CS Theory (Spring '25) Fall, 2025

Lecture Notes: Communication Complexity

Instructor: *Toniann Pitassi*

0 Some one-sided lower bound.

Let MAJ $= \{w \in \{0,1\}^* \mid w$ has at least as many 1s as 0s$\}$. For instance $w = 1010 \in$ MAJ but $w = 11000 \notin$ MAJ.

We want to show the following theorem:

Theorem 1. MAJ *isn't a regular language.*

To prove this, we can either use a direct argument based on DFAs, or we can show that no one-way communication protocol for MAJ uses $O(1)$ communication. The idea is that given $w = w^A w^B \in \{0,1\}^*$, Alice gets w^A and Bob gets w^B. If Alice doesn't send Bob the exact number of 1's in w^A, Bob can't be certain $w \in$ MAJ.

Hence, we will show the following:

Lemma 2. *Any one-way communication protocol for* MAJ *on inputs of length* $2n$ *requires* $\Omega(\log(n))$ *communication.*

Before diving into the proof, recall that if Alice sends only messages of length $\leq \ell$, then she can send a message of length $0, 1, 2 \ldots, \ell$. And for each length i, there's 2^i many binary strings of length i. So the number of messages is at most:

$$\sum_{i=1}^{\ell} 2^i = 2^{\ell+1} - 1.$$

Proof. Assume for contradiction that there exists a protocol where on inputs $w = w^A w^B$ of length $2n$, Alice only sends messages to Bob of length $\ell \leq \log_2(n) - 1$. [1]

Observe that the total number of different messages Alice can send is at most:

$$\sum_{i=0}^{\ell} 2^i \leq 2^{\ell+1} - 1 \leq 2^{\log_2(n)} - 1 < n.$$

Here the input w has length $2n$ so Alice's w^A has length n. Consider the following set X:

$$X = \{x \mid x = 1^k 0^{n-k}, k \geq 0\}.$$

[1] Here we use $\log_2(n) - 1$ since with this many bits, you can't encode n different messages. There's many other choices that would work.

Observe that every string in X has length n, hence each $x \in X$ is a possible input for Alice (we could have $w^A = x$). But every string in X has a different number of 1.

Why are we doing this ? Intuitively, if Alice doesn't send a unique message for each of $x \in X$, Bob wouldn't be able to know how many 1's Alice's string had. So something must go wrong.

Note that $|X| = n+1$ but as we already saw, there are only $< n$ many different messages Alice can send. Hence, by the pigeonhole principle, there exists $x, y \in X$ ($x \neq y$) such that $m(x) = m(y) = \alpha$. That is, there are two different inputs x, y for Alice such that Alice sends the same message on both x and y.

Since $x \neq y$ and they come from the set X, we can write:

$$x = 1^a 0^{n-a} \quad \text{and} \quad y = 1^b 0^{n-b} \quad \text{with } a \neq b.$$

Without loss of generality, we can assume $a < b$.[2]

Now consider the string $z = 1^{n-b}0^b$ which has length n.

If the input is $w = xz$, then Alice gets x and Bob gets z. We have $xz = 1^a 0^{n-a} 1^{n-b} 0^b$. This string has $t = n + a - b$ many 1's. Since $a < b$, we have $t < n$ so $xz \notin$ MAJ. So Bob must output 0 meaning:

$$0 = g(m(x), z) = g(\alpha, z). \tag{1}$$

However, if the input is $w = yz$, then Alice gets y and Bob gets z. We have $yz = 1^b 0^{n-b} 1^{n-b} 0^b$. This string has $t' = n + b - b = n$ many 1's. So $yz \in$ MAJ. So Bob must output 1 meaning:

$$1 = g(m(y), z) = g(\alpha, 1). \tag{2}$$

Now we have that Equations (5) and (6) give us a contradiction. Hence, there can't be a protocol where on inputs on length n, Alice sends a message of length $< \log_2(n) - 1$. Hence, any protocol for MAJ must use at least $\log_2(n) = \Omega(\log(n))$ communication on inputs of length $2n$. ■

Now, we know that no communication protocol for MAJ has $O(1)$ communication [3]. Hence, using the theorem from last lecture, we can conclude that MAJ isn't regular.

Here is an alternative proof, which is shorter. It fits with our intuition and with the upper bound, that Alice has to send the number of 1's in her input which requires $\log n$ message length.

