Week 8

Communication Complexity Basics

Communication complexity is an important model in complexity theory. It has surprising applications to many other parts of theoretical computer science, including circuit complexity, proof complexity, data structure lower bounds, and information theory.

8.1 Basic definitions

In query complexity, the main object of study was a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ (which can be generalized to be a partial function, or to have non-Boolean inputs and/or outputs, or to be a relation rather than a function). To compute $f(x)$, we were allowed to make queries to the bits of the input x, receiving the answer x_i when we query position $i \in [n]$ of $x \in \{0,1\}^n$.

In communication complexity, we study functions $F\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, or more generally $F\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ for some finite sets $\mathcal{X}$ and $\mathcal{Y}$. (Note that the function name F is usually capitalized; f is usually a query function, while F is usually a communication function.) An input to F will have the form (x, y) with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Unlike in query complexity, we are not allowed to make queries to the bits of x and y. Instead, we consider two players, Alice and Bob. When we are given an input (x, y), what actually happens is that Alice is given x and Bob is given y. They are allowed to do any computation of their half of the input for free; in other words, Alice can see all of x (without having to pay for queries to x) and Bob can see all of y. Their goal is to compute $F(x, y)$ using as little *communication* as possible.

Note that Alice and Bob are on the same team: they are cooperating together to try to get $F(x, y)$. Additionally, they can decide on a strategy in advance, before receiving their inputs. This means that Alice and Bob can be assumed to know the function F, and they can get together and decide on any strategy they wish for computing F; then they are separated, given an input (x, y), and must compute $F(x, y)$ using as little communication as possible. We will use $\mathrm{D}^{\mathrm{CC}}(F)$ to denote the worst-case number of bits Alice and Bob must send to each other in order to get the answer bit $F(x, y)$. This is called the deterministic communication complexity of F. We will also use $\mathrm{R}^{\mathrm{CC}}(F)$ and $\mathrm{Q}^{\mathrm{CC}}(F)$ to denote the randomized and quantum communication complexities of F; in the former, Alice and Bob have access to randomness, and in the latter, they may transmit qubits to each other instead of bits.

Note a few interesting things about communication complexity. First of all, it doesn't matter whether Alice or Bob holds the answer $F(x, y)$ at the end of the protocol, because either of them can transmit the answer to the other at the cost of just one additional bit of communication. Hence, up to a $+1$ term, it doesn't matter if we define F solved when Alice outputs $F(x, y)$, when Bob outputs $F(x, y)$, or if they both need to output $F(x, y)$ correctly.

Second, note that it doesn't matter what the set $\mathcal{X}$ looks like internally, whether it consists of strings in $\{0,1\}^n$ or of something else. That's because Alice can see all of her input x and can calculate any function of f for free; this means that if the input is given in the form $\{0,1,2,3,4,\dots,2^n\}$, Alice can convert this to an input of the form $\{0,1\}^n$ or any other form of her choice. For this reason, we can denote the input sets by $\mathcal{X}$ and $\mathcal{Y}$ without specifying them further.

8.1.1 Formalizing the definition

The definitions as we've stated them are a little vague. Let's try to formalize $\mathrm{D}^{\mathrm{CC}}(F)$. First, it will help to consider a third player besides Alice and Bob, which we will call the coordinator. Instead of Alice and Bob communicating with each other, they will instead communicate only with the coordinator. Also, we will make the coordinator output $F(x,y)$ at the end (rather than making Alice or Bob output the answer).

Further, consider the setting where the coordinator can ask Alice questions about x and ask Bob questions about y. Asking questions is free, but a 1-bit answer costs 1. In this model, Alice and Bob are completely passive, only answering questions about x and y asked by the coordinator.

Note that in the original model of communication complexity, Alice and Bob can agree in advance on the strategy. This means they can agree in advance to pretend there is a coordinator asking them questions, and they can also agree in advance on the coordinator's questioning strategy. This means that this new model, with the coordinator being active and Alice and Bob being passive, is equivalent to the original communication complexity model.

We can now formalize this coordinator model in order to formally define $\mathrm{D}^{\mathrm{CC}}(F)$. Note that each question asked by the coordinator is just some function $\alpha\colon \mathcal{X} \to \{0,1\}$ or $\beta\colon \mathcal{Y} \to \{0,1\}$; that is, the coordinator will send Alice α (for free), and will get back $\alpha(x)$. Or the coordinator may send Bob β (for free), and receive back $\beta(x)$.

Note that there are $2^{|\mathcal{X}|}$ functions $\alpha\colon \mathcal{X} \to \{0,1\}$. With this in mind, let ϕ_X be a mapping $\phi_X\colon \mathcal{X} \to \{0,1\}^{2^{|\mathcal{X}|}}$ which converts $x \in \mathcal{X}$ to a very long Boolean string. For each $\alpha\colon \mathcal{X} \to \{0,1\}$, we set $(\phi_X(x))_\alpha = \alpha(x)$. That is, the string $\phi_X(x)$ has one entry for each Boolean function defined on domain $\mathcal{X}$, and the entry of the string $\phi_X(x)$ at position α is simply the value of $\alpha(x)$. The single string $\phi_X(x)$ therefore stores the value of every possible function evaluated on x.

We now convert the communication function F into a query function f. The domain of f will be $\{\phi_X(x)\phi_Y(y) : (x,y) \in \mathcal{X} \times \mathcal{Y}\}$, and we will define $f(\phi_X(x)\phi_Y(y)) = F(x,y)$. In other words, f takes as input strings of length $2^{|\mathcal{X}|} + 2^{|\mathcal{Y}|}$, where the first $2^{\mathcal{X}}$ bits represent all possible function evaluations of x, and where the last $2^{\mathcal{Y}}$ bits represent all possible function evaluations of y. The output of f on such a string is $F(x,y)$.

