

Half-duplex communication complexity with adversary can be less than the classical communication complexity

Mikhail Dektiarev* Nikolay Vereshchagin*^{†‡}

Abstract

Half-duplex communication complexity with adversary was defined in [Hoover, K., Impagliazzo, R., Mihajlin, I., Smal, A. V. Half-Duplex Communication Complexity, ISAAC 2018.] Half-duplex communication protocols generalize classical protocols defined by Andrew Yao in [Yao, A. C.-C. Some Complexity Questions Related to Distributive Computing (Preliminary Report), STOC 1979]. It has been unknown so far whether the communication complexities defined by these models are different or not. In the present paper we answer this question: we exhibit a function whose half-duplex communication complexity with adversary is strictly less than its classical communication complexity.

1 Introduction

In the classical model of communication complexity introduced by Andrew Yao in [16], we consider a cooperative game between two players, Alice and Bob, who want to compute $f(x, y)$ for a given function f . Alice knows only x and Bob knows only y . To this end, Alice and Bob can communicate sending to each other messages, one bit per round. An important property of this communication model is the following: in each round one player sends a bit message while the other player receives it. Algorithms for Alice and Bob computing f are called *communication protocols*. The *depth* of a protocol is defined as the number of bits it communicates in the worst case. The minimum possible depth of a protocol to compute f is called the *communication complexity* of f .

This model was generalized in [4] to a model describing communication over the so called half-duplex channel. A well-known example of half-duplex communication is talking via walkie-talkie: one has to hold a “push-to-talk” button to speak to another person, and one has to release it to listen. We consider a communication model where players are allowed to speak simultaneously. If two persons try to speak simultaneously then they do not hear each other and both messages are lost.

*Moscow State University

[†]National Research University Higher School of Economics, Moscow, Russian Federation

[‡]This paper was prepared within the framework of the HSE University Basic Research Program.

The communication protocol over a half-duplex channel is also divided into rounds. To make such a division, we assume that the players have synchronized clocks. Every round each player chooses one of three actions: send 0, send 1, or receive. Thus we distinguish three types of rounds.

- If one player sends a bit and the other one receives, then we call this round *normal* or *classical*, in such a round the receiving player receives the sent bit.
- If both players send bits, then the round is called *spent*. Both bits are lost (such a situation occurs if both parties push “push-to-talk” buttons simultaneously).
- If both players receive, then the round is called *silent*. In such rounds players receive arbitrary bits that may be different. (We can think that those bits are chosen by a malicious adversary.)

More specifically, the described model is called the communication model with *adversary*. In [4] there were considered also two other models: the half-duplex model *with silence* (in a silent round both players receive a special symbol `silence` and hence know that a silent round occurred) and the half-duplex model *with zero* (in a silent round both players receive 0).

More specifically, the *half-duplex complexity with adversary* of a function is defined as minimum number of rounds needed to compute the function on its domain, assuming that the adversary can choose any bits in silent rounds. One can show that the half-duplex communication complexity with adversary is sandwiched between the classical complexity and a half of it [4].

The original motivation to study these kinds of communication models arose from the question of the complexity of Karchmer-Wigderson games [6] for multiplexors. A detailed exposition of this motivation and recent results in this direction can be found in [12, 10, 9, 11]. Here we will just present a toy example in which half-duplex complexity arises quite naturally. Assume that for each parameter a (ranging over a finite set) a function $f_a : X \times Y \rightarrow \{0, 1\}$ is given. Our goal is to prove that for some a the classical communication complexity of f_a is larger than a certain number d . Assume also that we can prove the same lower bound d for the classical communication complexity of the *multiplexor game* defined as follows: Alice gets a pair (x, a) , Bob gets a pair (y, b) and they want to compute $f_a(x, y)$, if $a = b$. Otherwise, if $a \neq b$, then their protocol may return any result. It may seem that from this we can deduce the sought lower bound for f_a for some a . Indeed, by way of contradiction, assume that for every a there is a classical depth- d communication protocol Π_a that computes f_a . Then consider the following depth- d communication protocol for the multiplexor game: if Alice gets a pair (x, a) and Bob a pair (y, b) , then Alice finds the lex first depth- d protocol for f_a and Bob the lex first depth- d protocol for f_b . Then they run the found protocols. If $a = b$, then they run the same protocol, which outputs $f_a(x, y)$. Otherwise, if $a \neq b$, their protocols can output different results. This problem is easy to overcome: Alice sends her output to Bob, which costs only one extra bit of communication. However, there is more complicated problem: protocols Π_a and Π_b may simultaneously receive or send bits.

Thus actually the constructed protocol is a half-duplex and not classical communication protocol for multiplexor game. Thus we need a lower bound d for half-duplex communication complexity of the multiplexor game!

There are several examples of functions whose half-duplex complexity with silence and with zero is less than the classical communication complexity. This happens for the following three functions, which are widely studied in Communication complexity: Equality function EQ_n , Disjointness function DISJ_n and Inner product IP_n . These functions are defined on bit strings of length n as follows:

$$\text{EQ}_n(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise,} \end{cases} \quad \text{DISJ}_n(x, y) = \bigwedge_{i=1}^n \neg(x_i \wedge y_i), \quad \text{IP}_n(x, y) = \sum_{i=1}^n (x_i \wedge y_i) \pmod{2}.$$

The classical communication complexity of all these functions is $n + 1$. The best known upper bounds for their half-duplex complexity are shown in the following table:

	EQ_n	IP_n	DISJ_n
Half-duplex complexity with silence	$n \log_5 2 + o(n)$	$n/2 + 2$ [3]	$n/2 + 2$
Half-duplex complexity with 0	$n \log_3 2 + o(n)$	$7n/8 + O(1)$ [7]	$3n/4 + o(n)$ [1]
Half-duplex complexity with adversary	$n + 1$	$n + 1$	$n + 1$

The best known lower bounds are the following:

	EQ_n	IP_n	DISJ_n
Half-duplex complexity with silence	$n \log_5 2$	$n/2$	$n \log_5 2$ [1]
Half-duplex complexity with 0	$n \log_3 2$	$n \log_{\frac{2}{3-\sqrt{5}}} 2$	$n \log_3 2$ [1]
Half-duplex complexity with adversary	$n \log_{2.5} 2$	$n \log_{7/3} 2$	$n \log_{2.5} 2$ [1]

Some cells of these tables contain references showing who established the corresponding bounds. The bounds with no reference are due to [4].

However no examples of functions for which half-duplex complexity with adversary is less than the classical complexity were known so far. Moreover, in [15, page 67] it was conjectured that these two complexities coincide. In this paper we exhibit a function f with a constant gap between these complexities (the half-duplex complexity with adversary is 5, and the classical complexity is 6). Also we exhibit a family of partial function g_n with a logarithmic gap between these complexities. More specifically, the function g_n is defined on length- n strings over a fixed alphabet, its half-duplex complexity with adversary is at most n while the classical complexity is at least $n + \log_2 n$.

