Introduction to Complexity Theory*
Lecture 21: Communication Complexity

Guest Lecturer: Ran Raz
Notes taken by Amiel Ferman and Noam Sadot

July 2, 1999

Summary: This lecture deals with Communication Complexity, which is the analysis of the amount of information that needs to be communicated by two parties which are interested in reaching a common computational goal. We start with some basic definitions and simple examples. We continue to consider both deterministic and probabilistic models for the problem, and then we develop a combinatorial tool to help us with the proofs of lower bounds for communication problems. We conclude by proving a probabilistic linear communication complexity lower bound for the problem of computing the inner product of two vectors where initially each party holds one vector.

1 Introduction

The communication problem arises when two or more parties (i.e. processes, systems etcetera) need to carry out a task which could not be carried out alone by each of them because of lack of information. Thus, in order to achieve some common goal, defined as a function of their inputs, the parties need to communicate. Often, the formulation of a problem as a communication problem serves merely as a convenient abstraction; for example, a task that needs to share information between diefferent parts of the same CPU could be formulated as such. Communication complexity is concerned with analaysing the amount of information that must be communicated between the different parties in order to correctly perform the intended task.

2 Basic model and some examples

In order to investigate the general problem of communication we state a few simplified assumptions on our model:

1. There are only two parties (called player 1 and player 2)
2. Each party has unlimited computing power and we are only concerned with the communication complexity
3. The task is a computation of a predefined function of the input

* Lecture Notes for a course given by Oded Goldreich at the Weizmann Institute of Science, Israel.

As we shall see, this model is rich enough to study some non-trivial and interesting aspects of communication complexity.

The input domains of player 1 and player 2 are the (finite) sets X and Y respectively. The two players start with inputs $x \in X$ and $y \in Y$, and their task is to compute some predefined function $f(x, y)$. At each step, the communication protocol specifies which bit is sent by one of the players (alternately), and this is based on information communicated so far as well as on the initial inputs of the players.

Let us see a few examples:

1. Equality Function (denoted EQ):

   The function $f(x, y)$ is defined as:

   $f(x, y) = 1$ if $x = y$

   $f(x, y) = 0$ if $x \neq y$

   That is, the two players are interested to know wheather their initial inputs are equal.

2. Disjointness:

   The inputs are subsets: $x, y \subseteq \{1, \dots, n\}$

   $f(x, y) = 1$ iff $x \cap y \neq \emptyset$

3. Inner Product (denoted IP):

   The inputs are: $x, y \in \{0, 1\}^n$

   $f(x, y) = \sum_i^n x_i \cdot y_i \bmod 2$

3 Deterministic versus Probabilistic Complexity

We begin with some definitions:

Definition 1 *A deterministic protocol P with domain $X \times Y$ and with range Z (where X and Y are the input domains of player 1 and player 2 respectively and Z is the domain of the function f) is defined as a deterministic algorithm in which at each step it specifies a bit to be sent from one of the players to the other. The output of the algorithm, denoted $P(x, y)$ (on inputs x and y), is the output of each of the players at the end of the protocol and it is required that:*

$$\forall x \in X, y \in Y \qquad : \qquad P(x, y) = f(x, y)$$

Definition 2 *The communication complexity of a deterministic protocol is the worst case number of bits sent by the protocol for some inputs.*

Definition 3 *The communication complexity of a function f is the minimum complexity of all deterministic protocols which compute f. This is denoted by $CC(f)$.*

A natural relaxation of the above defined deterministic protocol would be to allow each player to toss coins during his computation. This means that each player has an access to a random string and the protocol that is carried out depends on this string. The way to formulate this is to determine a distribution Π from which the random strings each player uses are sampled uniformly, once the strings are chosen, the protocol that is carried out by each of the players is completely deterministic. We consider the Monte-Carlo model, that is, the protocol should be correct for a large fraction of the strings in Π.