We can now define $\mathrm{D}^{\mathrm{CC}}(F)$ to simply be $\mathrm{D}(f)$. Why is this a reasonable thing to do? That's because the coordinator, who asks Alice and Bob questions and receive 1-bit answers (at a cost of 1 per question), is effectively just querying the strings $\phi_X(x)$ and $\phi_Y(y)$. The coordinator may make such queries in any order, adaptively; but this is simply saying that the coordinator can run any deterministic query algorithm on $\phi_X(x)\phi_Y(y)$. The minimum number of worst-case queries the coordinator needs to make is therefore exactly $\mathrm{D}(f)$, and this corresponds to the minimum number of worst-case bits of communication Alice and Bob must send to each other before they both know $F(x,y)$.

8.1.2 Some properties of the query function coming from a communication function

Note that this function f is defined on very long strings. If $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}^m$, then $|\mathcal{X}| = 2^n$ and $|\mathcal{Y}| = 2^m$. This means that $|\operatorname{Dom}(f)| = |\mathcal{X} \times \mathcal{Y}| = 2^{n+m}$, and each string in the domain of f has length $|\phi_X(x)| + |\phi_Y(y)| = 2^{|\mathcal{X}|} + 2^{|\mathcal{Y}|} = 2^{2^n} + 2^{2^m}$. This actually means that f is defined on a very small promise set: the total number of strings of length $2^{2^n} + 2^{2^m}$ is $2^{2^{2^n}+2^{2^m}}$. Supposing without loss of generality that $n \geq m$, this is between $2^{2^{2^n}}$ and $4^{2^{2^n}}$, whereas $|\operatorname{Dom}(f)|$ is between 2^n and 4^n. If we use N to denote the number of bits in an input string of f, then $|\operatorname{Dom}(f)|$ is between $\Omega(\log\log N)$ and $O(\log\log^2 N)$. This is a really tiny subset of strings of length N.

Moreover, since Alice can always send her entire string over to Bob and have Bob compute $F(x,y)$, we know that $\mathrm{D}^{\mathrm{CC}}(F) = O(\min\{n,m\})$. Hence we know that $\mathrm{D}(f) = O(\log\log\log N)$. What we will care about is whether the communication complexity is less than n, say $\sqrt{n}$, which means we will care about whether $\mathrm{D}(f)$ is $\log\log\log N$ or $\sqrt{\log\log\log N}$. Reasoning about the query complexity of such functions, which are defined on such a small promise set and where we care about such small log factors, turns out to be quite tricky. It will be much better to reason about communication complexity directly, rather than trying to compute the query complexity of the corresponding query function.

8.2 Randomized communication models

Having formally defined deterministic communication complexity, we now turn our attention to randomized communication complexity. One way to define $\mathrm{R}^{\mathrm{CC}}(F)$ would be to set it equal to $\mathrm{R}(f)$, where f is the query function defined above.

What does this correspond to in terms of a communication strategy between Alice and Bob? It is clear that this $\mathrm{R}(f)$ definition corresponds to the coordinator using a randomized strategy when picking the questions to ask Alice and Bob. If Alice and Bob wish to mimic such a strategy themselves (without the coordinator), they would have to pretend to ask each other the questions – without actually sending the questions across, as that's too costly – and then send each other the answers to those questions. To do this, they must pick the questioning strategy in advance.

However, if the questions are *random* (with the randomness being chosen after the input), Alice and Bob cannot fully decide on the questioning strategy in advance, since that strategy also depends on random bits that must be chosen later (after Alice and Bob are separated and the input is given to them). In order to simulate the coordinator's behavior, Alice and Bob would need to somehow pick the *same* random bits together. Such a model is called the *shared randomness* or *public coin* model. We could pretend that Alice and Bob each have a device that generates random bits, but these devices are correlated and always give the same random bits to Alice and to Bob. The key is that the input (x,y) cannot depend on the value of this randomness; we can imagine (x,y) being chosen by an adversary that knows Alice and Bob's strategy, but doesn't know their random choice of bits. So in the shared randomness model, Alice and Bob must succeed on such worst-case inputs, but they get to pick shared random bits.

This shared randomness model turns out to be equivalent to $\mathrm{R}(f)$. This is the usual way of defining the randomized communication complexity $\mathrm{R}^{\mathrm{CC}}(F)$.

8.2.1 Private randomness

However, it is also interesting to consider the situation where Alice and Bob only have access to private randomness. This model is weaker: if Alice and Bob had shared randomness, then Bob can pick k bits of shared randomness by agreeing with Alice in advance that Alice will ignore the first k bits of randomness she receives from her shared-randomness device; then Bob can use these bits, and they will be uncorrelated with anything Alice uses. In other words, the shared randomness model can simulate the private randomness model, but not vice versa. Let us use $\mathrm{R}^{\mathrm{CC},\textsc{priv}}(F)$ to denote the randomized communication complexity of F with private randomness; then $\mathrm{R}^{\mathrm{CC},\textsc{priv}}(F) \geq \mathrm{R}^{\mathrm{CC}}(F)$. The measure $\mathrm{R}^{\mathrm{CC},\textsc{priv}}(F)$ cannot be defined using a query function such as f, so we must always talk about it in terms of communication models. We won't give a formal definition.

8.2.2 Equivalence

It turns out that in the bounded-error setting, the private randomness and shared randomness models are equivalent up to constant factors and up to an additive $O(\log\log|\mathcal{X}| + \log\log|\mathcal{Y}|)$ term (which is $O(\log n)$ when $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$). Formally, we have the following theorem.

Theorem 8.1. *Let $F\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a communication function. Then*

$$\mathrm{R}_\epsilon^{\mathrm{CC},\textsc{priv}}(F) = O\left(\mathrm{R}_\epsilon^{\mathrm{CC}}(F) + \log\log(|\mathcal{X}||\mathcal{Y}|) + \log\frac{1}{\epsilon} + \log\frac{1}{1-2\epsilon}\right).$$

This theorem may look complicated, but just realize that when ϵ is a constant in $(0,1/2)$, then the $\log 1/\epsilon$ and $\log 1/(1-2\epsilon)$ terms are both additive constants, and also that $\log\log(|\mathcal{X}||\mathcal{Y}|)$ should be interpreted as $\log n$ (where n is the size of the input strings). Hence this is essentially saying that $\mathrm{R}^{\mathrm{CC},\textsc{priv}}(F) = O(\mathrm{R}^{\mathrm{CC}}(F) + \log n)$.