To prove our lower bounds, that is, the bound 6 for the classical complexity of f and the bound $n + \log_2 n$ for the classical complexity of g_n , we use some novel techniques. The first bound cannot be proven via the common techniques, which is based on partitions of the matrix of f into monochromatic rectangles. A matrix of a function $f : X \times Y \rightarrow Z$ is a matrix with $|X|$ rows numbered by elements of X and $|Y|$ columns numbered by elements of Y . Its element in x th row and y th column is equal to $f(x, y)$. A monochromatic rectangle

in that matrix is any subset of $X \times Y$ that has the form $A \times B$ and on which f is constant. One can show that any communication protocol of depth d computing f provides a partition of that matrix into at most 2^d monochromatic rectangles. Hence the classical complexity of f can be lower bounded by the binary logarithm of minimal possible number of rectangles in a partition of matrix of f into monochromatic rectangles. Usually, the lower bounds for the size of such partitions are obtained by “fooling sets”. A set $F \subset X \times Y$ is called *fooling* for f if every monochromatic rectangle can cover at most one pair in F . Obviously, in this case the matrix of f cannot be partitioned in less than $|F|$ monochromatic rectangles. So to prove that classical complexity of f is larger than d , it is enough to find a fooling set for f of size larger than 2^d .

However, the matrix of our function f , whose classical complexity is at least 6, *can* be partitioned into 2^5 monochromatic rectangles, thus this method fails to show that the classical complexity of f is at least 6. To prove the desired lower bound, we will use the following feature of partitions that are derived from communication protocols: every such partition can be obtained by a sequence of horizontal and vertical divisions. More precisely, we start with trivial partition $P = \{X \times Y\}$ and on each step we partition any rectangle from P either horizontally (a rectangle $A \times B$ is replaced by two rectangles $A_0 \times B$ and $A_1 \times B$), or vertically (a rectangle $A \times B$ is partitioned into $A \times B_0$ and $A \times B_1$). If initial protocol has depth d then any rectangle in the resulting partition is obtained from the initial rectangle by at most d such divisions.

We show that for every horizontal partition of the matrix of f into two sub-matrices V, W either U or V has a fooling set of size $17 > 2^4 = 16$. This is done using the following method: in the matrix of f we distinguish the so called “fooling rectangles”. They are pairwise disjoint and have the following feature: if we pick from every fooling rectangle any cell, then the resulting set of cells is a fooling set in the matrix. The number of fooling rectangles is 25, thus this implies only that the matrix has a fooling set of size $25 \leq 2^5$. We show however that for every horizontal partition of the matrix into two sub-matrices V, W one of V, W intersects at least 17 fooling rectangles. To show that, we consider the graph whose vertices are fooling rectangles and edges connect those rectangles R_1, R_2 that share a row. Then we prove that that graph has certain expanding property. More specifically, every subset of $25 - 16 = 9$ vertices in the graph has at least 17 neighbors. This implies the desired property. Indeed, let \mathcal{R} stand for the family of all fooling rectangles that share a row with V . If $|\mathcal{R}| \leq 16$, then its complement $\overline{\mathcal{R}}$ has at least 9 rectangles and hence at least 17 neighbors and all them share a row with W !

Then we prove similar statement for vertical partitions, using other fooling rectangles.

To prove the lower bound $n + \log_2 n$ for communication complexity of g_n , we generalize the notion of a monochromatic rectangle to partial functions. This time we call a rectangle $A \times B$ monochromatic, if f is constant in each row and in each column of $A \times B$ (in the points where it is defined). Note that if f is not total then it may still be non-constant in the entire rectangle $A \times B$. However, for total functions this definition coincides with the classical definition of monochromatic rectangles. The matrix of our function g_n can be *covered* by 2^n monochromatic rectangles. However, we show that every its *partition* into

monochromatic rectangles has at least $n2^n$ rectangles.

The next section contains main definitions. In Section 3 we study partial functions, and in Sections 4 and 5 we study total functions. Section 4 is a “warming up” section, we exhibit there an example of a total function with a gap between classical complexity and half-duplex complexity with the so called “honest” adversary (an honest adversary sends the same bits to both players in every silent round, those bits may depend on the round). In Section 5 we present our function f with a gap between classical and half-duplex complexities (6 vs. 5).

2 Preliminaries

Definition 1 ([16]). A (classical) communication protocol to compute a partial function $f : X \times Y \rightarrow Z$ is a finite rooted tree T each of whose internal nodes (including the root) has two children. Its leaves are labeled by elements of Z , and each internal node v (including the root) is labeled either by letter **A** and by some function from X to $\{0, 1\}$, or by letter **B** and by some function from Y to $\{0, 1\}$. Besides, one outgoing edge from v is labeled by 0 and the other one is labeled by 1.

A protocol of depth 3 to compute a function $f : \{0, 1\} \times \{0, 1\} \rightarrow \{1, 2, 3, 4\}$ is shown on Fig. 1.

Definition 2. The computation of a protocol for the input pair $(x, y) \in X \times Y$ runs as follows. At the start, the players, Alice who has x and Bob who has y , place their tokens on the root of the tree. Then they move them along edges in the following way. If the tokens are in an internal vertex u labeled by **A** and a function $h : X \rightarrow \{0, 1\}$, then they move the tokens along the edge labeled by $h(x)$. To this end Alice sends $h(x)$ to Bob letting him know where to move his token. If the tokens are in an internal vertex u labeled by **B** and a function $h : Y \rightarrow \{0, 1\}$, then the tokens move along the edge labeled by $h(y)$ (Bob sends $h(y)$ to Alice). Finally, if the tokens come to a leaf, then the label of that leaf is the result of the computation. The rules guarantee that their tokens are always in the same node, thus Alice and Bob output the same results.

Definition 3. The sequence of the sent bits (= the leaf to which the token comes) is called the *transcript* of the communication. A communication protocol computes a partial function $f : X \times Y \rightarrow Z$, if for all input pairs (x, y) from the domain of f the result of the computation is equal to $f(x, y)$. The minimal depth of a communication protocol to compute f is called the *communication complexity* of f .