Note that the above description of a randomized protocol implicitly allows for two kinds of possibilities: one in which the string that is initially sampled from Π is common to both players, and the other is that each player initially samples his own private string so that the string sampled by one player is not visible to the other player. These two possibilites are called respectively the public and the private model. How are these two models related ? First of all, it is clear that any private protocol can be simulated as a public protocol: the strings sampled privately by each user are concatenated and serve as the public string. It turns out that a weaker reduction exists in the other direction: any public protocol can be simulated as a private protocol with a small increase in the error and an additive of $O(\log n)$ bits of communication; the idea of the proof is to show that any public protocol can be transformed to a protocol which uses the same amount of communication bits but only $O(\log n + \log \delta^{-1})$ random bits with an increase of δ in the error. Next, each player can sample a string of that length and send it to the other player thus causing an increase of $O(\log n + \log \delta^{-1})$ in the communication complexity and of δ in the error. In view of these results we shall confine ourselves to the public model.

Definition 4 *A randomized protocol P is defined as an algorithm which initially samples uniformly a string from some distribution Π and then carries on exactly as in the deterministic case. The sampled string is common to both player, i.e. - this is the public model. It is required that an ϵ - error protocol will satisfy:*

$$\forall x \in X, y \in Y \qquad Pr_{r \in_R \Pi}[P(x,y) = f(x,y)] \geq 1 - \epsilon$$

Note that in a randomized protocol, the number of bits communicated may vary for the same input (due to different random strings). Hence the communication complexity is defined with respect to the strings sampled from Π. One can define the communication complexity of a protocol, viewed as a random variable (with respect to the distribution Π, and the worst possible input), in the average case. However we prefer the somewhat stronger worst-case behaviour:

Definition 5 *The communication complexity of a randomized protocol P on input (x, y) is the maximum number of bits communicated in the protocol for any choice of initial random strings of each player. The communication complexity of a randomized protocol P is the maximum communication complexity of P over all possible inputs (x, y)*

Definition 6 *The communication complexity of a function f computed with error probability ϵ, denoted $CC_\epsilon(f)$ is the minimum communication complexity of a protocol P which computes f with error probability ϵ.*

In Lecture 3 we have actually considered a private case of the above definition, one in which there is no error; i.e., $CC_0(f)$.

4 Equality revisited and the Input Matrix

Recall the Equality Function defined in Section 2, in which both players wish to know wheather their inputs (which are n-bit strings) are equal (i.e. whether $x = y$). Let us first present a randomized protocol which computes EQ with a constant error probability and a constant communication complexity:

Protocol for player i (i = 1,2) ($input_1 = x$ and $input_2 = y$):

1. sample uniformly an n-bit string r (this r is common to both players - public model)

2. compute $< input_i, r >_2$ (the inner product of $input_i$ and r mod 2)

3. send the product computed and receive the product computed by the other player (single bit)

4. if the two bits are equal then output 1 else output 0

If the inputs are equal, i.e. $x = y$, then clearly $< x, r >_2 = < y, r >_2$ for all r-s and thus each player will receive and send the same bit and will decide 1. However, if $x \neq y$, then for a string r sampled uniformly we have that $< x, r >_2 = < y, r >_2$ with probability exactly one half. Thus, the error probability of a single iteration of the above protocol is exactly one half. Since at each iteration we sample a random string r independantly from other iterations we get that after carrying out the protocol for exactly C times the error probability is exactly 2^{-C}. Furthermore, since the number of bits communicated in each iteration is constant (exactly two bits), we get that after C iterations of the above protocol, the communication complexity is $O(C)$. Hence, if C is a constant, we get both a constant error probability and a constant communication complexity (2^{-c} and $O(1)$ respectively for a constant c). However, if we choose C to be equal to $log(n)$ then the error probability and the communication complexity will be, respectively, $\frac{1}{n}$ and $O(log(n))$.