Proof. Assume for contradiction that there exists a protocol where on inputs $w = w^A w^B$ of length $2n$, such that Alice sends messages to Bob of length at most $\log_2(n) - 1$. This means the total number of messages Alice can send is at most $\sum_{i=0}^{\ell} 2^i \leq 2^{\ell+1} - 1 \leq 2^{\log_2(n)} - 1 < n$.

Here the input w has length $2n$ so Alice's w^A has length n. Note that the number of 1's in w^A can range from 0 to n, so there's $n + 1$ options for the number of 1s.

[2] We write "Without loss of generality" so that we don't have to write the "proof" twice for $a < b$ and $b > a$. Indeed, if $b > a$, the proof would be the same, but we would switch the role of x and y. This would be a lot of unnecessary work, so we only do the case $a < b$. You can see the Wikipedia explanation here.

[3] The communication can't be constant, as we saw if the input has length $2n$, Alice must send a message of length $\Omega(\log(n))$

Since, there are only $< n$ many different messages Alice can send, by the pigeonhole principle, there exists $x, y \in \{0,1\}^n$ where x, y have a different number of 1's and such that Alice sends the same message, α on both x and y. That is, $m(x) = m(y) = \alpha$.

Let a be the number of 1's in x and let b be the number of 1's in y, where without loss of generality $a < b$. Now let $z \in \{0,1\}^n$ with exactly $n - b$ many 1s and b many 0s.

If the input is $w = xz$, then Alice gets x and Bob gets z. Now the number of 1's in xz is $a+n-b < n$ since $a < b$. Therefore, $xz \notin$ MAJ (since $|xz| = 2n$ and there are less than n 1's). So Bob must output 0 on xz, meaning:

$$g(m(x), z) = g(\alpha, z) = 0. \tag{3}$$

On the other hand, if the input is $w' = yz$, then the number of 1's in $w' = yz$ is $b + n - b = n$, so $w' \in$ MAJ. So Bob must output 1 meaning:

$$g(m(y), z) = g(\alpha, z) = 1. \tag{4}$$

Now we have that equations (3) and (4) give a contradiction. Hence, there can't be a protocol where on inputs on length $2n$, Alice sends a message of length $< \log_2(n) - 1$. Hence, any protocol for MAJ must use at least $\log_2(n) = \Omega(\log(n))$ communication on inputs of length $2n$. ■

A "fun" exercise: You should try to to prove MAJ isn't regular by a direct argument. (Assume there exists a DFA for MAJ and get a contradiction using that DFA, like we did for EQ last lecture.)[4]

Another example.

Let First $= \{w \in \{0,1\}^* \mid w = xy$ and the first 1 in x doesn't appear after the first 1 in $y\}$.

For instance $w = 110\,011$ and $w = 010\,011$ are both in First. But $w = 011\,100 \notin$ First.

Theorem 3. *Any one-way communication protocol for* First *on inputs of length* $2n$ *requires* $\Omega(log(n))$ *communication.*

Proof. Assume for contradiction that there exists a protocol where on inputs $w = w^A w^B$ of length $2n$, Alice only sends messages to Bob of length $\ell \leq \log_2(n) - 1$. This means the total number of messages Alice can send is at most $\sum_{i=0}^{\ell} 2^i \leq 2^{\ell+1} - 1 \leq 2^{\log_2(n)} - 1 < n$.

Here the input w has length $2n$ so Alice's w^A has length n. Note that the first 1 in w^A can be at an index between 1 and n, so there's n options for the index of the first 1.

Since, there are only $< n$ many different messages Alice can send. By the pigeonhole principle, there exists x, y where x, y where the first one in x is at a different position then in y but: $m(x) = m(y) = \alpha$ (Alice sends the message on x and y).

We have that the first 1 in x appears at index as but the first 1 in y appears at index b. Without loss of generality we can assume $a > b$. So, consider a string $z \in \{0,1\}^n$ with exactly one 1 at index b.

If the input is $w = xz$, then Alice gets x and Bob gets z. The first one in x is at index a, the first one is z is at index $b < a$. So, $xz \notin$ First. So Bob must output 0 meaning:

$$0 = g(m(x), z) = g(\alpha, z). \tag{5}$$

[4]The proof should reuse ideas from the lower bound for one-way communication. What happens if two strings x, y with a different number of 1 end in the same state q ?