In the sequel we use the following well known facts (see for example [14]) about communication protocols to compute total functions:

- Let Π be a communication protocol. For every node u of Π the set of the input pairs (x, y) such that the token comes through u is a (combinatorial) rectangle, that is, it has the form $A_u \times B_u$ for some $A_u \subset X$ and $B_u \subset Y$. If u_0 and u_1 are children of u and u is labeled by **A** then $A_u = A_{u_0} \cup A_{u_1}$ and $B_u = B_{u_0} = B_{u_1}$. In this case we say that rectangles $A_{u_0} \times B_{u_0}$ and $A_{u_1} \times B_{u_1}$ are obtained from $A_u \times B_u$ by a horizontal

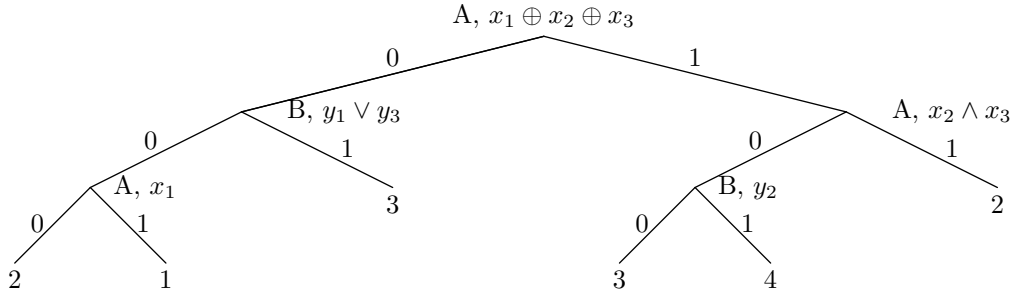


Figure 1: A protocol of depth 3 to compute a function $f : \{0, 1\}^3 \times \{0, 1\}^3 \rightarrow \{1, 2, 3, 4\}$. Each internal node is labeled by a letter indicating the turn to move and by a function computing the bit to send. For instance, if Alice has the string $x = 010$ and Bob the string $y = 110$, then in the first round Alice sends 1, in the second round Alice sends 0 and in the third round Bob sends 1. Then both parties output 4.

division. If u is labeled by B then $B_u = B_{u_0} \cup B_{u_1}$ and $A_u = A_{u_0} = A_{u_1}$. In this case we say that $A_{u_0} \times B_{u_0}$ and $A_{u_1} \times B_{u_1}$ are obtained from $A_u \times B_u$ by a vertical division.

- If f is constant in a rectangle R , then R is called a *monochromatic rectangle* for f .
- If a protocol Π computes a total function f , then the rectangle corresponding to any leaf of the protocol, is monochromatic for f . Hence to each protocol computing f we can assign a partition of $X \times Y$ into monochromatic rectangles. That partition has at most $2^{\text{depth of the protocol}}$ rectangles. Thus communication complexity is at least the binary logarithm of the minimal size of the partition of $X \times Y$ into monochromatic rectangles.
- A set $F \subset X \times Y$ is called a *fooling set* for function $f : X \times Y \rightarrow Z$, if for all different pairs $(x, y), (u, v) \in F$ not all values $f(x, y), f(u, v), f(x, v), f(u, y)$ are equal. In this case every partition of $X \times Y$ into monochromatic rectangles has at least $|F|$ rectangles.

The definition of a half-duplex communication protocol is more complicated.

Definition 4 ([4]). A half-duplex communication protocol to compute a partial function $f : X \times Y \rightarrow Z$ is a pair of rooted trees T_A, T_B . Each internal node of both trees has 4 children. Each internal vertex v of T_A is labeled by a function from X into the 3-element set consisting of *actions*, “send 0”, “send 1”, “receive”. Edges outgoing from v are labeled by *events* “sent 0”, “sent 1”, “received 0”, “received 1”. Similarly, each internal vertex of T_B is labeled by a function from Y into the set of actions, and outgoing edges are labeled by events. The leaves of both trees are labeled by elements of Z .

The computation for an input pair $(x, y) \in X \times Y$ runs as follows. First we calculate for each internal vertex of both trees the action of the player applying the corresponding function to her/his input (x or y). Second, each player puts a token on the root of her/his tree and moves it according to the following rules. If the tokens are in the vertices u, v , and

the actions of the players in those nodes are a, b , respectively, then the tokens are moved as follows:

a	b	Alice's token moves along the edge labeled by the event	Bob's token moves along the edge labeled by the event
send i	receive	"sent i "	"received i "
receive	send j	"received j "	"sent j "
send i	send j	"sent i "	"sent j "
receive	receive	"received j "	"received i "

If both players chose to receive (the last row of the table) then the round is called "silent". In this case the received values i, j are chosen by an adversary who decides where the tokens are moved to.

The computation stops when one of the token reaches a leaf. The result of the computation is defined as follows. If the other token is not in a leaf then the result is undefined. If both tokens reach leaves but their labels are different, then the result is undefined as well. Finally, if those labels coincide, then that label is the result of the computation.

An example of a 1-round half-duplex protocol is shown on Fig. 2.

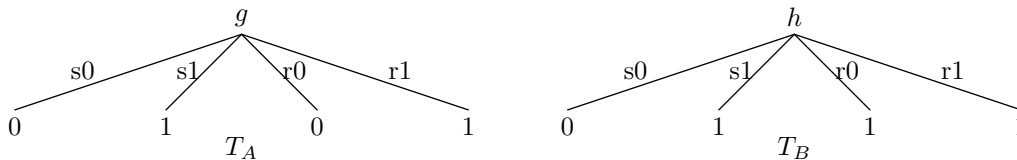


Figure 2: A 1-round half-duplex protocol. Here we assume that $X = Y = \{0, 1, 2\}$, $Z = \{0, 1\}$. The event "sent i " is abbreviated as si and similarly the event "receive i " is abbreviated as ri . The letter g denotes the mapping $0 \mapsto \text{send } 0, 1 \mapsto \text{send } 1, 2 \mapsto \text{receive}$, and h denotes the same mapping. This protocol computes the function $f : \{0, 1, 2\} \times \{0, 1, 2\} \rightarrow \{0, 1\}$ which is defined only on the pairs $(0, 0), (1, 1), (1, 2)$. On the first and the second pair both players send a bit which is lost, however they output the same result. On the third pair Alice sends 1, which is received by Bob, who actually does not care, since he outputs 1 any way.

Definition 5. A half-duplex protocol computes a function f , if for all pairs (x, y) in its domain and for all choices of the received bits in silent rounds (in different silent rounds different pairs (i, j) can be chosen) the computation ends with the result $f(x, y)$. If $f(x, y)$ is undefined, then the computation can end with any result or without any result.

Definition 6. There is a natural variation of computation via a half-duplex protocol, in which the adversary sends the same bits to Bob and Alice in each silent round (that bit may depend on the round). Such an adversary is called *honest*. To distinguish honest adversaries from general type adversaries we will call the latter *malicious*.

To distinguish normal communication protocols and complexity from half-duplex ones, we will call the former *classical*.

