CS Theory (Spring '25) Fall, 2025

Lecture Notes: Communication Complexity, cont'd

Instructor: *Toniann Pitassi*

Last class we defined general two-way communication protocols which generalize the one-way communication model. We also saw an example, the INDEX function, which requires $\Omega(n)$ one-way communication but has a must lower two-way communication protocol of cost $O(\log n)$. Thus two-way communication protocols can sometimes be significantly shorter than one-way protocols. In this lecture we will prove some lower bounds for the following languages.

Definition 1. *We define the following communication problems:*

1. *Let* $\mathrm{EQ}(x,y) = \begin{cases} 1 & \text{if } x = y \text{ (as integers)} \\ 0 & \text{otherwise} \end{cases}$

2. *Let* $\mathrm{GEQ}(x,y) = \begin{cases} 1 & \text{if } x \text{ is greater than or equal to } y \text{ (as integers)} \\ 0 & \text{otherwise} \end{cases}$

1 Some formal definitions

Before proving the above, we will give a slightly more formal definition of a two-party protocol. This definition makes it easier to do lower bound (but harder to explain lower bounds).

Definition 2. *We have two players, Alice and Bob and some function* $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$. *On input* $w = w^A w^B$, *players take turns, starting with Alice, sending one bit per turn. The last bit sent must be* $f(w^A, w^B)$.

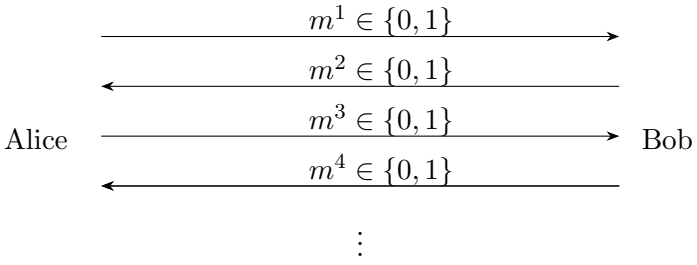

Figure 1: An example of communication between Alice and Bob.

More formally we have:
Alice sends message $m_1 = f^A(w^A) \in \{0,1\}$ *to Bob* *round 1*
Bob sends message $m_2 = f^B(m_1, w^B) \in \{0,1\}$ *to Alice*[1] *round 2*

⋮

[1]The message Bob sends depends on the message Alice sent him at first

Alice sends message $m_{2i+1} = f^A(m_1, \ldots, m_{2i}, w^A)$ *to Bob* *round* $2i+1$

Bob sends message $m_{2i+2} = f^B(m_1, \ldots, m_{2i+1}, w^B)$ *to Alice* *round* $2i+2$

The communication complexity of a protocol Π, $cc^{\Pi}(n)$=*# total of bits sent over all* $w = w^A w^B$, $|w| \leq n$.

Looking back at the Index example from the previous notes (where Bob sends an index i in binary to Alice), the protocol now would look like this:

- Alice sends 0.
- Bob the sends the first bit of the index i.
- Alice sends 0
- Bob the sends the second bit of i.
- Alice replies 0
- etc..
- Bob sends the last bit of i.
- Alice sends x_i to Bob.

This is more complicated, but does the same thing: We just need Alice to send "dummy messages" between each bit Bob sends. This is why when we give protocols, we just let Alice and Bob send whatever message they want. Note that the above also uses $O(\log(n))$ bits of communication. In particular, this more formal view where Alice and Bob each send one bit at a time only increases the communication by a constant factor (so it's the same in O-notation).

2 Low Cost Communication Implies the Matrix is Simple

Before proving lower bounds in the two-way communication setting, it is helpful to understand how a general communication protocol for a language L relates to the underlying matrix M_L.

Let L be any language, and consider inputs $w = w^A w^B$ where w has length $2n$. As discussed in previous lectures, we can express the communication problem for L on inputs of length $2n$ by a 2^n-by-2^n matrix, M_L. Let $X = \{0,1\}^n$ be the set of all possible inputs w^A for Alice, and similarly let $Y = \{0,1\}^n$ be all possible inputs w^B for Bob. The rows of M_L are labelled by elements of X and the columns are labelled by elements of Y, and entry (x, y) of M_L is 1 if $w = xy \in L$ and 0 otherwise.

Assume that we have a protocol of cost k for M_L (over inputs $w = w^A w^B$ of length $2n$). We now want to understand how the protocol partitions the matrix M_L into disjoint *rectangles*.