We now present an alternative protocol for solving EQ that also achieves an error probabilty of $\frac{1}{n}$ and communication complexity of $O(log(n))$. Interestingly, this protocol is (already) in the private model:

We present both (n-bit strings) inputs as the coefficients of polynomials over $GF(p)$ where p is an arbitrary fixed prime between n^2 and $2n^2$ (results in number theory guarantee the existence of such a prime). So, both inputs may be viewed as:

input of player 1:

$$A(x) = \sum_{i=0}^{n-1} a_i \cdot x^i \bmod p$$

input of player 2:

$$B(x) = \sum_{i=0}^{n-1} b_i \cdot x^i \bmod p$$

Protocol for player 1 (For player 2: just reverse A and B)

1. choose uniformly a number t in $GF(p)$

2. compute $A(t)$

3. send both t and $A(t)$ to the other player

4. receive s and $B(s)$ from other player

5. if $A(s) = S$ then decide 1 else decide 0

Clearly, if the inputs are equal then so are the polynomials and thus necessarily $A(t) = B(t)$ for every $t \in GF(p)$. If however, $A \neq B$, then these polynomials have at most $n - 1$ points on which they agree (i.e. t-s for which $A(t) = B(t)$) since their difference is a polynomial of degree

$n-1$ which can have at most $n-1$ roots. So the probability of error in this case is $\frac{n-1}{p} \leq \frac{n}{n^2} = \frac{1}{n}$. Notice that since t and $B(t)$ are $O(lgn)$ bits long, we may conclude that $CC_{\frac{1}{n}}(EQ) = O(lgn)$.

Proofs of lower bounds which relate to certain families of algorithms usually necessitate a formalization that could express in a non-trivial way the underlying structure. In our case, a combinatorial view proves to be effective: (Recall that the input domains of both parties are denoted X and Y) we may view the protocol as a process which in each of its steps partitions the input space $X \times Y$ into disjoint sets such that at step t each set includes exactly all input pairs which according to their first t bits cause the protocol to "act" the same (i.e., communicate exactly the same messages during the algorithm). Intuitively, each set at the end of this partioning process is comprised of exactly all pairs of inputs that "cause" the protocol to reach the same conclusion (i.e., compute the same output).

A nice way to visualize this is to use a matrix: each row corresponds to a $y \in Y$ and each column corresponds to a $x \in X$. The value of the matrix in position (i, j) is simply $f(i, j)$, where f is the function both parties need to compute. This matrix is called the Input Matrix. Since the Input Matrix is just another way to describe the function f,we may choose to talk about the communication complexity of an Input Matrix A - dentoed $CC(A)$ instead of the communication complexity of the corresponding function f. For example, the matrix corresponding to the Equality Function is the indentity matrix (since the output of the Equality Function must be 1 iff the inputs are of the form (i, i) for each input pair). The above mentioned partioning process can now be viewed as a partioning of the matrix into sets of matrix elements. It turns out that these sets have a special structure, namely rectangles. Formally, we define

Definition 7 *A* rectangle *in $X \times Y$ is a subset $R \subseteq X \times Y$ such that $R = A \times B$ for some $A \subseteq X$ and $B \subseteq Y$.* (Note that elements of the rectangle, as defined above, need not be adjacent in the input matrix.)

However, in order to relate our discussion to this definition we need an alternative characterization of rectangles given in the next proposition:

Proposition 4.1 *$R \subseteq X \times Y$ is a rectangle iff $(x_1, y_1) \in R$ and $(x_2, y_2) \in R \;\Rightarrow\; (x_1, y_2) \in R$*

Proof: $\Rightarrow$ If $R = A \times B$ is a rectangle then from $(x_1, y_1) \in R$ we get that $x_1 \in A$ and from $(x_2, y_2) \in B$ we get that $y_2 \in B$ and so we get that $(x_1, y_2) \in A \times B = R$.