Remark 1. Classical protocols can be viewed as a particular case of half-duplex protocols. To convert a classical protocol into a half-duplex one, we first transform its tree T as follows. For each internal v vertex of T we make two copies of the tree with the root in the left child of v and two copies of the tree with the root in the right child of v . The edge going to the first copy of the left child is labeled by the event “sent 0”, and the edge going to the second copy of the left child is labeled by the event “received 0”, and similarly for the right child. This transformation should be applied to all internal nodes in any order. The trees T_A, T_B are equal to the resulting trees. If an internal node v is labeled by \mathbf{A}, h in T , then in the Alice’s tree T_A all copies of v are marked by the function that maps x to the action “send $h(x)$ ”, and in Bob’s tree T_B — by the constant function “receive”. Similarly, if v is labeled by \mathbf{B}, h in T , then in Bob’s tree all copies of v are labeled by the function mapping y to “send $h(y)$ ”, and in Alice’s tree — by the constant function “receive”. By construction, in the computation of the resulting protocol, there are no silent and spent rounds.

Obviously, half-duplex complexity with honest adversary is less than or equal to the half-duplex complexity with malicious adversary, and the latter is less than or equal to the classical communication complexity. One can also show that half-duplex complexity with both adversaries is larger than or equal to the half of the classical complexity [4]¹.

3 A separation of half-duplex and classical communication complexities for partial functions

3.1 An example

Lemma 1. *There is a partial function g whose half-duplex complexity is 1 and classical complexity is at least 2.*

Proof. Let $X = Y = \{0, 1, \mathbf{r}\}$, $Z = \{0\mathbf{r}, 1\mathbf{r}, \mathbf{r}0, \mathbf{r}1\}$ and the partial function $g : X \times Y \rightarrow Z$ is defined by the following table:

	0	1	\mathbf{r}
0			0 \mathbf{r}
1			1 \mathbf{r}
\mathbf{r}	$\mathbf{r}0$	$\mathbf{r}1$	

Here rows are labeled by Alice’s inputs x and columns by Bob’s inputs y . The entry of the table in row x and column y is blank, if $g(x, y)$ is undefined, and is $g(x, y)$ otherwise.

The classical complexity of g is at least 2: in any protocol of depth 1 either Alice sends a bit (and hence Alice outputs the same result for input pairs $(\mathbf{r}, 0)$ and $(\mathbf{r}, 1)$), or Bob sends a bit, and hence Bob outputs the same results for the input pairs $(0, \mathbf{r})$ and $(1, \mathbf{r})$.

¹Actually, in [4] this was shown for half-duplex complexity with zero in place of the half-duplex complexity with honest adversary. Obviously, the former one is at most the latter one.

The half-duplex complexity of g is at most 1. The protocol is the following: Alice on the input 0 sends 0 and outputs $0\mathbf{r}$, and on the input 1, she sends 1 and outputs $1\mathbf{r}$. Finally, on the input \mathbf{r} she receives and after receiving z she outputs $\mathbf{r}z$. Bob acts in a similar way: on the input 0 he sends 0 and outputs $\mathbf{r}0$, on the input 1 he sends 1 and outputs $\mathbf{r}1$, and on the input \mathbf{r} he receives and after receiving z he outputs $z\mathbf{r}$.

Let us show that this protocol indeed computes g . Assume that Alice has 0 and Bob has \mathbf{r} . Then Alice sends 0 to Bob, Bob receives 0 and both output $0\mathbf{r}$. A similar analysis works for three other input pairs where the function is defined, $(1, \mathbf{r})$, $(\mathbf{r}, 0)$, and $(\mathbf{r}, 1)$.

For the pairs where g is undefined (e.g. $(0, 1)$) Alice and Bob may output different results, but we do not care. \square

Consider the following generalization of this function, which provides a linear gap between half-duplex and classical complexities. Let $X = Y = \{0, 1, \mathbf{r}\}^n$, $Z = \{0\mathbf{r}, 1\mathbf{r}, \mathbf{r}0, \mathbf{r}1\}^n$ and define g as follows: $g(x, y)$ is defined if $x_i = \mathbf{r} \wedge y_i \neq \mathbf{r}$ or $x_i \neq \mathbf{r} \wedge y_i = \mathbf{r}$ for all i . In this case the i -th symbol of $g(x, y)$ equals $\mathbf{r}y_i$ in the first case and equals $x_i\mathbf{r}$ in the second case. That is, if Alice's and Bob's inputs are "consistent" (in each position one party has \mathbf{r} and the other has 0 or 1), then the function is defined and otherwise it is not.

Theorem 1. (a) *The half-duplex complexity of g is at most n .* (b) *The classical complexity of g is at least $2n$*

Proof. (a) In i th round, independently on what has happened in previous rounds, each party receives if her/his i th symbol is \mathbf{r} , otherwise she/he sends i th symbol of the input.

As i th symbol of the output Alice takes $x_i\mathbf{r}$, if in i th round she sent a bit (equal to x_i), and $\mathbf{r}y_i$, if she received y_i . Bob acts in a similar way.

Let us show that this protocol indeed computes g . Assume that Alice gets x , Bob gets y where x and y are consistent. Then in i th round either Alice sends x_i and Bob receives x_i and then both take $x_i\mathbf{r}$ as i th symbol of the output, or Bob sends y_i and Alice receives y_i and then both take $\mathbf{r}y_i$ as i th symbol of the output. Thus the i th symbol of the output of both parties is equal to the i th symbol of $g(x, y)$ for all i .

(b) Note that the function g has 4^n possible values hence any classical communication protocol computing g must have at least 4^n leaves. Thus its classical communication complexity is at least $\log_2 4^n = 2n$. \square

3.2 Local communication complexity

There is a way to compute the function g from the previous section by communicating at most $n + \lceil \log_2(n + 1) \rceil$ bits (where n is the length n of the inputs):

1. Alice computes the number k of symbols \mathbf{r} in her input.
2. She then sends k to Bob in $\lceil \log_2(n + 1) \rceil$ bits. Note that $k \in \{0, 1, \dots, n\}$, thus k indeed can be encoded in this many bits.

3. Bob, after receiving the number k , halts with any result if the number of letters \mathbf{r} in his input is different from $n - k$.
4. Then Alice sends $(n - k)$ bits equal to symbols 0 and 1 in her input keeping their order.
5. Unless he has halted, Bob receives $n - k$ bits sent by Alice and then sends k bits equal to symbols 0 and 1 in his input, keeping their order.
6. Alice, after receiving a string z , outputs the string a of length n defined as follows. If $x_i \neq \mathbf{r}$, then $a_i = x_i \mathbf{r}$. The remaining symbols of a have the form $\mathbf{r}z_j$ where the bits of z are inserted into a in the same order in which they are arranged in z .
7. Bob computes his output string in a similar way.