Proof. Consider any communication protocol Π which uses shared randomness to solve F – equivalently, consider any randomized algorithm R which solves f to bounded error. Then R is a probability distribution over decision trees.

What we would like to do is to find k decision trees $D_1, D_2, \ldots, D_k$ in the support of R such that picking $i \in [k]$ at random and running D_i always gives nearly the same result as running R. (The decision trees D_i can have repetitions; they might even all be the same tree.) That is, we want the decision trees D_i to satisfy the property

$$|\Pr_{i\sim[k]}[D_i(z) = 1] - \Pr_{D\sim R}[D(z) = 1]| \leq \delta$$

for all $z \in \mathrm{Dom}(f)$, where δ is some small parameter. Note that if such a sequence of decision trees D_i exists, then Alice and Bob can solve F using private randomness, as follows: first, Alice picks a random i between 1 and k, and sends it to Bob. This requires $O(\log k)$ bits of communication. Next, Alice and Bob pretend there is a coordinator questioning them with questioning strategy D_i, and send each other the answer bits accordingly. Then they output $D_i(\phi_X(x)\phi_Y(y))$ where (x, y) is their input. Note that their success probability is within δ of the success probability of R, so their probability of error is at most $\epsilon + \delta$. This cost them an additive $O(\log k)$ more communication than in the shared randomness version.

It remains to show that for k not too large, such a sequence $D_1, D_2, \ldots, D_k$ of decision trees must exist. To do so, we consider picking each D_i independently at random from R, and calculate the probability that such a random sequence has the desired property. For each $z \in \mathrm{Dom}(f)$, the probability that

$$|\Pr_{i\sim[k]}[D_i(z) = 1] - \Pr_{D\sim R}[D(z) = 1]| \leq \delta$$

is just the probability that

$$|\frac{1}{k}\sum_{i=1}^{k} D_i(z) - \Pr[R(z)=1]| \le \delta.$$

Let X_i be the random variable $D_i(z)$, where D_i is chosen from R. Then the X_i variables are identically distributed Bernoulli random variables, with mean $\mu = \Pr[R(z)=1]$. Then the Chernoff-Hoeffding inequality tells us that the probability of the average of k i.i.d. Bernoulli random variables is more than δ away from the mean is at most $2e^{-2\delta^2 k}$, so the randomly-chosen sequence D_i is good on a single input z with probability at least $1-2e^{-2\delta^2 k}$. By the union bound, the probability that a randomly chosen sequence D_i is good for all $z \in \operatorname{Dom}(f)$ is at least $1-2e^{-2\delta^2 k}|\operatorname{Dom}(f)| = 1-2e^{-2\delta^2 k}|\mathcal{X}||\mathcal{Y}|$. To ensure that such a sequence exists, all we need to do is pick k large enough that this expression is greater than 0; equivalently, we need $e^{-2\delta^2 k} < 1/2|\mathcal{X}||\mathcal{Y}|$ or $k > \ln(2|\mathcal{X}||\mathcal{Y}|)/2\delta^2$. Picking $k = \lfloor \ln(2|\mathcal{X}||\mathcal{Y}|)/2\delta^2 \rfloor + 1$, we get that taking $k = O(\log(|\mathcal{X}||\mathcal{Y}|)/\delta^2)$ suffices, so Alice and Bob can achieve error $\epsilon+\delta$ with private randomness using an additive overhead of $O(\log\log(|\mathcal{X}||\mathcal{Y}|)+\log 1/\delta)$ extra bits of communication.

We can now combine this result with amplification. If ϵ is close to 0, we can take $\delta = \Theta(\epsilon)$ and then amplification from $\epsilon+\delta$ error to ϵ error costs only a constant factor. On the other hand, if ϵ is close to 1/2, we can take $\delta = \Theta(1-2\epsilon)$ and then amplification against costs only a constant factor. In both cases, we get

$$\mathrm{R}_\epsilon^{\mathrm{CC,PRIV}}(F) = O\left(\mathrm{R}_\epsilon^{\mathrm{CC}}(F) + \log\log(|\mathcal{X}||\mathcal{Y}|) + \log\frac{1}{\epsilon} + \log\frac{1}{1-2\epsilon}\right),$$

as desired. □

8.3 Quantum communication complexity

At first, one might be tempted to define quantum communication complexity in a way which is analogous to deterministic and (shared-randomness) randomized communication complexity, by setting it equal to $\mathrm{Q}(f)$ for the query function f. Unfortunately, this definition doesn't work. The reason is that $\mathrm{Q}(f) \le 2$ for all query functions f that come from a communication function F. Indeed, recall that if we have access to subset-parity queries for a string $x \in \{0,1\}^n$, then the Bernstein-vazirani algorithm lets us extract all of x using a single quantum query (a query to the parity of subsets of x, in superposition). The string $\phi_X(x)$ contains all possible functions applied to x, so in particular, some positions in the string $\phi_X(x)$ represent the parities of the subsets of x. Hence, using a single quantum query to $\phi_X(x)$, we can extract the entire string x. Doing a similar thing to y using another query, we can get both x and y, and then output $F(x,y)$ after only two queries.

The reason this definition failed is that such a definition essentially allows the coordinator to ask Alice and Bob questions in superposition; however, we did not charge for the questions, only for the answers. Setting up a really large superposition, over 2^n different questions, allows a single quantum query to extract n classical bits. Alice and Bob have no way to simulate this without a coordinator.

Instead, we must define quantum communication complexity differently, in a way which actually talks about the communication. This is a bit tedious to do, but we will go through it anyway.

What happens is that Alice holds a register $|A\rangle$ and Bob holds a register $|B\rangle$. At the beginning of the protocol, the registers are initialized to $|A_0\rangle = |x\rangle|0\rangle$ and $|B_0\rangle = |y\rangle|0\rangle$, where the extra $|0\rangle$ states are some work space of arbitrarily large (but finite) dimension. In addition to these register, there is also a communication register $|C\rangle$, which can only store values in $\{0,1\}$. Let's initialize $|C_0\rangle = |0\rangle$. This register $|C\rangle$ will move back and forth between Alice and Bob.

