = Z_i - \mathbb{E}[Z_i | Z_{i-1}, \ldots, Z_1]$. Note that $\mathbb{E}[T_i | T_{i-1}, \ldots, T_1] = 0$ (in fact the stronger condition that $\mathbb{E}[T_i | Z_{i-1}, \ldots, Z_1] = 0$ holds). For every $w \in \mathcal{B}_{vxy}$, we have that

$$\sup(T_i | w_{\leq i-1}) \leq \max \left\{ \log \left(\frac{\Pr[\pi(x,Y)_i = 0 | w_{\leq i-1}]}{\Pr[\pi(X,Y)_i = 0 | w_{\leq i-1}]} \right), \log \left(\frac{\Pr[\pi(x,Y)_i = 1 | w_{\leq i-1}]}{\Pr[\pi(X,Y)_i = 1 | w_{\leq i-1}]} \right) \right\}$$

$$\inf(T_i | w_{\leq i-1}) \geq \min \left\{ \log \left(\frac{\Pr[\pi(x,Y)_i = 0 | w_{\leq i-1}]}{\Pr[\pi(X,Y)_i = 0 | w_{\leq i-1}]} \right) - \mathbb{D}^{\pi}_{x,i}(w), \log \left(\frac{\Pr[\pi(x,Y)_i = 1 | w_{\leq i-1}]}{\Pr[\pi(X,Y)_i = 1 | w_{\leq i-1}]} \right) - \mathbb{D}^{\pi}_{x,i}(w) \right\}$$

By Proposition 3.8 and using the fact that $\pi(x,Y) = 1 \in [1/2 - \beta, 1/2 + \beta]$ we can bound

$$\begin{aligned} \sup(T_i | w_{\leq i-1}) &\leq \log \left(\frac{1/2 - \beta + \sqrt{\mathbb{D}^{\pi}_{x,i}(w)}}{1/2 - \beta} \right) \\ &= \log \left(1 + O\left(\sqrt{\mathbb{D}^{\pi}_{x,i}(w)} \right) \right) \\ &= O\left(\sqrt{\mathbb{D}^{\pi}_{x,i}(w)} \right) \end{aligned} \tag{5}$$

$$\begin{aligned} \inf(T_i | w_{\leq i-1}) &\geq \log \left(\frac{1/2 - \beta}{1/2 - \beta + \sqrt{\mathbb{D}^{\pi}_{x,i}(w)}} \right) - \mathbb{D}^{\pi}_{x,i}(w) \\ &= \log \left(1 - O\left(\sqrt{\mathbb{D}^{\pi}_{x,i}(w)} \right) \right) \\ &= -O\left(\sqrt{\mathbb{D}^{\pi}_{x,i}(w)} \right), \end{aligned} \tag{6}$$

as long as $\beta < 1/10$.

Equation 5 and Equation 6 imply that for $w \in \mathcal{B}_{vxy}$,

$$\sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)} (\sup(T_i) - \inf(T_i) | w_{\leq i-1})^2 \leq \sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)} O(\mathbb{D}^{\pi}_{x,i}(w)) = O(\beta) \tag{7}$$

For every w,

$$\begin{aligned} \left(\sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)} T_i \right) | w &= \left(\sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)} Z_i \right) | w - \sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)} \mathbb{E}[Z_i | w_{\leq i-1}] \\ &= \sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(w)} \log \left(\frac{\Pr[\pi(x,Y)_i = w_i | v\pi(x,Y)_{\leq i-1} = w_{\leq i-1}]}{\Pr[\pi(X,Y)_i = w_i | \pi(X,Y)_{\leq i-1} = vw_{\leq i-1}]} \right) - \sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(w)} \mathbb{D}^{\pi}_{x,i}(w) \\ &\geq \log \left(\frac{B_{vx}(w)}{B_v(w)} \right) - O(\beta) \end{aligned} \tag{8}$$

where the last inequality follows from the definition of $\mathcal{B}_{vxy}$, Proposition 7.6 and Equation 4.