Definition 3. *Given a matrix M with rows X and columns Y: A rectangle R is a set $X' \times Y'$ where $X' \subseteq X$ and $Y' \subseteq Y$.* [2]

Definition 4. *Given a matrix M with rows X and columns Y. A rectangle $X' \times Y'$ is monochromatic if $M(x, y)$ has the same value for all $(x, y) \in X' \times Y'$.*

[2] A rectangle is obtained by picking a subset of the rows X' and the columns Y', then picking all entries at the intersection of these.

Claim 5. *Let* Π *be a cost* k *protocol for* L *over inputs* $w = w^A w^B$*, where* $|w| = 2n$*. Then* Π *partitions* M_L *into* 2^k *disjoint monochromatic rectangles.*

To prove the claim, consider the first round where Alice sends a one-bit message $m(w^A)$ to Bob. This partitions her inputs X into two subsets, X^0 and X^1 where X^0 are the inputs $w^A \in X$ where Alice's first message is 0, and X^1 are the inputs w^A where Alice's first message is 1. The figure below (Figure 4) illustrates an example where she sends 0 on ouputs 00,001,010 and on the other inputs she sends a 1, where the blue horizontal line indicates the partition, with one subset above and the other subset below the line. Thus her first 1-bit message partitions the entire matrix into two *rectangles*, $X^0 \times Y$, and $X^1 \times Y$.

Now in the second round, Bob sends a bit which depends on his input as well as Alices message. This further divides the matrix into four subrectangles, illustrated by Figure 4. In the third round, Alice sends a bit depending on her input as well as the two messages so far, which divides the matrix into 8 subrectangles. Then in the 4th round, Bob sends the final bit which divides the matrix into 16 subrectangles, and so on. Assuming the protocol has cost k, it follows that this partitions M_L into 2^k disjoint rectangles, where each rectangle corresponds to a distinct *transcript* [3] where the last bit of the transcript is the output. Now if the protocol is correct, for each of the 2^k subrectangles, all cells in that subrectangle must be either all labelled by "1" or all labelled by "0". Thus, at the end of the protocol M_L has been partitioned into 2^k disjoint monochromatic rectangles, so we have proven the claim.

As an example, Figure 2 below shows the matrix M_{PARITY} for PARITY on all inputs $w = w^A w^B$ of length $2n$, where $n = 3$.

	000	001	010	011	100	101	110	111
000	0	1	0	1	0	1	0	1
001	1	0	1	0	1	0	1	0
010	0	1	0	1	0	1	0	1
011	1	0	1	0	1	0	1	0
100	0	1	0	1	0	1	0	1
101	1	0	1	0	1	0	1	0
110	0	1	0	1	0	1	0	1
111	1	0	1	0	1	0	1	0

Figure 2: Communication Matrix for PARITY

In the standard protocol, Alice's first message is 1 if the number of 1's in w^A is odd, and 0 otherwise. If we rearrange the rows/columns of the matrix, we get the picture in Figure 3. In the second and final round, Bob computes the number of 1's in w^B plus the value m_1 sent by Alice, and outputs 1 if this sum is odd, and 0 if the sum is even. Thus his message m_2 is the output of the protocol. Thus, this protocol partitions M_{PARITY} into 4 disjoint monochromatic rectangles.

[3] A transcript is the ordered list of bits sent by Alice and Bob during the execution of the protocol.

	000	011	101	110	001	010	100	111
000	0	0	0	0	1	1	1	1
011	0	0	0	0	1	1	1	1
101	0	0	0	0	1	1	1	1
110	0	0	0	0	1	1	1	1
001	1	1	1	1	0	0	0	0
010	1	1	1	1	0	0	0	0
100	1	1	1	1	0	0	0	0
111	1	1	1	1	0	0	0	0

Figure 3: Communication Matrix for PARITY reordered

3 A two-party lower bound for EQ

As we saw, any one way protocol for EQ requires $\Omega(n)$ communication. Alice must send her whole input to Bob. But is that also needed for two-party protocols ? It turns out that the answer is yes!