We will assume that Alice and Bob take turns speaking. This is a safe assumption up to a factor of 2; for example, if Alice and Bob wanted to arrange that only Alice speaks, we can just have Bob send back $|0\rangle$ each time, which uses twice as many bits of communication while gaining the property that Alice and Bob alternate their speech.

The communication protocol itself will be a sequence of unitaries $U_0, U_1, \ldots, U_T$. At first, Alice will apply U_0 to $|A_0\rangle |C_0\rangle$ (with U_0 acting as identity the register $|B_0\rangle$, so that Alice may apply U_0 on her side only). We define $|A_1\rangle |C_1\rangle |B_1\rangle = U_0 |A_0\rangle |C_0\rangle |B_0\rangle$ (note that $|B_1\rangle$ is the same as $|B_0\rangle$). Then the communication register $|C_1\rangle$ gets sent to Bob, and Bob applies the unitary U_1 to $|B_1\rangle |C_1\rangle$, and sends the C register back to Alice. They keep going in this way, with Alice applying even-numbered unitary matrices and Bob applying odd-numbered unitary matrices.

After the last unitary is applied, a special part of Alice and Bob's registers is measured. That is, we assume that Alice and Bob each hold special output registers as part of their work tapes, and that these special registers take values in $\{0,1\}$. For convenience, we will require that both Alice and Bob output the right answer: that is, we measure each of their output registers, and we say that the protocol succeeded on input (x,y) if both measurement outcomes were equal to $F(x,y)$.

Having defined a quantum communication protocol, we can now define $\mathrm{Q}^{\mathrm{CC}}(F)$ to be the minimum number of rounds T in a quantum communication protocol which computes $F(x,y)$ to error at most $1/3$ against worst-case inputs (x,y).

8.3.1 Shared entanglement

The definition we gave for $\mathrm{Q}^{\mathrm{CC}}(F)$ allows Alice and Bob to use private randomness (they can generate random numbers using quantum states, using some standard tricks), but it does not allow Alice and Bob to have shared randomness. We can extend the definition to allow them to share randomness as well, giving a potentially more powerful model.

Going further, we can consider allowing Alice and Bob to start with a shared entangled state. They will agree on this state in advance, and the shared state will span the work registers of both Alice and Bob; this starting state cannot depend on the input. It will replace the $|0\rangle$ part of $|A_0\rangle$ and $|B_0\rangle$ respectively, and the protocol will otherwise be defined to be the same as before.

Since the shared entanglement is arbitrary (and must be specified along with the unitary matrices $U_0, U_1, \ldots, U_T$ when defining a quantum communication protocol), Alice and Bob can choose it in a way that gives them access to shared randomness. Hence shared entanglement is more powerful than shared randomness.

Another cool property is that using appropriately-chosen shared entanglement, Alice and Bob only need to send each other classical bits. This is because they can use quantum teleportation, which uses an entangled state together with classical communication to transmit quantum bits from Alice to Bob.

Recall that for randomized communication complexity, the private and shared randomness settings ended up being equivalent (at least up to a $\log n$ term and up to constant factors and when we only care about constant error). On the other hand, for quantum communication complexity, it is still open whether the shared entanglement and the non-shared entanglement models are equivalent in this way. Some authors use $\mathrm{Q}^{\mathrm{CC}}(F)$ to denote the shared entanglement quantum communication complexity of F. Other authors use $\mathrm{Q}^{\mathrm{CC}}(F)$ to denote the non-shared entanglement quantum communication complexity, and they use $\mathrm{Q}^{\mathrm{CC},*}(F)$ to denote the shared entanglement version. Note that $\mathrm{Q}^{\mathrm{CC},*}(F) = O(\mathrm{Q}^{\mathrm{CC}}(F)$.

8.4 Communication matrices and partial functions

We sometimes represent a communication function $F\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ by the corresponding *communication matrix.* This is a matrix with rows indexed by $\mathcal{X}$ and columns indexed by $\mathcal{Y}$, in which the entry corresponding to (x,y) is $F(x,y)$. We denote this matrix by the same symbol F, so we will sometimes write $F[x,y]$ for $F(x,y)$ when we want to think of F as a matrix. Note that if $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$, then this matrix is $2^n \times 2^n$; in other words, the matrix is exponentially large in what we would normally think of as the input size.

Communication functions we have seen are analogous to total Boolean functions in query complexity. However, we can also define partial communication functions. This works by defining F on a *subset* of $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are again arbitrary finite sets. In other words, a partial communication function is a function $F\colon S \to \{0,1\}$ where $S \subseteq \mathcal{X} \times \mathcal{Y}$. We can also write $F\colon \mathcal{X} \times \mathcal{Y} \to \{0,1,*\}$ where we set $F(x,y) = *$ if (x,y) is not in the domain of F.

The definitions above for deterministic, randomized, and quantum communication complexities all generalize naturally to partial functions. We can also extend the notion of a communication matrix: if F is a partial communication function, then its communication matrix is again F, where now some entries $F[x,y]$ might be $*$ instead of 0 or 1.

8.4.1 The rank of a communication matrix

Surprisingly, the rank of a communication matrix turns out to be an important measure for a communication function. In fact, we have the following.

Lemma 8.2. *Let $F\colon \mathcal{X} \to \mathcal{Y} \to \{0,1\}$ be a total communication function. Then $\mathrm{D}^{\mathrm{CC}}(F) \geq \log_2 \operatorname{rank}(F)$.*

Proof. Take any deterministic communication protocol for F. This can be thought of as a deterministic query algorithm D for f, which makes queries to the string $\phi_X(x)\phi_Y(y)$.