However, if the input is $w = yz$, then Alice gets y and Bob gets z. In both x and y the first one is at index a. So $yz \in$ First. So Bob must output 1 meaning:

$$1 = g(m(y), z) = g(\alpha, 1). \tag{6}$$

Now we have that Equations (5) and (6) give us a contradiction. Hence, there can't be a protocol where on inputs on length n, Alice sends a message of length $< \log_2(n) - 1$. Hence, any protocol for First must use at least $\log_2(n) = \Omega(\log(n))$ communication on inputs of length $2n$. ■

1 Communication protocols

Last class we discussed one-way communication protocols, and showed that lower bounds for one-way communication for a decision problem imply that the associated language is not regular. Recall that in that setup, to compute a decision problem f_L on an input w, the input w was split into two halves, where Alice sees the first half of the input, w^A, and Bob receives the second half, w^B. In a one-way protocol, Alice sends a single message to Bob (based on her input), and then Bob should output either accept or reject (where he should output accept whenever $f_L(w) = 1$ and reject whenever $f_L(w) = 0$. We then proved strong lower bounds on the one-way communication complexity of EQ, which implied that the associated language is not regular.

Today we will discuss a vast generalization of the one-way communication model, which is known as the 2-party communication complexity model. This model is very well studied and has an enormous variety of applications across many subareas of computer science and mathematics, including applications in game theory, distributed computing, AI, blockchains, complexity theory, and combinatorics. In a 2-party protocol, both Alice and Bob send messages to another one another in a series of rounds, and at the end the last player should announce the final answer. Thus 2-party protocols generalize the one-round case in that they can have any number of rounds. We now describe the model a bit more formally.

Once upon a time, there were two persons, Alice and Bob. Suppose that Alice and Bob have $w^A \in \{0,1\}^*$ and $w^B \in \{0,1\}^*$, as inputs respectively, and their goal is to compute $f(w^A, w^B)$ for some function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ and for both party to learn $f(w^A, w^B)$. To achieve this Goal, Alice and Bob communicates. Their communication might be something like the following:[5]

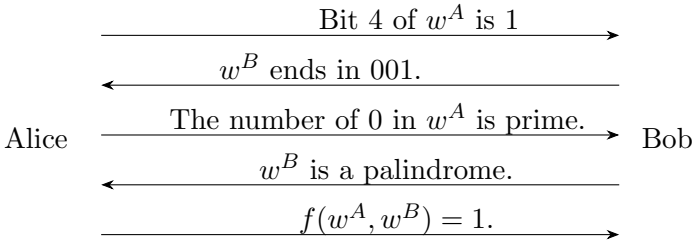

Figure 1: An example of communication between Alice and Bob.

We will assume that the last message of the protocol always has to be the value of $f(w^A, w^B)$ (which is 0 or 1). Below are a few examples of communication problems, that is, functions for which we study communication protocols. The convention is to use all capital letters for communication problems.

[5]This is just an arbitrary example.

Example 4. $\mathrm{EQ}(x,y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$.

Example 5. $\mathrm{MAJ}(x,y) = \begin{cases} 1 & \textit{if } xy \textit{ contains more 1's than 0's} \\ 0 & \textit{otherwise} \end{cases}$.

Example 6. $\mathrm{PARITY}(x,y) = \begin{cases} 1 & \textit{if } xy \textit{ contains an odd number of } 1\textit{'s} \\ 0 & \textit{otherwise} \end{cases}$.

The next example shows that for any communication problem, there is always a trivial protocol of cost $|w^A| + 1$; in fact, it is even a one-way protocol.

Example 7. *For any communication problem f, on input $w = w^A w^B$, we have the following trivial protocol: Alice first sends her whole input w^A to Bob. Bob then computes $f(w^A, w^A)$ and sends it to Alice. This protocol takes $O(|w^A| + 1)$ bits of communication, which is $O(|w|)$.*

However, for many communication problems f, there are much better protocols than the naïve one, in terms of total bits of communication. For example, we will see soon that MAJ has a protocol with $O(\log n)$ communication, and PARITY has a protocol with $O(1)$ communication.

In this and the next lecture, we will focus on the following question:

> For a given communication problem, what is the least amount of communication any protocol for the problem have to use?