Why this protocol does not contradict to the lower bound $2n$ for the communication complexity of the function g ? This is because this protocol is not a communication protocol according to Definition 4. Indeed, Alice and Bob form their outputs based not only on the transcript of the computation but also on their inputs. As we will see later, for total functions, this possibility does not yield anything new. However, for partial functions this possibility can decrease communication complexity, as our example shows. A simpler example of this phenomenon will be given later. By this reason for partial functions a more adequate communication model should allow players to use their inputs when computing the output.

As far as we know, such a communication model appeared quite recently in [13, page 58]. Let us define it more formally.

Definition 7 ([13]). A local communication protocol for a function $f : X \times Y \rightarrow Z$ is a communication protocol in which each leaf is labeled by a pair of functions $X \rightarrow Z$ and $Y \rightarrow Z$. If a computation on an input pair (x, y) has ended in a leaf labeled by a pair of functions (p, q) , then the result of computation is defined as $p(x)$ provided $p(x) = q(y)$, and is undefined otherwise. A local communication protocol computes a partial function $f : X \times Y \rightarrow Z$ if for all input pairs (x, y) from its domain the protocol outputs $f(x, y)$ (the result can be both defined and undefined for the pairs outside the domain of the function). Local communication complexity of a function is the minimal depth of a local communication protocol computing the function.

Local half-duplex communication protocols are defined in a similar way. When we want to distinguish local protocols from normal ones, we call the latter ones *global*.

3.3 An example

Consider the partial function $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ defined as follows:

$$f(x, y) = \begin{cases} x, & \text{if } x = y \\ \text{undefined} & \text{if } x \neq y \end{cases}$$

Since f takes 2^n different values, any global (classical) protocol for it has at least 2^n leaves and hence its depth is at least n . Thus the global (classical) complexity of f is at least n . On the other hand the local (classical) complexity of f is 0: label the root of the tree with pair (h, h) , where h is the identity function $h(x) = x$ (each player just outputs her/his input).

It is important in this example that the function is not total. For total functions there is no such example both in classical and half-duplex models.

3.4 Local and global complexity of total functions

Lemma 2. *For classical communication complexity: the local complexity of any total function is equal to its global complexity.*

Proof. Assume that a local (classical) communication protocol for f is given and let l be any its leaf. Let $R_l = A_l \times B_l$ denote the rectangle consisting of all input pairs (x, y) for which the computation reaches l . We claim that f is constant on R_l .

Indeed, let p_l, q_l denote the functions which label the leaf l . Then for all $(x, y) \in R_l$ and $(x', y') \in R_l$ we have

$$f(x, y) = q_l(y) = p_l(x') = f(x', y').$$

Here the first equality holds, since Bob outputs the correct result on the pair x, y , the second equality holds, since Alice and Bob output same results on the pair x', y , which also belongs to R_l , and the third equality holds, since Alice outputs the correct result on the pair x', y' , which also belongs to R_l .

Hence any local protocol to compute any total function can be converted into a classical protocol of the same depth just by changing the label (p_l, q_l) to the constant value of f on R_l . \square

A similar fact holds for half-duplex complexity.

Lemma 3. *For half-duplex communication complexity: the local complexity of any total function is equal to its global complexity.*

Proof. The proof is similar to the proof of the previous lemma. However for half-duplex protocols the connection between rectangles and leaves in the protocol trees is more complicated. Therefore we will not establish this connection explicitly. Instead we will use a direct argument.

Again we show that any local protocol to compute a total function is essentially global. More precisely, all computations ending in a leaf output the same result. Thus we can replace the function labeling each leaf in T_A by the constant it outputs and then we can make the same thing for T_B . In this way we transform a local protocol to the global one.

Consider a local communication protocol (T_A, T_B) to compute a total function f . Assume that for an input pair (x, y) the computation can end in the leaves a, b (we use the word “can”, since now the leaves depend on the choices of the adversary). Assume also that for another input pair (x', y') the computation of Alice can end in the same leaf a . Let us show that for the input pair (x', y) the computation can also end in leaves (a, b) .

Let a_i and b_i denote i th vertices on the paths from the roots to a and b , respectively. Since Alice's computation on both inputs (x, y) and (x', y') can end in the leaf a , for all i Alice's action in the vertex a_i for inputs x and x' is the same. Thus, if the adversary's action in the computation on the input pair (x', y) is identical to that on the input (x, y) , when the computation ends in (a, b) , then both players in each round perform the same actions on inputs (x, y) and (x', y) and all events are identical as well. Hence the computation on the input (x', y) can also end in the same leaves a, b .

Let the leaf a is labeled by the function p and the leaf b by the function q . Then $p(x) = f(x, y) = q(y) = f(x', y) = p(x')$. (It is important here that f is a total function, as otherwise $f(x', y)$ can be undefined and hence $q(y)$ may differ from $p(x')$.)

Thus each leaf of T_A is labeled by a function that is constant on all x 's for which the computation can end in that leaf for some y . \square

3.5 Local classical complexity of the function g

For partial functions, local protocols seem to be a more natural communication model than the global ones. In the Introduction, before we have given formal definition, we talked about the lower bound $n + \log_2 n$ for classical communication complexity of the function g_n . We meant there the local complexity. Now we will prove this statement.

Theorem 2. (a) *The local classical complexity of g is at most $n + \lceil \log_2(n + 1) \rceil$.*

(b) *The local classical complexity of g is at least $n + \lceil \log_2 n \rceil - \log_2 3 + o(1)$.*

Proof. (a) This statement was proved above. (b) This statement is more difficult to prove.

Lemma 4. *If the local classical complexity of a partial function g is at most c , then its matrix can be partitioned into at most 2^c rectangles with the following property: if inputs (x, y) and (x, y') are in the same rectangle, and g is defined on both of them, then $g(x, y) = g(x, y')$, and similarly, if inputs (x, y) and (x', y) are in the same rectangle, and g is defined on both of them, then $g(x, y) = g(x', y)$. (Rectangles with this property will be called monochromatic.)*

Proof. Leaves of classical communication protocol define matrix partition into rectangles. Each of these rectangles has the mentioned property: if a function p is written in Alice's leaf, and g is defined on (x, y) and (x, y') , then we necessarily have $g(x, y) = p(x) = g(x, y')$. Similarly for Bob.

If the depth of the protocol is c , then the number of leaves is at most 2^c , and its rectangles form the required partition. \square

We will call any input of the form $(\{0, 1\}^k \mathbf{r}^{n-k}, \mathbf{r}^m \{0, 1\}^{n-m})$ *simple*. We denote by $[x]$ the total number of symbols 0 and 1 in x . We color in green all simple inputs (x, y) with $[x] + [y] = n$. The function g is defined on such inputs. We color in blue simple inputs (x, y) such that $[x] + [y] < n$. The function g is not defined on them (see Fig. 3). The number of green inputs is $(n + 1) \cdot 2^n$. The number of blue inputs is

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-i-1} 2^{i+j} = \sum_{i=0}^{n-1} 2^i \cdot (2^{n-i} - 1) = n \cdot 2^n - (2^n - 1) = (n - 1) \cdot 2^n + 1$$

	0...0	...	1...1	r0...0	...	r1...1	rr0...0	...	rr1...1	...	r...r
r...r											
0r...r											
1r...r											
00r...r											
01r...r											
10r...r											
11r...r											
...											
0...0											
...											
1...1											

Figure 3: The part of matrix of function g consisting of simple inputs.