If the algorithm makes T queries, then there are 2^T possible sequences of answers to those queries. For each such sequence $s \in \{0,1\}^T$, consider the set $A_s \subseteq \mathrm{Dom}(F)$ of all inputs (x,y) which are consistent with these query responses; that is, all inputs (x,y) such that if we run D on that input, the sequence of query responses would be exactly s. Note that the sets A_s are disjoint, and that their union is $\mathrm{Dom}(F)$, so they partition $\mathrm{Dom}(F)$.

Further, note that each set A_s is actually a *rectangle.* In communication complexity, we say a subset of $\mathcal{X} \times \mathcal{Y}$ is a rectangle if it is equal to $S \times T$ for some $S \subseteq \mathcal{X}$ and $T \subseteq \mathcal{Y}$. For each $s \in \{0,1\}^T$, the set A_s must be a rectangle, because each question to Alice about input x only eliminates possibilities from $\mathcal{X}$ and question to Bob about input y y only eliminates possibilities from $\mathcal{Y}$; there is no way to rule out a pair (x,y) from being the input without also ruling out all pairs of the form (x,y') and (x',y) for all x' and y'.

Finally, note that since D computes f, the matrix F restricted to the entries in A_s must either be all 0s or all 1s. This is because after the T queries, when the algorithm saw the responses $s \in \{0,1\}^T$, the algorithm knows with certainty the value of $F(x,y)$; hence all inputs (x,y) consistent with seeing s must have the same value under F. Let F_s be the matrix with the same dimensions as F which is equal to F on coordinates in A_s and equal to 0 elsewhere. Then $F = \sum_{s\in\{0,1\}^T} F_s$. Moreover, each F_s is either the all-0 matrix, or else it is a matrix with 1s in A_s and 0s elsewhere. Since A_s is a rectangle, by rearranging rows and columns of F_s we can get it to be a rectangle of 1s, surrounded by 0s. This is a rank-1 matrix.

Finally, we write

$$\operatorname{rank}(F) = \operatorname{rank}\left(\sum_{s\in\{0,1\}^T} F_s\right) \le \sum_{s\in\{0,1\}^T} \operatorname{rank}(F_s) \le \sum_{s\in\{0,1\}^T} 1 = 2^T.$$

Since $T = \mathrm{D}^{\mathrm{CC}}(F)$, the desired result follows. □

A long standing open problem in communication complexity is called the logrank conjecture. It says that the above relationship can also go the other way, up to polynomial factors. In other words, the conjecture states that logrank is always a decent lower bound on deterministic communication complexity.

Conjecture 8.3. *For every total communication function $F\colon \mathcal{X}\times\mathcal{Y}\to\{0,1\}$, we have $\mathrm{D}^{\mathrm{CC}}(F) = O(\operatorname{polylog}(\operatorname{rank}(F)))$.*

8.5 A separation between deterministic and randomized communication

We can give an exponential separation between randomized and deterministic communication complexities for a total Boolean function. In fact, we give a function $F\colon \{0,1\}^n\times\{0,1\}^n\to\{0,1\}$ for which $\mathrm{D}^{\mathrm{CC}}(F) = \Omega(n)$ but $\mathrm{R}^{\mathrm{CC}}(F) = O(1)$. This is the maximum possible separation, because the deterministic communication complexity is always at most $O(\min\{\log|\mathcal{X}|, \log|\mathcal{Y}|\})$ (since one of Alice or Bob can send their whole input to the other).

The function which achieves this separation is called equality, or EQ_n. This function is defined by $\mathrm{EQ}_n(x,y) = 1$ if $x = y$ and $\mathrm{EQ}_n(x,y) = 0$ if $x \neq y$. The domain of EQ_n is $\{0,1\}^n\times\{0,1\}^n$.

Let's first prove a lower bound on $\mathrm{D}^{\mathrm{CC}}(\mathrm{EQ}_n)$. To do so, we will use logrank. Note that the communication matrix of this function EQ_n is the $2^n\times 2^n$ identity matrix. Its rank is 2^n, and hence the logarithm of its rank is n. Hence we get $\mathrm{D}^{\mathrm{CC}}(\mathrm{EQ}_n) = \Omega(n)$.

Next, we can upper bound the randomized communication complexity $\mathrm{R}^{\mathrm{CC}}(F)$. We will do so using public randomness. Alice and Bob have n-bit strings x and y, and they wish to know if they are equal. What they will do is pick a random subset $S \subseteq [n]$ using their shared randomness, and then Alice will send Bob the parity of the bits of x that are in S. Bob will compare that to the parity of the bits of y that are in S. They will repeat this procedure a few times, say 10 times with newly chosen random subsets each time, and if the parities agree all 10 times, they output 1. Otherwise, if the parities ever disagree, they output 0.

Note that if $x = y$, then Alice and Bob will always output 1 when following this protocol. Now, suppose $x \neq y$. Then let $T \subseteq [n]$ be the non-empty set of positions at which x and y disagree. For any $S \subseteq [n]$, the parities of the bits of x in S and the bits of y in S disagree if and only if $|S\cap T|$ is odd. Since S is randomly chosen, each $i \in T$ occurs in S with probability 1/2. It follows that no matter what size T is, the probability that $|S\cap T|$ is odd when S is chosen randomly is exactly 1/2. Hence in each round, Alice and Bob have a 1/2 chance of discovering that $x \neq y$ (if indeed x and y are different). After 10 rounds, the probability that Alice and Bob did not discover that $x \neq y$ is $2^{-10} < 0.001$. Hence this protocol achieves bounded error, and $\mathrm{R}^{\mathrm{CC}}(\mathrm{EQ}_n) = O(1)$.

Note that this protocol used shared randomness. We can convert it to a protocol with private randomness as we've seen, but that will use $O(\log n)$ bits of communication instead of $O(1)$. Still, $O(\log n)$ is an exponential improvement over $\mathrm{D}^{\mathrm{CC}}(F) = \Omega(n)$.

