CS Theory (Spring '25) Fall, 2025

Lecture Notes: Proving a Language is Not Regular

Instructor: *Toniann Pitassi*

In this lecture we will give a fairly simple argument showing that there exist languages that are not regular. That is, we will give some specific language and prove that it cannot be computed by any DFA, and therefore by the equivalences we have shown, it follows that: this language also cannot be computed by any NFA, and cannot be expressed by a regular expression, and cannot be recognized by any constant-space streaming algorithm.

Definition 1. *Let* EQ $= \{w \in \{0,1\}^* \mid w = uu\}$. *That is,* EQ *is the set of all strings such that the first half of the string is equal to the second half of the string.*

Theorem 2. EQ *is not regular.*

We will first give a direct argument showing that no DFA can accept EQ. Then we will give a more general and abstract treatment of the lower bound argument, which will hopefully give intuition for how and when this type of argument can be applied.

Intuition for proof. Intuitively, since a DFA can only read each symbol once, it seems like it would have to remember the first n symbols exactly in order to determine if the first half equals the second half. Formally this means that there needs to be 2^n distinct states, one for each string of length n. If not then (by the pigeonhole principle) there would be two different strings, u and v, each of length n that lead to the same state, call it state q. But this would cause an error since when reading the second half of the string starting at q, we can't distinguish whether it equals the first string u or the second string v. We now proceed to the formal argument.

Proof. Assume for sake of contradiction that there is a DFA, $M = (Q, \Sigma = \{0,1\}, q_0, F, \delta)$ that accepts EQ, and let $|Q| = S$; that is, the number of states in M is S. First a bit of notation. For any state $q \in Q$ and string $w \in \{0,1\}^*$, let $\delta(q, w)$ be the state that M will be in if we run M on the string w, starting the computation in state q. (Note that q may not be the actual start state of M, but still this definition makes sense.)

We will consider what happens when we run M starting in the start state q_0 on inputs of length n, where n satisfies $2^n > S$. Since there are 2^n inputs of length n and $2^n > S$, by the pigeonhole principle, there must exist some state $q \in Q$ and two different inputs $u \neq v$, $|u| = |v| = n$ such that $\delta(q_0, u) = \delta(q_0, v) = q$. Now since $uu \in L$, when we run M on uu starting at q_0, it will be in q after reading the first u (by the above argument), and then it will end in an accept state when we further process the second copy of u, starting in state q. In other words, we must have $\delta(q, u) \in F$ since the string uu should be accepted by M. By similar reasoning, since $vv \in L$, $\delta(q, v) \in F$.

But on the other hand, the string uv must also end up in an accept state, since $\delta(q_0, u) = q$, and $\delta(q, v) \in F$. But since uv is not in L, we have reached a contradiction. ■

1 One-way Communication Model

We will now given an abstraction of the above lower bound proving that EQ is not regular (and therefore also not recognizable by any constant-space streaming algorithm). The abstraction is

via the notion of a communication complexity protocol. We will see that for any language L, if it has a streaming algorithm with space $s(n)$, then there is a one-way communication protocol of complexity $O(s(n))$ that recognizes L. Then using the same argument that we gave above (for proving a lower bound for EQ), we will show that there is no one-way communication protocol for EQ of constant cost. Putting these two things together gives a more abstract proof that EQ is not regular (as we saw last lecture regular languages have constant space streaming algorithms). While this may seem like an unnecessarily complicated argument (given that we already proved that EQ is not regular with a one-paragraph proof), by abstracting the argument in this way, we will gain intuition about how to prove that other languages are not regular. Another reason to abstract the argument is because communication complexity is a beautiful concept. It is a natural generalization of constant-space algorithms that can only read the input once (e.g., DFAs and constant-space streaming algorithms), and as we will see, it has literally hundreds of applications throughout computer science.

Imagine two people, Alice and Bob, who want to compute some joint function or property of their inputs. For example, suppose Alice has an input $x \in \{0,1\}^n$ and Bob also has an input $y \in \{0,1\}^n$. Say they want to know if their inputs are equal. Now Alice and Bob live far apart and they don't have any way to communicate cheaply. In a one-way protocol (from Alice to Bob), Alice gets to send a single message, $m(x)$, about her input to Bob. Of course she could simply send Bob x and then he could figure out whether or not $x = y$. But we want to design a better protocol, where she sends a much shorter message, if possible.

More formally, we have the following definitions of a communication problem associated with a language.

Definition 3.