Assume that classical complexity of g is at most c . Apply the above lemma to a half-duplex protocol of depth c to compute g and consider the corresponding partition into at most $L = 2^c$ monochromatic rectangles.

Let (x, y) and (u, v) be different green inputs with $[x] = [u]$. Then g is defined on both (x, v) and (u, y) and not all values $g(x, y)$, $g(x, v)$, $g(u, y)$, $g(u, v)$ are the same. Thus, (x, y) and (u, v) are in different rectangles of the partition.

Note that, if a rectangle contains k green inputs (x_i, y_i) , $[x_1] < [x_2] < \dots < [x_k]$, then it also contains $\frac{k \cdot (k-1)}{2}$ blue inputs (x_i, y_j) , $i < j$. Let k_i be the number of green inputs in the i th rectangle, then i th rectangle also contains at least $\frac{k_i \cdot (k_i-1)}{2}$ blue inputs. Then $\sum_{i=1}^L k_i = (n+1) \cdot 2^n$. On the other hand, $\sum_{i=1}^L \frac{k_i \cdot (k_i-1)}{2} \leq (n-1) \cdot 2^n + 1$ as rectangles are disjoint. Re-writing the second inequality, we get

$$\begin{aligned} \sum_{i=1}^L (k_i^2 - k_i) &\leq (2n-2) \cdot 2^n + 2 \\ \sum_{i=1}^L k_i^2 &\leq (2n-2) \cdot 2^n + 2 + (n+1) \cdot 2^n \\ \sum_{i=1}^L k_i^2 &\leq (3n-1) \cdot 2^n + 2. \end{aligned}$$

By Cauchy–Schwarz inequality, $\left(\sum_{i=1}^L k_i\right)^2 \leq L \cdot \sum_{i=1}^L k_i^2$. Therefore,

$$\begin{aligned} (n+1)^2 \cdot 2^{2n} &\leq L \cdot ((3n-1) \cdot 2^n + 2) \Rightarrow \\ L &\geq \frac{(n+1)^2 \cdot 2^{2n}}{(3n-1) \cdot 2^n + 2} \Rightarrow \\ L &\geq \frac{n \cdot 2^n}{3} \cdot (1 + o(1)) \Rightarrow \\ c = \log_2 L &\geq n + \log_2 n - \log_2 3 + o(1) \quad \square \end{aligned}$$

So, the function g is an example of a partial function with logarithmic gap between local classical complexity and half-duplex complexity with adversary. We would also like to find an example of a total function with the same gap. Unfortunately, we could not find any such example. In the rest of the paper we present an example of a total function with a constant gap — our function has half-duplex complexity 5 and classical complexity 6. We will start with a simpler example of a constant gap between classical complexity and half-duplex complexity with honest adversary.

4 A separation of half-duplex complexity with honest adversary from classical complexity for total functions

Theorem 3. *There is a total function with classical complexity 4 and half-duplex complexity with honest adversary at most 3.*

Proof. Let $X = Y = \{\mathbf{r}, 00, 01, 10, 11\}$ and $Z = \{0, 1, 2\}$. First we design a 3 round half-duplex protocol, computing a total function $f : X \times Y \rightarrow Z$ (against honest adversary) and then we prove that its classical complexity is at least 4.

In that protocol only the first round is not classical. In that round both players depending on their inputs choose one of the three actions. That action is determined by the first symbol of the input: \mathbf{r} means “receive”, and 0,1 mean “send 0,1”, respectively. In the second round Bob sends a bit and Alice receives, and in the third round vice versa. In those two rounds the players send the bit that was received in the first round provided they chose to receive in the first round. Otherwise players send the second bits of their inputs.

The result of the protocol is a function of bits sent (and received) in the second and third rounds (we will call it *the 2–3-transcript* in the sequel). Since those rounds are classical, that guarantees that Alice and Bob output the same result, whatever happens in the first round. Let us see what is the 2–3-transcript equal to. If Bob’s input is \mathbf{r} and Alice’s input is 0, 1, then 2–3-transcript coincides with Alice’s input. Similarly, if Bob’s input is 0,1 and Alice’s input is \mathbf{r} , then 2–3-transcript coincides with Bob’s input reversed. If both Alice’s and Bob’s inputs are 0,1, then 2–3-transcript is formed from the second bits of Bob and

Alice. Finally, if both Alice’s and Bob’s inputs are equal to \mathbf{r} , then 2–3-transcript consists of bits chosen by the adversary. Since we assume honest adversary, it is equal either to 00, or to 11. The 2–3-transcript is shown in the following table where Alice’s input indicates the row and Bob’s input the column:

	\mathbf{r}	00	01	10	11
\mathbf{r}	00 or 11	00	10	01	11
00	00	00	10	00	10
01	01	01	11	01	11
10	10	00	10	00	10
11	11	01	11	01	11

We can see that if the output function takes the same values on the arguments 00 and 11, then the protocol computes a total function. We will define the output function as the 2–3-transcript with identified 00 and 11. Let us represent 2–3-transcripts by natural numbers. In this way we get the following function:

	\mathbf{r}	00	01	10	11
\mathbf{r}	0	0	2	1	0
00	0	0	2	0	2
01	1	1	0	1	0
10	2	0	2	0	2
11	0	1	0	1	0

The key point is the following: since the adversary is honest, we need to identify only 00 and 11. For a malicious adversary, we would need to identify all four 2–3-transcripts and the function would become constant.

To show that the classical complexity of the defined function is at least 4, we will find a fooling set of size $10 > 2^3$. The members of that set are colored magenta in the matrix:

$$\begin{pmatrix} 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 0 & 2 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad \square$$

Recently T. Kuptsov [8] has shown that the half-duplex complexity with malicious adversary of this function is the same as its classical complexity, that is, is equal to 4. Thus this function separates half-duplex complexity with adversary from half-duplex complexity with honest adversary. This also means that, to separate half-duplex complexity with malicious adversary from classical complexity, we have to look for another example.