$\Leftarrow$ We define the sets $A = \{x | \exists y \ s.t. \ (x, y) \in R\}$ and $B = \{y | \exists x \ s.t. \ (x, y) \in R\}$. On the one hand it is clear that $R \subseteq A \times B$ (directly from A and B's definition). On the other hand, suppose $(x, y) \in A \times B$. Then since $x \in A$ there is a y' such that $(x, y') \in R$ and similiarly there is an x' such that $(x', y) \in R$ from this, according to the assumption we have that $(x, y) \in R$. ■

We shall now show that the sets of matrix elements partitioned by the protocol in the sense described above actually form a partition of the matrix into rectangles: Suppose both pairs of inputs (x_1, y_1) and (x_2, y_2) cause the protocol to exchange the same sequence of messages. Since the first player (with input x_1) cannot distinguish at each step between (x_1, y_1) and (x_1, y_2) (he computes a function of x_1 and the messages so far in any case) then he will communicate the same message to player 2 in both cases. Similarly, player 2 cannot distinguish at each step between (x_2, y_2) and (x_1, y_2) and will act the same in both cases. We showed that if the protocol acts the same on inputs (x_1, y_1) and (x_2, y_2) then it will act the same on input (x_1, y_2), which, using proposition 4.1 establishes the fact that the set of inputs on which the proctocol behaves the same is a rectangle.

Since the communication is the same during the protocol for the pair of inputs (x_1, y_1) and (x_2, y_2) (and for the pairs of inputs in the rectangle defined by them, as was explained in the last paragraph) then the protcol's output must be the same for these pairs, and this implies that the value of the f function must be the same too. Thus, a deterministic protocol partitions the Input Matrix into rectangles whose elements are identical, that is, the protocol computes the same output for each pair of inputs in the rectangle. We say that a deterministic protocol partitions the Input Matrix into rectangles of monochromatic rectangles (where color is identified with the input matrix value). Since at each step the protocol partitions the Input Matrix into two (usually not equal in size) parts we have the following:

Fact 4.2 *A deterministic protocol P of communication complexity k partitions the Input Matrix into at most 2^k monochromatic rectangles*

Recalling the fact that the Input Matrix of a protocol to the equality problem is the identity matrix, then since the smallest monochromatic rectangle that contains each entry of 1 in the matrix is the singleton matrix which contains exactly one element and since the matrix is of size $2^n \times 2^n$ (for inputs of size n), we get that every protocol for the equality problem must have partitioned the Input Matrix into at least $2^n + 1$ monochromatic rectangles (2^n for the 1's and at least 1 for the zeros). Thus, from Fact 4.2 and from the trivial protocol for solving EQ in which player 1 sends its input to the player 2 and player 2 sends to player 1 the bit 1 iff the inputs are equal ($n + 1$ bits of communication), we get the following corollary:

Corollary 8 $CC(EQ) = n + 1$

5 Rank Lower Bound

Using the notion of an Input Matrix developed in the previous section, we now state and prove a useful theorem regarding the lower bound of communication complexity:

Theorem 9 *Let A be an Input Matrix for a certain function f, then $CC(A) \geq log_2(r_A)$ where r_A is the rank of A over any fixed field F*

Proof: The proof is by induction on $CC(A)$.

Induction Base: If $CC(A) = 0$ then this means that both sides were able to compute the function f without any communication. This means that for every pair of inputs, both sides compute a constant function which could be either $f(x, y) = 1$ or $f(x, y) = 0$ for all x and y. This implies that A must be the all 0-s or the all 1-s matrix, and so, by definition $r_A \in 0, 1$. Thus, indeed, $CC(A) \geq log_2(r_A)$ as required.

Induction Step: Suppose the claim is true for $CC(A) \leq n - 1$ and we shall prove the claim for $CC(A) = n$. Consider the first bit sent: This bit actually partitions A into two matrices A_0 and A_1 such that the rest of the protocol can be seen as a protcol that relates to only one of these matrices. Since the maximal communication complexity needed for both matrices cannot surpass $n - 1$ (otherwise $CC(A)$ could not have been equal to n), we get the following equation:

$$CC(A) \geq 1 + Max\{CC(A_0), CC(A_1)\} \geq 1 + Max\{log_2(r_{A_0}), log_2(r_{A_1})\} \tag{1}$$

where the second inequality is by the induction hypothesis. Now, since $r_A \leq r_{A_0} + r_{A_1}$, we have that $r_A \leq 2 \cdot Max\{r_{A_0}, r_{A_1}\}$. Put differently, we have $Max\{log_2(r_{A_0}), log_2(r_{A_1})\} \geq log_2(r_A) - 1$. Combining this with Eq. (1) we get that $CC(A) \geq 1 + log_2(r_A) - 1 = log_2(r_A)$. ■