Thus we can use Theorem A.1 to bound

$$
\begin{aligned}
&\Pr_{w\in_{\mathrm{R}} B_{vxy}}\left[\frac{B_{vx}(w)}{B_v(w)} \geq t\right] \\
&\leq \Pr_{w\in_{\mathrm{R}} B_{vxy}}\left[\log\left(\frac{B_{vx}(w)}{B_v(w)}\right) \geq \log t\right] \\
&\leq \Pr\left[\sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)} T_i \geq \log t - O(\beta)\right] && \text{by Equation 8} \\
&\leq \exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\sum_{i=\mathsf{CC}(v)+1}^{\mathsf{CC}(\pi)}(\sup(T_i) - \inf(T_i)|w_{\leq i-1})^2}\right)\right) \\
&\leq \exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right) && \text{by Equation 7}
\end{aligned}
$$

□

7.3.2 The whole protocol

Our final protocol for computing f is shown in Figure 10.

Protocol τ_t
1. The players publicly sample the public randomness $v \in_{\mathrm{R}} R$ for π.   2. The players repeatedly run $\tau_{v,t}$ to get a new prefix v. They stop only when they reach a leaf of the protocol tree for π.

Figure 10: The protocol to sample a subsequent part of the transcript

We first argue that our simulation returns the correct answer with decent probability. We shall actually argue that the probability for any returned transcript does not increase by too much. To ease notations, let us set

$$\alpha \overset{def}{=} \exp\left(-\Omega\left(\frac{(\log t - O(\beta))^2}{\beta}\right)\right)$$

Set t to be a large enough constant so that $\alpha = \exp(-\Omega(1/\beta)) = \exp(-\Omega(k\log(\mathsf{CC}(\pi)/\epsilon)))$. Set k to be large enough so that $\alpha \leq \epsilon/\mathsf{CC}(\pi)$.

Let L denote the random variable of the sampled transcript returned by τ_t. Then by Claim 7.7, we get that for every leaf l,

$$\frac{\Pr[L = l|xy]}{\Pr[\pi(x,y) = l]} \leq (1+\alpha)^{\mathsf{CC}(\pi)} = \exp(O(\epsilon)) \tag{9}$$

Next we bound the expected communication of the protocol. First observe that if the protocol accepts a leaf l, then the protocol must have involved $O((\mathbb{D}_x^\pi(l) + \mathbb{D}_y^\pi(l))/\beta)$ rounds. The expected number of bits communicated in each of these rounds is independent of l by Proposition B.1, and is $\frac{t^2}{1-\alpha}$ by Claim 7.8. Thus the expected communication complexity of the protocol can be bounded

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x,y,l \in_{\mathrm{R}} X,Y,L}\left[O\left((\mathbb{D}_x^\pi(l) + \mathbb{D}_y^\pi(l))\frac{t^2}{\beta(1-\alpha)}\right)\right] \\
&= \frac{O(t^2)}{\beta(1-\alpha)} \mathop{\mathbb{E}}_{x,y \in_{\mathrm{R}} X,Y}\left[\sum_l \Pr[L = l|x,y](\mathbb{D}_x^\pi(l) + \mathbb{D}_y^\pi(l))\right] \\
&\le O(1/\beta) \mathop{\mathbb{E}}_{x,y \in_{\mathrm{R}} X,Y}\left[\sum_l \exp(O(\epsilon)) \Pr[\pi(x,y) = l](\mathbb{D}_x^\pi(l) + \mathbb{D}_y^\pi(l))\right] \qquad \text{by Equation 9} \\
&= \frac{\exp(O(\epsilon))}{\beta} \mathop{\mathbb{E}}_{X,Y}\left[\mathbb{D}_X^\pi(\pi(X,Y)) + \mathbb{D}_Y^\pi(\pi(X,Y))\right] \\
&= \frac{\exp(O(\epsilon))}{\beta} \cdot \mathsf{IC}_\mu(\pi)
\end{aligned}$$

This completes the proof of Theorem 7.4.

8 Open problems and final thoughts

The main problem that remains open is whether optimal, or near-optimal compression is possible for protocols in the general setting.

Open Problem: Is there a generic way to convert any two-party protocol π over a general distribution μ into a protocol that uses only $\mathsf{IC}_\mu(\pi)\,\mathrm{polylog}(\mathsf{CC}(\pi))$ bits of communication?

An affirmative answer to this problem would immediately yield an optimal direct-sum theorem for randomized communication complexity, showing that the communication complexity of f^n is $\tilde{O}(n)$ times as high as the communication complexity of f. Curiously enough, it turns out [BR] that the converse is true as well, and the problem above is *complete* for randomized communication direct sum — one can show that if there is no such compression scheme, then there is a (partial) function U for which a direct sum theorem fails to hold.[6] In this function U, each player gets as input a protocol tree, as well as the probabilities for all the nodes he owns, and the output is simply the output of the protocol. Unfortunately, by design, information theoretic techniques seem to be powerless in proving lower bounds for U.