Since fooling sets have the direct product property, this example provides a sequence of functions with a linear gap between half-duplex complexity with honest adversary and classical complexity. More specifically, let the function $f^n : X^n \times Y^n \rightarrow Z^n$ be the direct

product of n copies of f , that is, $f^n(x_1x_2\dots x_n, y_1y_2\dots y_n) = f(x_1, y_1)f(x_2, y_2)\dots f(x_n, y_n)$. Here X, Y, Z, f are sets and the function from the proof of the previous theorem. The arguments of f are length- n strings over the alphabets X, Y and its value is a length- n string over Z .

Theorem 4. *The classical complexity of f^n is at least $\log_2(10)n$ while its half-duplex complexity with honest adversary is at most $3n$.*

Proof. The second statement is obvious, since we can just apply n times the protocol from the previous theorem to compute n copies of f .

The lower bound for classical complexity is proven as follows. Recall that the function f has a fooling set F of size 10. We claim that its Cartesian power F^n is a fooling set for f^n .

For the sake of contradiction assume that that different pairs (x, y) and (a, b) are in F^n and are in a monochromatic rectangle R . Let $x = x_1\dots x_n, y = y_1\dots y_n, a = a_1\dots a_n, b = b_1\dots b_n$. Then for some i the pairs (x_i, y_i) and (a_i, b_i) differ. W.l.o.g. assume that $i = 1$. Since F is a fooling set for f , the rectangle $\{x_1, a_1\} \times \{y_1, b_1\}$ is not monochromatic. That is, this rectangle has two pairs on which f has different values.

There are six possible cases. Case 1: Assume first that $f(x_1, y_1) \neq f(a_1, b_1)$. Then $f(x, y) \neq f(a, b)$, which is a contradiction, as both pairs $(x, y), (a, b)$ are in R . Case 2: Now assume that $f(x_1, y_1) \neq f(x_1, b_1)$. Then $f(x, y) \neq f(x, b)$, which is again a contradiction, as both pairs $(x, y), (x, b)$ are in R . The remaining cases are similar to these two cases.

Since $|F^n| = 10^n$, the classical complexity of f^n is at least $\log_2 |F^n| = n \log_2(10)$. \square

5 A separation of half-duplex complexity with malicious adversary from classical complexity for total functions

Theorem 5. *There is a total function with classical complexity 6 and half-duplex complexity (with malicious adversary) at most 5.*

The proof of this theorem is similar to that of the previous one but is much more complicated. The proof is divided into three sections. In the first section we define a 5 round half-duplex protocol Π to compute a total function. In the second and third sections we prove that the classical complexity of the computed function is at least 6.

5.1 The half-duplex protocol

5.1.1 A general protocol

Our general plan is the following. The first round of the protocol Π is not classical and all the remaining 4 rounds are classical, in those rounds the players receive bits in the following sequence: Alice, Bob, Alice, Bob. The result of the protocol is a function of the sequence of bits sent in rounds 2–5 (that sequence will be called the *2–5-transcript*). The inputs of the

players consist of one or two symbols. The first symbol of player’s input is \mathbf{r} , 0 or 1 and has the same meaning as before: it indicates what to do in the first round. The remaining symbol (if any) indicates how to act in the remaining 4 rounds.

More specifically, Bob’s input is either \mathbf{r} , or has the form 0φ or 1φ where the range of φ will be defined later. Alice’s input is either $\mathbf{r}\eta$, or has the form 0ψ or 1ψ where the ranges of η, ψ will be defined later. The protocol is the following:

1. If Bob’s input is \mathbf{r} , then in the first round Bob receives (let m denote the received bit), then he sends m (the bit he just received), then receives again, then again he sends m and then again receives.
2. If Alice’s input is $\mathbf{r}\eta$, then she receives in the first round (let m denote the received bit), then she receives again (denote the received bit by i), then sends m (the bit from the first round), then again receives (let it be k), then again sends m if $i = k$ and otherwise (if $i \neq k$) in the last round Alice sends a bit that depends on η, i, k (how that bits depends on them, will be defined later).
3. If Bob’s input is $i\varphi$, $i = 0, 1$, then Bob sends i in the first round, then he sends a bit that depends on φ , then receives (let it be j), then sends a bit that depends on j, φ , then again receives a bit.
4. If Alice’s input is $j\psi$, $j = 0, 1$, then Alice sends j in the first round, then she receives (let it be i), then sends a bit that depends on i, ψ , then receives (let it be k) and finally sends a bit that depends on i, k, ψ .

According to this protocol, on the input pair $(\mathbf{r}\eta, \mathbf{r})$ the events of the players are the following (assuming that the adversary sends j to Alice and i to Bob):

Round	1	2	3	4	5
Bob’s event	“received i ”	“sent i ”	“received j ”	“sent i ”	“received j ”
Alice’s event	“received j ”	“received i ”	“sent j ”	“received i ”	“sent j ”

So far, this is only a general plan of protocol’s construction. We have yet to define the ranges of η, φ, ψ and to define the dependencies mentioned above. However we can already see that, if the first round happens to be silent, then the 2–5-transcript of the protocol is one of these four sequences: 0000, 0101, 1010 and 1111. We will define Alice’s and Bob’s output be 0, if the 2–5-transcript of the protocol is one of these four sequences, and to the 2–5-transcript otherwise. Such a definition guarantees that even for malicious adversary the computed function is total. We have identified only a quarter of 2–5-transcripts, which leaves an opportunity for the computed function be non-trivial.

5.1.2 The first realization of the plan

The simplest realization of this plan is the following: let $\eta \in \{0, 1\}$ and $\varphi, \psi \in \{0, 1\}^2$ and let the protocol run as follows:

1. If Bob's input is \mathbf{r} , then in the first round Bob receives (let m denote the received bit), then he sends m , then receives again, then again he sends m and then again receives.
2. If Alice's input is $\mathbf{r}\eta$, then she receives in the first round (let m denote the received bit), then she receives again (let it be i), then sends m , then again receives (let it be k), then again sends m , if $i = k$, and η otherwise.
3. If Bob's input is $i\varphi$, $i = 0, 1$, $\varphi = \varphi_1\varphi_2$, then Bob sends i in the first round, then he sends φ_1 , then receives, then sends φ_2 , then again receives.
4. If Alice's input is $j\psi$, $j = 0, 1$, $\psi = \psi_1\psi_2$, then Alice sends j in the first round, then she receives, then sends ψ_1 , then receives and finally sends ψ_2 .