8.6 Composing a query function with a communication function

Communication functions cannot be composed with each other; if we have $F\colon \mathcal{X}\times\mathcal{Y}\to\{0,1\}$ and $G\colon \mathcal{X}'\times\mathcal{Y}'\to\{0,1\}$, it is not clear what $F\circ G$ should even mean. However, an interesting operation is to take a Boolean function $f\colon\{0,1\}^n\to\{0,1\}$ and a communication function $G\colon\mathcal{X}\times\mathcal{Y}\to\{0,1\}$ and compose them together.

The way this works is as follows. The function $f\circ G$ will be a communication function $\mathcal{X}^n\times\mathcal{Y}^n\to\{0,1\}$. On input $((x_1,x_2,\dots,x_n),(y_1,y_2,\dots,y_n))$, the function $f\circ G$ will evaluate to $f(G(x_1,y_1),G(x_2,y_2),\dots,G(x_n,y_n))$. In other words, we take as input n different inputs to G, and we give Alice all the n Alice-inputs and give Bob all the n Bob-inputs. Now Alice holds n different inputs x_i and Bob holds n different strings y_i. Their new goal is to compute $f\circ G$, which means they want to compute $G(x_i,y_i)$ for each i, and then apply f to the resulting n-bit string of outputs from G. They just need to produce this final bit $f(G(x_1,y_1),G(x_2,y_2),\dots,G(x_n,y_n))$. We can also extend this definition to partial functions f and G in the usual way (the promise of $f\circ G$ will be that the inputs to all n copies of G satisfy the promise of G, and also that the n-bit output string from the n copies of G satisfies the promise of f).

One might wonder: is it true that $\mathrm{D}^{\mathrm{CC}}(f\circ G)=\mathrm{D}(f)\,\mathrm{D}^{\mathrm{CC}}(G)$ for all Boolean functions f and communication functions G? The upper bound certainly holds (at least ignoring constant factors): Alice and Bob can pick a deterministic query algorithm for f, and then execute that algorithm, and whenever that algorithm queries a bit i of the input, Alice and Bob will compute the i-th copy of G.

However, the lower bound does not hold. As an example, consider $f=\textsc{Parity}_n$ and $G=\mathrm{XOR}$, where $G\colon\{0,1\}\times\{0,1\}\to\{0,1\}$ is the function $G(x,y)=x\oplus y$ which takes two bits and returns their XOR. It should be clear that $\mathrm{D}^{\mathrm{CC}}(G)=1$ and $\mathrm{D}(f)=n$. However, the function $f\circ G$ is just the function $\{0,1\}^n\times\{0,1\}^n\to\{0,1\}$ which takes two strings (x,y) and returns the parity of the concatenated string xy. Alice and Bob can compute this function using $O(1)$ communication: Alice will simply send Bob the parity of x, and Bob will compute the parity of y and XOR with the bit from Alice. This will give the right answer for the joint parity of their strings.

On the other hand, what does seem to be true is that for *sufficiently hard* functions G, it may be the case that $\mathrm{D}^{\mathrm{CC}}(f\circ G)=\Omega(\mathrm{D}(f)\,\mathrm{D}^{\mathrm{CC}}(G))$. So while we cannot hope that this statement will always be true, we can still hope that it is true for all "non-trivial" G, for an appropriate definition of non-trivial.

However, this is still open. What's known is a weaker result, called a *lifting theorem.*

Theorem 8.4. *For every input size n, there exists a communication function G_n such that $\mathrm{D}^{\mathrm{CC}}(G_n)=O(\log n)$ and such that for any Boolean function $f\colon\{0,1\}^n\to\{0,1\}$, we have $\mathrm{D}^{\mathrm{CC}}(f\circ G_n)=\Omega(\mathrm{D}(f)\,\mathrm{D}^{\mathrm{CC}}(G))$.*

This theorem, first shown by [GPW15] based on a protocol of [RM99], says that although we don't know how to compose every f with every G, we do know that some sufficiently "rich" G does compose with every f. This "lifting theorem" allows us to take a query function f and "lift" it to a communication function $f\circ G$ which shares many of the properties of f. For starters, it will have similar deterministic communication complexity to the query complexity of f. Moreover, since $f\circ G$ composes in the upper bound direction in most models (e.g. randomized and quantum, though we may lose a log factor due to amplification), we also have an upper bound on $\mathrm{R}^{\mathrm{CC}}(f\circ G)$ in terms of $\mathrm{R}(f)$ and an upper on $\mathrm{Q}^{\mathrm{CC}}(f\circ G)$ in terms of $\mathrm{Q}(f)$.

It's important that the communication complexity of G_n is small (only $O(\log n)$), because this way, when we compose a query function f with G, the communication function we get has almost the same communication complexity as the query complexity of f. Ideally, one could also prove the above theorem for a *constant* communication function G, whose size does not depend on n (the input size of f). This is believed to be true, but is currently open.

Conjecture 8.5. *There is a fixed communication function G such that for all query functions f, $\mathrm{D}^{\mathrm{CC}}(f \circ G) = \Omega(\mathrm{D}(f))$.*

Note that we phrased this "lifting with a constant-sized gadget" conjecture for deterministic query and communication complexities, but we can make a similar conjecture for other models, such as randomized (query and communication) complexity and quantum (query and communication) complexity. All these conjectures are currently open, at least for a constant-sized gadget. (We often use the term "gadget" to refer to the inner function G, because we think of it as small but rich/complex.)

Lifting a query function into communication complexity is a good way of constructing interesting communication functions. For example, the best separation we have between $\mathrm{R}^{\mathrm{CC}}(F)$ and $\mathrm{Q}^{\mathrm{CC}}(F)$ for a total communication function comes from taking a total query function with a separation between $\mathrm{R}(f)$ and $\mathrm{Q}(f)$, and lifting it. This requires a lifting theorem for *randomized* query-to-communication complexity, which was only shown recently [GPW17] (the gadget size there is not constant, but has communication complexity that grows like $O(\log n)$ where n is the input size of f). Note that it does not require a quantum lifting theorem, since such a separation only requires an upper bound on the quantum communication of a composed function, which is the easy direction (lifting theorems prove the lower bound).

A lifting theorem for quantum query to quantum communication complexity is not currently known with any gadget. This is an important open problem.