Acknowledgements

We thank Noga Alon, Emanuel Milman, Alex Samorodnitsky, Avi Wigderson and Amir Yehudayoff for useful discussions.

[6]In a partial function / promise problem the protocol only needs to compute the function if the pair of inputs come from some subset. Our results in this paper for non-product distributions carry over to promise problem as well.

References

[BR] M. Braverman and A. Rao. Work in progress.

[BYJKS04] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[CSWY01] A. Chakrabarti, Y. Shi, A. Wirth, and A. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In B. Werner, editor, *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 270–278, Los Alamitos, CA, Oct. 14–17 2001. IEEE Computer Society.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley series in telecommunications. J. Wiley and Sons, New York, 1991.

[FKNN91] T. Feder, E. Kushilevitz, M. Naor, and N. Nisan. Amortized communication complexity. *SIAM Journal on Computing*, 24(4):736–750, 1995. Prelim version by Feder, Kushilevitz, Naor FOCS 1991.

[FPRU94] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

[GF81] G. Galbiati and M. Fischer. On the complexity of 2-output boolean networks. *Theor. Comput. Sci.*, 16:177–185, 1981.

[HJMR07] P. Harsha, R. Jain, D. A. McAllester, and J. Radhakrishnan. The communication complexity of correlation. In *IEEE Conference on Computational Complexity*, pages 10–23. IEEE Computer Society, 2007.

[JHM$^+$98] M. Jerrum, M. Habib, C. McDiarmid, J. L. Ramirez-Alfonsin, and B. Reed. *Probabilistic Methods for Algorithmic Discrete Mathematics*, volume 16 of *Algorithms and Combinatorics*. Springer-Verlag, 1998.

[JRS03] R. Jain, J. Radhakrishnan, and P. Sen. A direct sum theorem in communication complexity via message compression. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 300–315. Springer, 2003.

[KN97] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, Cambridge, 1997.

[KRW91] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3/4):191–204, 1995. Prelim version CCC 1991.

[KS92] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, Nov. 1992.

[New91] I. Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 31 July 1991.

[Pau76] W. Paul. Realizing boolean functions on disjoint sets of variables. *Theor. Comput. Sci.*, 2:383–396, 1976.

[Raz92] Razborov. On the distributed complexity of disjointness. *TCS: Theoretical Computer Science*, 106, 1992.

[Raz95] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998. Prelim version in STOC '95.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948. Monograph B-1598.

[Sha03] R. Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1-2):1–22, 2003.

[Uhl74] D. Uhlig. On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. *Matematicheskie Zametki*, 15(6):937–944, 1974.

[Yao82] A. C.-C. Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91. IEEE, 1982.

A A simple generalization of Azuma's inequality

We shall need the following theorem, whose proof appears in [JHM+98]. For completeness, we reproduce the part of the proof we need here:

Theorem A.1 (Azuma)**.** *Let $T_1, \dots, T_k$ be real valued random variables such that for every i, we have $\mathbb{E}[T_i | T_{i-1}, \dots, T_i] \le 0$. Set $A_i = (\sup(T_i) - \inf(T_i) | T_{i-1}, \dots, T_1)^2$. Then if $\sum_{i=1}^k A_i \le c$, for every $\alpha > 0$,*

$$\Pr\left[\sum_{i=1}^k T_i \ge \alpha\right] \le \exp(-2\alpha^2/c).$$

To prove the theorem, we need the following lemma appearing as Lemma 2.6 in [JHM+98]:

Lemma A.2. *Let X be a real valued random variable with $\mathbb{E}[X] = 0$ and $X \in [a, b]$ almost surely. Then $\mathbb{E}[\exp(X)] \le \exp\left(\frac{(b-a)^2}{8}\right)$.*

Proof of Theorem A.1. First, we assume without loss of generality that $\mathbb{E}[T_i | T_{i-1}, \dots, T_i] \le 0$. We can do this by changing each random variable T_i to $T_i - \mathbb{E}[T_i | T_{i-1}, \dots, T_1]$. This does not change any of the conditions above, and only increases $\Pr[\sum_i T_i \ge \alpha]$.