1. *Given a string $w \in \{0,1\}^*$, we will first partition w into two parts, $w = w^A w^B$. In the default partitioning, w^A and w^B have roughly equal length; that is, w^A consists of the first $\lceil |w|/2 \rceil$ symbols of w, and w^B consists of the last $\lfloor |w|/2 \rfloor$ bits of w.*[1] *However sometimes we may want to consider a more imbalanced partition of the input w. If we do not specify the partition, then we will use the default partitioning.*

2. *Given a language L over $\{0,1\}$, the associated decision problem is defined to be $f_L(w) = 1$ if $w \in L$, and $f_L(w) = 0$ otherwise. Now we define the communication problem associated with L to be: on input $w \in \{0,1\}^*$, Alice gets w^A and Bob gets w^B and they want to output $f_L(w^A w^B)$.* [2]

3. *A one-way communication protocol for solving f_L is defined as follows.* [3]

   Alice has a function $m : \{0,1\}^* \to \{0,1\}^*$ to compute the message she sends to Bob. And Bob has a function $g : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, taking as input Alice's message and his input to give the answer (0 or 1).

   Given an input $w = w^A w^B \in \{0,1\}^*$, Alice is given the string w^A and Bob receives the string w^B. The protocol proceeds in two phases. In the first phase, Alice computes a message, $m(w^A) \in \{0,1\}^*$, and sends it to Bob. Then in the second phase, Bob outputs a bit

[1] What does that mean ? When the input has even length, Alice gets the first half, Bob the second. When the input has odd length, Alice also receives the middle symbol. For instance if $w = 001011$, then $w^A = 001, w^B = 011$. If $w = 0001111$ then $w^A = 0001$ and $w^B = 111$.

[2] Note that for each partition we actually get different communication problems. For now and for most our discussion, we will use the default partition.

[3] We may sometimes write $f_L(uv)$ as $f_L(u, v)$ to make it more clear that when solving f_L via a one-way communication protocol, Alice gets the first half u and Bob gets the second half, v, of the input.

$g(m(w^A), w^B)$ is the function taking Alice's message and Bob that depends on Alice's message and his input w^B. We say that the protocol (given by the two functions m and g) solves f_L if for every $w = w^A w^B$, Bob's output bit $g(m(w^A), w^B)$ is equal to $f_L(w^A w^B)$.

4. *The complexity of the protocol is denoted by $c(n) : \mathbb{N} \to \mathbb{N}$. Is the the maximal length of $m(w^A)$ over all $w \in \{0,1\}^n$.* [4]

Example 4. *Suppose that $L = \text{PARITY}$, which contains the set of all strings that contain an odd number of 1's. Then $f_{\text{PARITY}}(w)$ is 1 if w contains an odd number of 1's, and f_L is 0 if w contains an even number of 1's. Then there a simple protocol where Alice only sends a message of length one! Alice on w^A computes the parity of the number of 1's in w^A and sends this single bit to Bob. Bob then computes the parity of the number of 1's on w^B and adds this to Alice's bit. If this is odd then Bob knows that $f_{\text{PARITY}}(w^A, w^B) = 1$ and otherwise $f_{\text{PARITY}}(w^A, w^B) = 0$.*

Example 5. *Suppose that $L = \text{EQ}$. It will be useful view the associated communication problem f_{EQ} on inputs w of length $2n$ as an N-by-N matrix, where there are $N = 2^n$ rows corresponding to all possible values of the first half w^A (these are all possible inputs that Alice can get on inputs of length $2n$), and similarly there are $N = 2^n$ columns corresponding to all possible values for the second half w^B (the set of all possible inputs for Bob). An entry (w^A, w^B) of the matrix corresponds to the input $w = w^A w^B$, and this entry is labelled 1 if this string w is in EQ, and otherwise the entry is labelled 0. Note that the matrix is nothing other than the identity matrix! We will see next that any one-way protocol this communication matrix requires maximal communication.*

	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 1: Communication Matrix for EQUALITY

Theorem 6. *Any one-way protocol for* EQ *on inputs of length $2n$ requires communication complexity n.*

[4] That is, how long is Alice message when the input w has length n.

Proof. Assume for sake of contradiction that there is a one-way protocol for EQ on inputs of length $2n$ of communication complexity $s(2n) < n$ (meaning $s(2n) \leq n-1$). Let $X = \{x \mid x \in \{0,1\}^n\}$ be the set of all possible inputs for Alice. Note that there are at most $S = \sum_{i=0}^{s(2n)} 2^i \leq \sum_{i=0}^{n-1} 2^i = 2^n - 1$, possible messages that Alice can send to Bob. The messages Alice send yields a partition of Alice's inputs X into $2^n - 1$ groups. (For each message α, the group of inputs associated with message α is the set of $x \in X$ such that $m(x) = \alpha$.) We have $S \leq 2^n - 1 < 2^n = |X|$, and therefore by the pigeonhole principle there exists a message $\alpha \in \{0,1\}^s$ and two different inputs $u, v \in \{0,1\}^n$, such that $m(u) = m(v) = \alpha$; that is, there are two different inputs u, v for Alice such that Alice sends the same message on both u and v.