Conjecture 8.6. *For each n, there is a communication function G_n such that $\mathrm{Q}^{\mathrm{CC}}(G_n) = O(\mathrm{polylog}\, n)$ and such that for any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we have $\mathrm{Q}^{\mathrm{CC}}(f \circ G_n) = \tilde{\Omega}(\mathrm{Q}(f))$.*

Another important open problem is whether there is a super-polynomial separation between randomized and quantum communication complexities for total functions. Recall that in query complexity, we know that $\mathrm{R}(f) = O(\mathrm{Q}(f)^6)$ for all total functions f (and this has recently improved to a power 4 relation). In query complexity, this even holds for $\mathrm{D}(f)$ vs. $\mathrm{Q}(f)$ – they are also polynomially related for total functions.

However, in communication complexity, we saw an exponential separation between $\mathrm{D}^{\mathrm{CC}}(\mathrm{EQ})$ and $\mathrm{R}^{\mathrm{CC}}(\mathrm{EQ})$ already. Note that EQ is a total function. On the other hand, such a separation between R^{CC} and Q^{CC} is not known, and is an important open problem.

Conjecture 8.7. *We have $\mathrm{R}^{\mathrm{CC}}(F) = O(\mathrm{poly}(\mathrm{Q}^{\mathrm{CC}}(F)))$ for all total communication functions F.*

Remember that lifting query functions to communication functions can give us *separations* between measures, but it cannot give us relations between communication complexity measures, because although every query function can be lifted to a communication function, not every communication function can be viewed as a lifted query function.

As a final note, for *partial* communication functions, exponential separations are known; one way to get them would be to start with a partial query function which gives an exponential separation between R and Q, and then "lift" it to a communication function by composing it with a communication function G. Lifting theorems generally work for partial functions too, not just for total functions.

8.7 Other communication measures

We've defined a whole bunch of query complexity measures: approximate degree, exact degree, certificate complexity, block sensitivity, fractional block sensitivity, the positive and negative adversary methods, etc. Several of these measures have communication complexity analogues, though others do not.

If we have a communication complexity measure $M^{\mathrm{CC}}(F)$, how do we know whether it corresponds to a query measure $M(f)$? One way would just be to argue that the models are similar, but a better (more formal) way is to show that there is a lifting theorem. Ideally, we would want a lifting theorem in both the upper and lower bound directions. Usually it is easy to show $M^{\mathrm{CC}}(f \circ G) = \tilde{O}(M(f)M^{\mathrm{CC}}(G))$ (at least if the query measure M and the communication measure M^{CC} are really analogous to each other), and so usually the missing piece is a lifting theorem in the lower bound direction, showing that at least for some G, $M^{\mathrm{CC}}(f \circ G) = \tilde{\Omega}(M(f)M^{\mathrm{CC}}(G))$. If both the upper bound and the lower bound statements hold, we say that M^{CC} is a communication version of the query measure M.

This is particularly useful if we want to show lower bounds on communication complexity. Showing lower bounds in query complexity on measures like $\mathrm{Q}(f)$ is not easy to do directly – it usually requires a technique such as the polynomial method or the adversary method. In communication complexity, everything tends to be harder. Lower bounding $\mathrm{Q}^{\mathrm{CC}}(F)$ for a function F is often quite difficult. Even proving $\mathrm{R}^{\mathrm{CC}}(F)$ or $\mathrm{D}^{\mathrm{CC}}(F)$ lower bounds can many times be tricky. What we would like to do is find communication complexity analogues of query complexity measures like $\widetilde{\deg}(f)$ and $\mathrm{Adv}(f)$, as they would be helpful in proving communication complexity lower bounds.

It turns out that the polynomial measures ($\widetilde{\deg}(f)$, $\deg(f)$, $\deg_{\mathrm{PP}}(f)$, and $\deg_{\pm}(f)$) have reasonably good communication complexity analogues, as does certificate complexity $\mathrm{C}(f)$. On the other hand, sensitivity, block sensitivity, fractional block sensitivity, and the adversary methods do not seem to have communication complexity analogues.

8.7.1 Parallels between query complexity and communication complexity

In order to make sense of which communication complexity measures are analogues of things like $\deg(f)$ or $\widetilde{\deg}(f)$, it will help to first draw some basic parallels between query concepts and communication concepts.

Partial assignments correspond to rectangles. Recall that *partial assignments*, which represent partial knowledge of a Boolean string in $\{0,1\}^n$ and are often denoted by a string in $\{0,1,*\}^n$, were a basic concept in query complexity. We needed partial assignments to define things like certificates and certificate complexity. A partial assignment also represents a *subcube* of the Boolean hypercube: if we know that the input x is consistent with a certain partial assignment, say $p = 00{*}1{*}01{*}$, then we know that x lies in the 3-dimensional subcube of the Boolean hypercube which consists of all 8 strings consistent with p. Note that if we had a total Boolean function f defined on strings like x, then the restriction of f to strings consistent with p yields a total Boolean function on $\{0,1\}^n$. That is to say, restricting total functions to partial assignments preserves their “total” property.

In summary, partial assignments give us partial knowledge about the input string, as we might know in the middle of a (classical) query algorithm. The communication analogue of partial assignments is *rectangles*: subsets $S \subseteq \mathcal{X} \times \mathcal{Y}$ which can be written $S = A \times B$ for $A \subseteq \mathcal{X}$ and $B \subseteq \mathcal{Y}$. A rectangle represents partial knowledge of the input (x, y), as a coordinator running a deterministic algorithm might acquire. Restricting a total communication function F to a rectangle gives another total communication function.

Instead of measuring size, take the log of the count. In query complexity, we often define measures in terms of the size of partial assignments: the number of non-$*$ bits in the partial assignment. This is because in order for a deterministic algorithm to know that the input is consistent with a large partial assignment, it must make a large number of queries. Note that a large partial assignment corresponds to a small hypercube: the more bits are revealed in the partial assignment, the fewer $*$ bits remain, and so the smaller the subcube defined by the partial assignment. A deterministic algorithm has a hard time reaching the conclusion that the input is in a small subcube, so the size of a partial assignment is a often a meaningful way to define its cost.