Applying this theorem to the Input Matrix of the equality problem (the identity matrix), we easily get the lower bound $CC(EQ) \geq n$. A linear lower bound for the deterministic communication complexity of the Inner Product problem can also be achieved by applying this theorem. In the next section we'll see a linear lower bound on the randomized communication complexity of the Inner Product function.

6 Inner-Product lower bound

Recalling Inner-Product problem from section 2, we prove the following result:

Theorem 10 $CC_\epsilon(IP) = \Omega(n)$

To simplify the proof, and the mathematical techniques needed for the proof, we will assume that $0 < \epsilon < (\frac{1}{4} - \tau)$, for arbitarary small $\tau > 0$.

To prove the above theorem, we assume that there is a probabilistic communication protocol P in the public coin model using random string R that uses less than $n\epsilon$ communication bits, and we will show a contradiction. By definition, we know that

$$\mathrm{Pr}_R[P_R(x,y) = f(x,y)] \geq 1 - \epsilon$$

for every string x and y. Since it is true for each pair of strings, the following property is true:

$$\mathrm{Pr}_{(x,y),R}[P_R(x,y) = f(x,y)] \geq 1 - \epsilon$$

in which the probability measure over (x,y) is taken as the uniform probability over all such pairs. Changing the order of the probability measures, we obtain:

$$\mathrm{Pr}_{R,(x,y)}[P_R(x,y) = f(x,y)] \geq 1 - \epsilon$$

And since $\epsilon < 1$, we conclude that there is a fixed random string r that for the deterministic protocol induced by P_r, the following property exists:

$$\mathrm{Pr}_{(x,y)}[P_r(x,y) = f(x,y)] \geq 1 - \epsilon$$

By this method, we produce a deterministic protocol on which we can work now and prove lower bound. In contrast to the previous section, the protocol P_r should work well only for most of the inputs, but not necessarily for all of them. Proving lower bound on this deterministic protocol P_r will immediately give the lower bound of the original randomized protocol P.

The method for proving the lower bound is to show that there are not "big" enough rectangles that are not balanced in the input matrix after $n\epsilon$ time, and hence to conclude that we need more than that time. The following definition will be helpful:

Definition 11 *A rectangle $U \times V \subset \{0,1\}^n \times \{0,1\}^n$ is* big *if its size satisfies $|U \times V| \geq 2^{2n(1-\epsilon)}$. Otherwise, the rectangle is* small.

Note that there must be at least one big rectangle in the above matrix. Otherwise, using Fact 4.2 and the fact that we have at most $2^{n\epsilon}$ rectangles (for at most $n\epsilon$ communiaction), we infer that the size of the entire matrix is not more than $2^{n\epsilon} \cdot 2^{2n(1-\epsilon)} = 2^{2n(1-\frac{\epsilon}{2})} < 2^{2n}$ which leads to a contradiction.

Claim 6.1 *If P_r works in $n\epsilon$ time then there exists a big rectangle $U \times V \subset \{0,1\}^n \times \{0,1\}^n$ such that $f(x,y)$ is the same for at least $1-2\epsilon$ fraction of the elements in the rectangle $U \times V$.*