Now if Bob's input was equal to u, then since $\mathrm{EQ}(uu) = 1$, this implies that $1 = g(m(u), u) = g(\alpha, u)$. But on the other hand, if Alice's input is u and Bob's input is v, since $\mathrm{EQ}(vu) = 0$, we have $0 = g(m(v), u) = g(\alpha, u)$. So we have a contradiction since have $g(\alpha, u) = 0$ and $g(\alpha, u) = 1$. ■

The picture below illustrates a one-way protocol for EQ where $n = 3$. As in the picture above, there are $N = 2^n$ rows corresponding to all possible inputs for Alice, and similarly there are N columns corresponding to all possible inputs for Bob. In this protocol, say Alice sends messages of length exactly 2, so there are $S = 2^2 = 4$ possible messages. This partitions her 8 possible inputs (the rows of the matrix) into 4 different subsets, indicated by the dark blue horizontal lines. Let's label the first subset (consisting of just 000) by the message 00, the second subset (containg 001,010,011) by the message 01, the third subset by the message 10, and the last subset by the message 11. Since $S < N$ we can see that there exists a subset containing more than one Alice input. For example, in this protocol Alice will send the same message, 01, on all three inputs (001,010,011) in the second subset. Now this leads to an incorrect protocol for EQ: Bob's function g depends on his input v and Alice's message. So if she sends the message 01, he only learns that her input is one of 001,010,011, and not which of these three inputs she has. So when he has one of these three inputs as well, he cannot distinguish whether his input is equal to hers or not.

We also want to point out that while this picture is just an illustration of one example, it can be viewed as general up to permutations of the rows. That is, if Alice's messages have length s, it partitions the rows into disjoint subsets although not necessarily consecutive subsets as in the picture. But we can permute the rows/colums so that under this permutation we get a picture like the one below.

Finally we show that a streaming algorithm of space $s(n)$ implies a protocol with $s(n)$ communication. The key idea is that a streaming algorithm reads input symbols one by one. So it doesn't need to see all the input w at once: Alice can just feed it the first half w^A, and then Bob feeds it the second half w^B.

Theorem 7. *Let L be a language with a streaming algorithm $\mathcal{A} = (\Sigma, \Pi = \{0,1\}, I, \delta, \gamma)$ for L using space $s(n)$, then there is a a one-way communication protocol for solving L of cost $O(s(n))$.*

Proof. On input $w = w^A w^B \in \{0,1\}^n$, Alice gets w^A and Bob gets w^B. First, Alice runs $\mathcal{A}$ on w^A, once the algorithm has seen all of w^A she sends the final memory M of the algorithm to Bob. Now Bob starts a copy of $\mathcal{A}$ with memory set to M (the memory Alice sent to him) and runs this copy on w^B. That is, knowing the memory that $\mathcal{A}$ ended after seeing w^A, Bob continues to simulate the computation of $\mathcal{A}$ on the second half of the input w.

Finally, if the algorithm $\mathcal{A}$ outputs 0 ($\mathcal{A}$ rejects $w^A w^B$, meaning $w \notin L$), Bob outputs 0. If $\mathcal{A}$ outputs 1 ($\mathcal{A}$ accepts $w^A w^B$, meaning $w \in L$), Bob outputs 1.

It's easy to see this is indeed a correct protocol for L. How much communication is used by this protocol ? Alice sends the memory M to Bob, we must have $|M| \leq s(n)$ (by the definition of space usage), so we have that the cost of the protocol is also at most $s(n)$. ■

	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: One-way Protocol for EQUALITY

Why did the above theorem matter ? Recall that a language is regular if and only if it has a $O(1)$ space streaming algorithm. Hence if there's no $O(1)$ cost one-way communication protocol for L, then L can't be regular !

Corollary 8. *If there are no constant cost one-way communication protocol for L, then L isn't regular.*

Combining the above the above corollary with our lower bound on the communication needed for EQ we have:

Corollary 9. *The language* EQ *is not regular.*

Note that in the one-way communication model we always fixed the partition of w into two roughly equal sized pieces $w = xy$, where Alice received the first half and Bob the second half. What would happen if we allowed any partition of the symbols of w into two pieces, not necessarily consecutive ones? Does a DFA for a language still imply a one-way protocol under an arbitrary partition? Can you give a counterexample, showing a partition where EQ is easy.