By Markov's inequality, for every positive λ we have

$$\Pr\left[\sum_{i=1}^k T_i \ge \alpha\right] = \Pr\left[\exp\left(\lambda \sum_{i=1}^k T_i\right) > \exp(\lambda\alpha)\right] \le \mathbb{E}\left[\exp\left(\lambda \sum_{i=1}^k T_i\right)\right] \exp(-\lambda\alpha)$$

Next we show by induction on k that $\mathbb{E}\left[\exp\left(\lambda \sum_{i=1}^k T_i\right)\right] \le \sup\left(\prod_{i=1}^k \mathbb{E}[\exp(\lambda T_i) | T_{i-1}, \dots, T_1]\right)$. The case $k = 1$ is trivial. For general k we compute

$$
\begin{aligned}
\mathbb{E}\left[\exp\left(\lambda\sum_{i=1}^{k} T_i\right)\right] &= \mathbb{E}\left[\exp\left(\lambda T_1\right)\mathbb{E}\left[\exp\left(\lambda\sum_{i=2}^{k} T_i\right)|T_1\right]\right] \\
&\le \mathbb{E}\left[\exp\left(\lambda T_1\right)\right]\sup\left(\prod_{i=2}^{k}\mathbb{E}\left[\exp(\lambda T_i)|T_{i-1},\dots,T_1\right]\right) && \text{by induction} \\
&= \sup\left(\mathbb{E}\left[\exp\left(\lambda T_1\right)\right]\prod_{i=2}^{k}\mathbb{E}\left[\exp(\lambda T_i)|T_{i-1},\dots,T_1\right]\right) \\
&= \sup\left(\prod_{i=1}^{k}\mathbb{E}\left[\exp(\lambda T_i)|T_{i-1},\dots,T_1\right]\right)
\end{aligned}
$$

Thus we can bound

$$
\begin{aligned}
\Pr\left[\sum_{i=1}^{k} T_i \ge \alpha\right] &\le \exp(-\lambda\alpha)\sup\left(\prod_{i=1}^{k}\mathbb{E}\left[\exp(\lambda T_i)|T_{i-1},\dots,T_1\right]\right) \\
&\le \exp(-\lambda\alpha)\sup\left(\prod_{i=1}^{k}\exp\left(\frac{\lambda^2 A_i}{8}\right)\right) && \text{by Lemma A.2} \\
&= \exp(-\lambda\alpha)\sup\left(\exp\left(\frac{\sum_{i=1}^{k}\lambda^2 A_i}{8}\right)\right) \\
&= \exp(-\lambda\alpha)\exp\left(\sup\left(\frac{\sum_{i=1}^{k}\lambda^2 A_i}{8}\right)\right) \\
&\le \exp(-\lambda\alpha+\lambda^2 c/8)
\end{aligned}
$$

Setting $\lambda = 4\alpha/c$, we get that

$$
\Pr\left[\sum_{i=1}^{k} T_i \ge \alpha\right] \le \exp(-2\alpha^2/c)
$$

□

B Analyzing Rejection Sampling

In this section we give some basic facts about *rejection sampling*. For a distribution C supported on some finite set $\mathcal{C}$ and a function $a : \mathcal{C} \to [0,1]$, Figure 11 describes a generic rejection sampling algorithm.

We prove some simple properties of this kind of sampling. Let D' denote the random variable of the sampled element. Let R denote the random variable that counts the number of rounds before the algorithm accepts the sample. Then we see that D' is independent of R, since for any integers c, c', $D'|R = c$ has the same distribution as $D'|R = c'$.

```
Algorithm Rejection Sampling.

1. Sample an element $z \in_R C$.
2. Accept it with probability $a(z)$, else go to the first step.
```

Figure 11: Generic Rejection Sampling

Proposition B.1. *D' is independent of R.*

We then see that $D'(w) = \Pr[(R = 1) \wedge w \text{ is accepted}]/\Pr[R = 1] = C(w)a(w)/\Pr[R = 1]$. We have shown the following claim:

Claim B.2. *For some constant α, $D' = \alpha a \cdot C$.*

Now let $a' : \mathcal{C} \to [0, 1]$ be a function such that $a'(w) \geq a(w)$ for all $w \in \mathcal{C}$, and let D denote the random variable of the sampled element. Set $b = a' - a$. Then $D = \beta a' \cdot C = \beta C \cdot (a + b)$ for some $\beta > 0$. Thus, by Claim B.2, there exists a distribution D'' such that D' is a convex combination $D = \beta' D'' + (1 - \beta')D'$. In particular, this implies that $\frac{D'(w)}{D(w)} \leq \frac{1}{1-\beta'}$. We bound $\beta' \leq \Pr[D' \in \mathsf{Supp}(D'')] = \Pr_{w \in_R D}[a'(w) > a(w)]$. This gives us the following two bounds:

Proposition B.3. *Let $D = \beta a' \cdot C$ be a distribution such that $a'(w) \geq a(w)$ for every w. Then for every w,*

$$\frac{D'(w)}{D(w)} \leq \frac{1}{1 - \Pr_{w \in_R D}[a'(w) > a(w)]}.$$

Proposition B.4. *The expected number of rounds that the above protocol runs for is $1/\Pr[R = 1]$.*

Proof. From the construction, we see that $\mathbb{E}[R] = \Pr[R = 1] + (1 - \Pr[R = 1])(1 + \mathbb{E}[R])$. Rewriting this, we get $\mathbb{E}[R] = 1/\Pr[R = 1]$. □

C Finding The First Difference in Inputs

Proof Sketch for Lemma 3.15. Without loss of generality, we assume that $k = 2^t$ for an integer t (if not, we can always pad the input strings with 0's until the lengths are of this form before running the protocol). For a parameter C, we define a labeled tree of depth $C \log(k/\epsilon) = C(t + \log(1/\epsilon)$ as follows. The root of the tree is labeled by the interval $[1, 2^t]$. For i ranging from 0 to $t - 1$, every node at depth i labeled by $[a, b]$ has two children, corresponding to splitting the interval $[a, b]$ into equal parts. Thus the left one is labeled by the interval $[a, b - 2^{t-i+1}]$ and the right one is labeled by $[a + 2^{t-i+1}, b]$. Thus at depth t there are 2^t nodes, each labeled by $[a, a]$ for distinct a's from $[2^t]$. Every node at depth $\geq t$ has exactly one child, labeled the same as the parent.

In the protocol, the players shall try to narrow down where the first difference in their inputs is by taking a walk on the tree. At each step, the players first check that the interval they are on is correct, and then try to narrow down their search. For any integer $a \in [k]$, let x_a denote the prefix of x of length a. To check whether a given interval $[a, b]$ contains the index that they seek, the players will use public randomness to pick random functions $h_1 : \{0, 1\}^a \to [18]$ and

$h_2 : \{0,1\}^b \to [18]$ and compare $h_1(x_a)$ with $h_1(y_a)$ and $h_2(x_b)$ with $h_2(y_b)$. The probability of getting an incorrect answer is thus at most $1/9$.

For a parameter C, the protocol works as follows:

1. The players set v to be the root of the tree.
2. The players run the tests described above to check whether the index with the first difference lies in the interval corresponding to v and in those corresponding to v's children. If the tests are consistent, and indicate that the interval for v does not contain the index, the players set v to be the parent of the old v (or leave it unchanged if v is the root). If the tests are consistent and indicate that the interval of one of the children contains the index, the players set v to be that child. If the tests are inconsistent, the players leave v unchanged.
3. Step 2 is repeated $C(t + \log(1/\epsilon))$ times.
4. If the final vertex is labeled by an interval of the form $[a, a]$, output a. Else conclude that the input strings are equal.

To analyze the protocol, fix x and y. Note that if $x = y$, then the protocol never fails. So let us assume that $x \neq y$ and assume that a is the first index at which x, y differ. Then let w denote the vertex in the tree of largest depth that is labeled by $[a, a]$. Next we direct the edges of the tree so that at every vertex, the only outgoing edge points to the neighbor that is closer to w in terms of shortest path distance. Then observe that at every step of our protocol, v is changed to a neighbor that is closer to w with probability at least $2/3$. Further, our protocol succeeds as long as the number of correct steps on the tree exceeds the number of incorrect steps by t. This happens as long as the number of correct steps is at least $C/2(t + \log(1/\epsilon)) + t/2$. Since the expected number of correct steps is $2C/3(t + \log(1/\epsilon))$, we get that the bad event happens only when we deviate from the expected number by $C/6(t+\log(1/\epsilon)) - t/2 > (C/6 - 1/2)(t + \log(1/\epsilon))$. By the Chernoff bound, the probability that this happens is at most $\exp(\Omega((C/6 - 1/2)^2(t + \log(1/\epsilon))))$. Setting C to be a large enough constant makes this error at most ϵ.

$\square$