Proof: To prove the claim, we recall that $\Pr_{(x,y)}[P_r(x,y) = f(x,y)] \geq 1-\epsilon$. In other words, at most ϵ fraction of the elements in the matrix do not satisfy $P_r(x,y) = f(x,y)$. In addition, after at most $n\epsilon$ time, we will have a partition of the matrix into at most $2^{n\epsilon}$ rectangles. By Definition 11, each small rectangle has size less than $2^{2n(1-\epsilon)}$, and so the number of the elements that belong to small rectangles is less than $2^{n\epsilon} \cdot 2^{2n(1-\epsilon)} = 2^{2n(1-\frac{\epsilon}{2})} < 2^{2n-1}$, and so big rectangles contain more than half of the matrix elements. Thus, if all big rectangles have more than 2ϵ error, the total error due only to these rectangles would be more than ϵ, which leads to a contradiction. The claim follows. ■

By using this claim, we fix a big rectangle R that satisfies the conditions of the previous claim. Without loss of generality, we can assume that the majority of this rectangle is 0. If it is not 0, we just switch every element in the matrix.

Let us denote by B_n the $2^n \times 2^n$ input matrix for Inner Product problem, which looks like that:

$$B_n = \Big\{x \cdot y \bmod 2\Big\}_{(x,y)\in\{0,1\}^n\times\{0,1\}^n}$$

The inner elements are scalar products on the field $GF(2)$. The matrix B_n contains two types of elements: zeroes and ones. By switching each 0 into 1 and each 1 into -1, we get a new matrix H_n (which is a version of Hadamard matrix) that looks like that:

$$H_n = \Big\{(-1)^{x\cdot y}\Big\}_{(x,y)\in\{0,1\}^n\times\{0,1\}^n}$$

This matrix has the following property:

Claim 6.2 *H_n is an orthogonal matrix over the reals.*

Proof: We will prove that each two rows in the matrix H_n are orthogonal. Let r_x be a row corresponds to a string x and r_z be a row corresponds to a string $z \neq x$. The scalar product between these two rows is

$$\sum_{y\in\{0,1\}^n} (-1)^{x\cdot y} \cdot (-1)^{z\cdot y} =$$
$$\sum_{y\in\{0,1\}^n} (-1)^{\Delta\cdot y}$$

where $\Delta = x \oplus z$. Since $x \neq z$ there is an index $j \in \{1,\ldots,n\}$ such that $\Delta_j = 1$, then the previous expression is equal

$$\sum_{y\in\{0,1\}^n} (-1)^{\sum_{i\neq j} y_i\Delta_i + y_j} \tag{2}$$

Let us denote by $y' = y_1 \ldots y_{j-1}y_{j+1}\ldots y_n$, then we can write (2) as:

$$\sum_{y'}\sum_{y_j}(-1)^{\sum_{i\neq j} y_i\Delta_i + y_j} =$$

$$\sum_{y'}(-1)^{\sum_{i\neq j} y_i\Delta_i}\sum_{y_j}(-1)^{y_j}$$

Clearly,

$$\sum_{y_j \in \{0,1\}} (-1)^{y_j} = -1 + 1 = 0$$

which proves the claim. ■

We have 2^n rows and columns in the matrix H_n. Let us enumerate the rows by r_i, for $i = 0, 1, \ldots, 2^n - 1$. Then by the previous claim, we have the following properties, where here $\cdot$ denotes inner product over the reals:

1. $r_i \cdot r_j = 0$ for $i \neq j$
2. $r_i \cdot r_i = \|r_i\|^2 = 2^n$ for $i = 0, 1, \ldots, 2^n - 1$. This follows easily from the fact that the absolute value of each element in H_n is 1.

Thus, the rows in the matrix define an orthogonal base over the reals.