Theorem 6. *Any communication protocol for EQ on inputs of length* $2n$ *requires* $\Omega(n)$ *communication.*

Proof. Our proof will follow the same general strategy as our lower bound for one-way protocols. First we recall the communication matrix associated with EQ. Below (in Figure 3) is a figure of the EQ matrix on inputs $w = w^A w^B$ where w has length 6. The rows are labeled with all $x \in X$, where X is the set of all n strings that Alice could get as input, and similarly the columns are labeled with all $y \in Y$. In our running example, X consists of all strings w^A of length 3, and Y consists of all strings w^B of length 3. Thus there are $|X| = 2^3 = 8$ rows and $|Y| = 2^3$ columns, where the rows/columns correspond to all possible inputs for Alice/Bob respectively, and an entry (w^A, w^B) is labelled with the value $\text{EQ}(w^A, w^B)$. Crucially the matrix is the identity matrix, which we will see has maximal communication complexity even for general protocols.

Now we can use a very similar pigeonhole principle argument to prove that EQ cannot be solved by any protocol where less than n bits are sent. As we saw in the example, if s bits are sent altogether, this partitions the EQ matrix into at most 2^s monochromatic subrectangles. Now consider the 2^n diagonal entries, which correspond to the inputs where EQ is 1. If $2^s < 2^n$, then by the pigeonhole principle there must exist a subrectangle that contains at least 2 different diagonal inputs. That is, there must be a transcript (corresponding to a subrectangle) and two yes-inputs, $w = uu$ and $w' = vv$ that land in the same subrectangle. Now since it is a subrectangle, the inputs uv and vu must also lie in this same subrectangle. Now we reach a contradiction: since every subrectangle is monochromatic, the protocol will either output "1" on all 4 of these inputs or it will output "0" on all four of these inputs. But the inputs uu and vv should be accepted while the inputs uv and vu should be rejected.

■

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

Figure 4: Communication Matrix for EQUALITY

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

Figure 5: Round 1 of EQUALITY Protocol

Theorem 7. *Any communication protocol for GEQ requires $\Omega(n)$ communication complexity.*

4 A Communication Lower Bound for GEQ

Proof. GEQ on input (u, v) should output 1 if and only if $u \leq v$, where we are viewing u and v as binary representations of nonnegative integers. Note that the communication matrix for GEQ is similar to the matrix for EQ. Recall that the matrix for EQ is the identity matrix: all entries are 0 except for those on the diagonal which are 1. For GEQ, it is still a lower triangular matrix (meaning that all entries above the diagonal are 0), but now all other entries are 1.

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

Figure 6: Round 2 of EQUALITY Protocol

We will see that the same argument that we saw last class to prove a lower bound for EQ works here as well. Assume for sake of contradiction that the communication complexity of GEQ on inputs (u, v) where $|u| = |v| = n$ is at most s, where $2^s < 2^n$. The protocol partitions the GEQ matrix into 2^s monochromatic subrectangles. Now consider the 2^n diagonal entries, which correspond to some inputs where GEQ is 1. Since $2^s < 2^n$, by the pigeonhole principle there must exist a subrectangle that contains at least 2 different diagonal inputs. That is, there must be a transcript (corresponding to a subrectangle) and two diagonal inputs, $w = uu$ and $w' = vv$ that land in the same subrectangle. Now since it is a subrectangle, the inputs uv and vu must also lie in this same subrectangle and therefore the protocol must give the same answer on all four inputs (uu, vv, uv, and vu). Assume without loss of generality that $u < v$. Then the inputs uu, vv, and vu should all be accepted, but on the other hand the input uv should be rejected. Therefore we reach a contradiction since the protocol must make a mistake on one of these inputs. ■

Final Remarks. This wraps up our whirlwind study of lower bounds for showing that specific languages are not regular, via techniques from communication complexity. To conclude, we want to mention that the property of being a regular language is extremely rare! There is *counting argument* showing that almost all languages $L \subseteq \{0,1\}^*$ are *not* regular. The counting argument at a high level is based on the observation that any regular language has a finite-length description by a DFA. Therefore the class of regular languages is *countable* – they can be ordered, and enumerated. On the other hand, the class of *all* languages if not countable. This counting argument shows that nearly all languages are not regular, but it is a nonconstructive argument in that we can't extract from the argument any *particular* language that is not regular. In contrast the lower bounds presented earlier are all constructive: we gave explicit natural languages and proved that they are not regular (which requires substantially more work in general). We will study countable versus uncountable later in the course when we discuss Turing Machines and computable languages. Like regular languages, the class of computable languages is very rare, again by a very similar counting argument.

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

Figure 7: Round 3 of EQUALITY Protocol

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

Figure 8: Last round of EQUALITY Protocol