However, in communication complexity, the size of rectangles is not meaningful. Even for small rectangles (which might sound like they are analogous to small subcubes, i.e. large partial assignments), a deterministic coordinator can check if the input (x, y) is in this rectangle just by asking Alice and Bob one question each. Indeed, from the coordinator's perspective, all rectangles are pretty much equally difficult. We cannot use the size of a rectangle as a measure of its cost.

Instead, in communication complexity, we measure the *number* of rectangles (and take the logarithm of the result). As we've seen, a deterministic communication protocol of length T splits up $\mathcal{X} \times \mathcal{Y}$ into at most 2^T rectangles, with one rectangle corresponding to each sequence of T answer bits that the coordinator could have received for the questions sent to Alice and Bob. Hence taking the logarithm of the number of rectangles often gives a useful measure. More generally, instead of measuring the *size* of things (such as the size of the largest monomial for polynomial degree), in communication we measure the *number* of things, and take a logarithm (so we might take the log of the number of monomials instead).

8.7.2 Logrank corresponds to degree

The measure $\log_2 \operatorname{rank}(F)$, which we've already seen is a lower bound for $\mathrm{D}^{\mathrm{CC}}(F)$, turns out to be analogous to exact degree $\deg(f)$ in query complexity.

The reason for the analogy is as follows. A matrix has rank at most k if and only if it can be written as a sum of at most k rank-1 matrices. This is also if and only if it can be written as a linear combination of k rank-1 matrices, and we can assume the latter are normalized so that their average entry is 1. Now, such normalized rank-1 matrices aren't quite the same as all-1 rectangles, but we'll ignore the difference for now and say that a matrix has rank at most k more-or-less if it can be written as a linear combination of at most k all-1 rectangles, where by an all-1 rectangle we mean a matrix which has 1s inside a certain rectangle and 0s outside of it.

We now note that rectangles roughly correspond to partial assignments in query complexity; an all-1 rectangle, interpreted as a communication matrix, therefore corresponds to a query function which evaluates to 1 on inputs that agree with a certain partial assignment p, and which evaluates to 0 for inputs that are not consistent with p. This function is actually a monomial, sort of. For example, if $p = 011***0$, then the function which evaluates to 1 on inputs consistent with p and evaluates to 0 on inputs not consistent with p can be written $\overline{x_1}x_2x_3\overline{x_7}$, where $\overline{x_i} = 1 - x_i$. This is a monomial in the variables x_i and the negated variables $\overline{x_i}$. Hence, very roughly speaking, all-1 rectangles in communication complexity correspond to monomials in query complexity.

The rank of a communication matrix is therefore roughly analogous to the minimum number k such that f can be written as a linear combination of at most k monomials. Of course, representing f in terms of a linear combination of monomials is the same as representing f as a polynomial, which can be done in only one way (if we want an exact representation by a multilinear polynomial). Hence the rank of a matrix is similar to degree, except it asks about the number of monomials in the representation of f instead of about the largest degree of a monomial.

Note that the number of monomials in a representation of f is not a very robust property: it can change when we switch from the $\{0,1\}$ basis to the $\{+1,-1\}$ basis or vice versa. For example, the polynomial for AND in the $\{0,1\}$ basis is $x_1x_2x_3\dots x_n$, but if we switch bases to $\{+1,-1\}$, we get the polynomial

$$\left(\frac{x_1+1}{2}\right)\left(\frac{x_2+1}{2}\right)\dots\left(\frac{x_n+1}{2}\right),$$

which expands to have 2^n monomials. However, changing bases in this way will never increase the number of monomials by more than a factor of 2^d, where d is the degree of the polynomial. It turns out that taking the logarithm of the number of monomials, maximized between the $\{0,1\}$ basis and the $\{+1,-1\}$ basis, will always give a number between $\Omega(d)$ and $O(d\log n)$, so the log of the number of monomials really is roughly the degree – as long as we try different bases to not get stuck in a basis that happens to have very few, high degree monomials.

It turns out that degree also lifts to logrank, in both the upper bound and lower bound directions, up to a $O(\log n)$ factor. This lifting even works with a constant-sized gadget G.

8.7.3 Certificate complexity

The communication complexity analogue of one-sided certificate complexity $\mathrm{C}_1(f)$ is the logarithm of the minimum number of all-1 rectangles needed to *cover* the 1s of the communication matrix F. That is, it is the logarithm of the minimum number of rectangles $A_j \times B_j \subseteq \mathcal{X} \times \mathcal{Y}$ such that for all $(x,y) \in \mathrm{Dom}(F)$, if $F[x,y]=1$ then $(x,y) \in A_j \times B_j$ for some j, and if $F[x,y]=0$, then $(x,y) \notin A_j \times B_j$ for all j. Note that if Alice and Bob hold a 1-input (x,y), then someone can prove to them that they have a 1-input by telling them j, the index of the rectangle $A_j \times B_j$ containing their 1-input (x,y). Then Alice and Bob can separately check that $x \in A_j$ and $y \in B_j$. Communicating j to them costs log of the number of rectangles used to cover all the 1s.

We can also define C_0 in communication complexity analogously, and then set C to the maximum of C_0 and C_1. The result $\mathrm{D}(f) \le \mathrm{C}_0(f)\,\mathrm{C}_1(f)$, which holds for total functions in query complexity, turns out to also hold for total functions in communication complexity.

References

[GPW15] Mika Göös, Toniann Pitassi, and Thomas Watson. "Deterministic Communication vs. Partition Number". In: *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2015. DOI: 10.1109/FOCS.2015.70. ECCC: 2015/050 (p. 9).

[GPW17] Mika Göös, Toniann Pitassi, and Thomas Watson. "Query-to-Communication Lifting for BPP". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 2017 (p. 10).

[RM99] Ran Raz and Pierre McKenzie. "Separation of the Monotone NC Hierarchy". In: *Combinatorica* 19.3 (1999). DOI: 10.1007/s004930050062 (p. 9).