The following definition will be helpful in the construction of the proof of Theorem 10:

Definition 12 (discrepency): *The* discrepency *of a rectangle $U \times V$ is defined as*

$$D(U \times V) = \left| \sum_{(x,y) \in U \times V} (-1)^{f(x,y)} \right|$$

Let R_0 be a big rectangle (of small error) as guaranteed by Claim 6.1. Suppose without loss of generality that R_0 has a majority of zeros (i.e., at least $1 - 2\epsilon$ fraction of 0's). Recall that the size of R_0 is at least $2^{2n(1-\epsilon)}$. Thus, R_0 has a big discrepency; that is,

$$D(R_0) \; \geq \; (1 - 2\epsilon - 2\epsilon) \cdot 2^{2n(1-\epsilon)} \; = \; (1 - 4\epsilon) \cdot 2^{2n(1-\epsilon)} \tag{3}$$

On the other hand, we have an upper bound on the discrepency of any rectangle (an in particular of R_0):

Lemma 6.3 *The discrepency of any rectangle R is bounded from above by $2^{-\frac{n}{2}} \cdot 2^{2n}$*

Proof: Let us denote $R = U \times V$. The matrix H_n has the property that each bit b in B_n changes into $(-1)^b$ in H_n. Let us consider the following charactaristic vector $I_U : \{0,1\}^n \to \{0,1\}^n$ that is defined in the following way:

$$I_U(x) = \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{otherwise} \end{cases}$$

Observe that $I_U \cdot r_j$ is exactly the number of 1's minus the number (-1)'s in r_j, so

$$\begin{aligned} D(U \times V) \;\; &= \;\; \left| \sum_{j \in V} I_U \cdot r_j \right| \leq \sum_{j \in V} |I_U \cdot r_j| \\ &\leq \;\; \sum_{j \in \{0,1\}^n} |I_U \cdot r_j| \end{aligned}$$

where both inequalities are trivial. Using Cauchy-Schwartz inequality (for the second line), we obtain

$$\begin{aligned} D(R) \;\; &\leq \;\; \sum_{j \in \{0,1\}^n} 1 \cdot |I_U \cdot r_j| \\ &\leq \;\; \sqrt{2^n \cdot \sum_{j \in \{0,1\}^n} |I_U \cdot r_j|^2} \end{aligned} \tag{4}$$

Recalling that H_n is an orthogonal matrix, and the norm of each row is $\sqrt{2^n}$, we denote $\hat{r_j} = \frac{1}{\sqrt{2^n}} r_j$ which define an orthonormal base. With this notation, Eq. (4) can be written as:

$$\begin{aligned}
\sqrt{2^n \cdot \sum_{j \in \{0,1\}^n} |I_U \cdot \sqrt{2^n} \hat{r_j}|^2} &= \sqrt{2^n \cdot 2^n \cdot \sum_{j \in \{0,1\}^n} |I_U \cdot \hat{r_j}|^2} \\
&= 2^n \cdot \sqrt{\sum_{j \in \{0,1\}^n} |I_U \cdot \hat{r_j}|^2}
\end{aligned}$$

Since $\{\hat{r_j}\}_{j=0,1,\ldots,2^n-1}$ is an orthonormal base, the square root above is merely the norm of I_U (as the norm is invariant over all orthonormal bases). However, looking at the "standard" (point-wise) base, we have that the norm of I_U is $\sqrt{|U|} \leq \sqrt{2^n}$ (since each element in the vector I_U is 0 or 1). To conclude, we got that:

$$D(R) \leq 2^n \cdot \sqrt{2^n} = 2^{\frac{3n}{2}} \tag{5}$$

which proves the lemma. ■

We now derive a contradiction by contrasting the upper and lower bounds provided for R_0. By Eq. (3), we got that:

$$D(R_0) \geq (1 - 4\epsilon) \cdot 2^{2n(1-\epsilon)}$$

which is greater than $2^{3n/2}$ for any $0 < \epsilon < \frac{1}{4}$ and all sufficiently large n's (since for such ϵ the exponent is strictly bigger than $3n/2$ which for sufficiently big n's compensates for the small positive factor $1 - 4\epsilon$). In contrast, Lemma 6.3 applies also to R_0 and implies that $D(R_0) \leq 2^{3n/2}$, in contradiction to the above bound (of $D(R_0) > 2^{3n/2}$).

To conclude, we showed a contradiction to our initial hypothesis that the comminication complexity is lower than ϵn. Theorem 10 thus follows.