Definition 8. *The number of bits of communication used by a protocol is called the* communication complexity *of the protocol, which is some function $c : \mathbb{N} \to \mathbb{N}$. $c(n)$ is the maximum number of bits communicated by Alice and Bob on input $w = w^A w^B$.*

We also define the communication complexity of a communication problem to be the minimum communication complexity among all protocols for the problem.

The rationale that we focus on communication and not the local computation by Alice or Bob themselves is that communication usually takes much longer than local computation. For example, Alice might be in Manaus (Amazon) while Bob is in Yakutsk (Siberia). As another example, when designing chips, it is common for the bottleneck to be the communication between different parts of the chip, and this is also one of the reasons why people started to study communication complexity.

Now we discuss better-than-naïve protocols for a couple of well-studied problems. First, we review the simple and efficient protocol for PARITY that we discussed last class.

Example 9 (Protocol for PARITY)**.** *Let the input be $w = w^A w^B$, $|w^A| = \lceil |w|/2 \rceil$, $|w^B| = \lceil |w|/2 \rceil$. Alice computes the number of 1's in w^A mod 2; that is, she sends Bob '1' if the number of 1's is odd, and '0' otherwise. Then Bob computes the number of 1's in y mod 2. From this he can determine the number of 1's in the whole string w mod 2 and outputs this value. (If the the number of 1's in w^A and the number of 1's in w^B are both odd or both even, then he outputs '0' and otherwise he outputs '1'.)*

There are 2 bits of communication in this protocol regardless of $|w|$ and therefore PARITY has an $O(1)$-cost communication protocol. We can prove that this is optimal, as Bob needs 1 bit from Alice to learn about the parity of the number of 1's in x, and similarly Alice needs 1 bit of message from Bob.

Remark 10. *Bob already knows the answer before sending it to Alice. However, our requirement is that the last message is the value of* $f(w^A, w^B)$.

Next we discuss the MAJ problem.

Example 11 (Protocol for MAJ)**.** *Let* $w = w^A w^B$ *be the input. Alice first sends the number of* 1*'s in* w^A *and the number of* 0*'s in* w^A *(both in binary representation) to Bob. Bob computes the number of* 1*'s in* w^B *and the number of* 0*'s in* w^B*, and then computes the total number of* 1*'s in* $w^A w^B$ *and also the total number of* 0*'s, using Alice's message. Bob compares the two numbers and sends the answer to Alice.*

If $|w| = n$*, the protocol takes* $O(\log n)$ *communication, since the two numbers Alice sends to Bob have binary representation length at most* $\log |w^A| = O(\log n)$.

For EQ on inputs of length $2n$, the naïve protocol uses $O(n)$ communication. Last class we proved that a one-way communication protocol for EQ (on inputs of length $2n$) requires $\Omega(n)$ communication. We will now generalize that lower bound argument to prove the same lower bound for *all* communication protocols, with any number of rounds of communication between Alice and Bob.

Theorem 12. *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 is a figure of the EQ matrix on inputs $w = w^A w^B$ where w has length 6. The rows are labelled 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 labelled 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 $\mathrm{EQ}(w^A, w^B)$. Crucially the matrix is the identity matrix, which we will see has maximal communication complexity even for general protocols.

We now want to understand a two-way protocol in terms of this matrix. Assume that in each round of communication Alice and Bob alternate, and in every round they send one bit (which will depend on their input, together with the messages that have been received so far). Assume without loss of generality that Alice goes first, so she sends a one-bit message $m(w^A)$ which partitions her inputs (the rows) into two subsets. The figure below 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*. A rectangle R is a set $X' \times Y'$ where $X' \subseteq X$ and $Y' \subseteq Y$.

Now in the next 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 3.

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. Finally in the 4th round, Bob sends the final bit which divides the matrix into 16 subrectangles.

Since this final bit determines the output, we can see that each of the 16 subrectangles corresponds to a unique *transcript* where the last bit of the transcript is the output. Thus if the protocol is correct, in the final figure the matrix is divided into 16 *monochromatic* subrectangles, meaning that for each subrectangle, all cells in that subrectangle are either all labelled by "1" or all labelled by

"0" (the last bit that Bob sent), and this label is the value output by the protocol on each of the inputs in the subrectangle.

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 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 2: 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 3: Round 1 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 4: Round 2 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 5: 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 6: Last round of EQUALITY Protocol