As a result, Alice and Bob produce the following 2–5 transcript (each transcript is represented by a natural number using its binary expansion):

Bob's input \rightarrow	\mathbf{r}	000	001	010	011	100	101	110	111
Alice's input \downarrow									
$\mathbf{r}0$	0, 5, 10, 15	0	2	8	10	5	6	12	15
$\mathbf{r}1$	0, 5, 10, 15	0	3	9	10	5	7	13	15
000	0	0	2	8	10	0	2	8	10
001	1	1	3	9	11	1	3	9	11
010	4	4	6	12	14	4	6	12	14
011	5	5	7	13	15	5	7	13	15
100	10	0	2	8	10	0	2	8	10
101	11	1	3	9	11	1	3	9	11
110	14	4	6	12	14	4	6	12	14
111	15	5	7	13	15	5	7	13	15

Thus our protocol computes the function with the following matrix (zeros have colors indicating the respective 2–5 transcripts):

0	0	2	8	0	0	6	12	0
0	0	3	9	0	0	7	13	0
0	0	2	8	0	0	2	8	0
1	1	3	9	11	1	3	9	11
4	4	6	12	14	4	6	12	14
0	0	7	13	0	0	7	13	0
0	0	2	8	0	0	2	8	0
11	1	3	9	11	1	3	9	11
14	4	6	12	14	4	6	12	14
0	0	7	13	0	0	7	13	0

By construction the half-duplex complexity of this function is at most 5. Unfortunately, its classical complexity is also at most 5. To show this, let us note that this matrix has 7

different columns (the 2nd column coincides with the 6th one and the 5th column with the 9th one). Each column of this matrix, except the first one, has at most 4 different numbers. Therefore we can compute this function in 5 rounds using the following protocol. First Bob lets Alice know the entire column, corresponding to his input. If his input is not the first column, then he sends 3 bits, and otherwise he sends only 2 bits. In the first case Alice in two rounds sends Bob the value of the function. In the second case she does that in 3 rounds.

5.1.3 The second (and final) realization of the plan

To increase the classical complexity of this function, let us use most general dependencies in our protocol. Namely, let φ range over all mappings that determine which bits sends Bob in the second and fourth rounds in all possible situations. In other words, φ is a function with the domain $\{\Lambda, 0, 1\}$ and values 0,1 where $\varphi(\Lambda)$ is the bit sent in the second round and $\varphi(j)$ is the bit sent in the fourth round after receiving j in the third round. Thus Bob has $1 + 2 \times 8 = 17$ different inputs.

Similarly, η ranges over all mappings that determine which bits sends Alice in all possible situations (assuming that she receives in the first round). The domain of η is the set $\{001, 101, 010, 110\}$ where $\eta(mik)$ is the bit which Alice sends in the fifth round provided she receives m, i, k in the first, second and fourth rounds, respectively.

The domain of ψ is $\{0, 1, 00, 01, 10, 11\}$ where $\psi(i)$ is the bit Alice sends in the third round provided she received i in the second round and $\psi(ik)$ is the bit Alice sends in the fifth round provided she received i, k in the second and fourth rounds, respectively. Thus Alice has $2 \times 2^6 + 2^4 = 144$ different inputs.

Now our half-duplex protocol is well defined. By construction it computes a total function and we denote that function by U . We will prove that the classical communication complexity of U is 6.

The matrix of U is very large, its size is 144×17 , and therefore that proof cannot be visualized. Because of that, we found two moderately small sub-functions of U with the same classical complexity. By a sub-function here we mean a restriction of U on a set of the form $X' \times Y'$ where $X' \subset X, Y' \subset Y$. In terms of the matrix of the function, this means deleting some rows and columns from the matrix. Note that restricting the function cannot increase its half-duplex complexity, thus the half-duplex complexity of any sub-function of U is at most 5.

5.1.4 Sub-functions of U with communication complexity 6: the function S

The smallest sub-function of U with classical complexity 6 we managed to find has the matrix shown on Fig. 4. We denote this sub-function by S . Communication complexity of S was found on a computer in 4 minutes using the dynamic programming. Since this computation cannot be performed by hand, we continued to look for a sub-function of U with a moderately small domain for which we can prove by hand the lower bound 6 for the classical complexity.

0	0	0	2	9	0	0	0	7	7	12	12	0
0	0	0	3	8	0	0	0	7	7	13	13	0
0	0	0	2	8	0	0	0	6	6	13	13	0
0	0	7	7	13	13	0	0	7	7	13	13	0
0	0	6	6	12	12	14	0	6	6	12	12	14
4	4	6	6	12	12	14	4	6	6	12	12	14
4	4	6	6	9	11	11	4	6	6	9	11	11
1	1	1	3	9	0	0	1	1	3	9	0	0
0	0	0	2	8	0	0	0	0	2	8	0	0
0	0	7	7	13	13	0	0	7	7	13	13	0
14	0	6	6	12	12	14	0	6	6	12	12	14
14	4	6	6	12	12	14	4	6	6	12	12	14
11	4	6	6	9	11	11	4	6	6	9	11	11
0	1	1	3	9	0	0	1	1	3	9	0	0
0	0	0	2	8	0	0	0	0	2	8	0	0

Figure 4: The matrix of the function S . Its classical complexity is 6 and its half-duplex complexity is at most 5.

5.1.5 Sub-functions of U with communication complexity 6: the function M

The matrix of the function M we found is shown on Fig. 5. On Fig. 6 we indicated, on the left and on the top, Alice’s and Bob’s inputs. Contemplating this matrix, we can imagine how the matrix of U looks like. The domain of M is the smallest sub-domain of U for which our method to prove the lower bound 6 works. Later we will explain this in more detail.

Looking at Fig. 6 one can verify that each entry in the matrix is indeed the result of the above described half-duplex protocol Π on the corresponding input pair. Also one can verify that matrices on Fig. 5 and Fig. 6 are identical. This implies that the function with the matrix shown on Fig. 6 is a sub-function of U and hence its half-duplex complexity is at most 5. However both verifications are quite time consuming and therefore we will explain in the next section how to verify faster that its half-duplex complexity is at most 5.

5.1.6 A fast verification that the half-duplex complexity of the function with matrix on Fig. 5 is at most 5

We exhibit the partition the matrix from Fig. 5 into monochromatic rectangles corresponding to leaves of Alice’s and Bob’s trees (in the protocol Π). Such a partition is shown on Fig. 7. The double lines partition the matrix according to events happening in the first round. For example, the bottom right rectangle consists of all input pairs for which both Alice and Bob send 1, the upper right rectangle consists of all input pairs for which Alice receives 1 from Bob and the upper left rectangle of all input pairs for which both parties receive in the first round. These rectangles will be denoted by $P_{rr}, P_{r0}, P_{r1}, P_{0r}, P_{1r}, P_{00}, P_{01}, P_{10}, P_{11}$.

In Fig. 7, we have restored the 2–5-transcripts, that is, 0, 5, 10 and 15 are not yet

0	0	2	0	2	8	8	0	0	6	6	12	0	12	0
0	0	2	0	2	8	8	0	0	6	6	13	0	13	0
0	0	2	0	2	9	9	0	0	7	7	12	0	12	0
0	0	3	0	3	8	8	0	0	7	7	13	0	13	0
0	0	3	0	3	9	9	0	0	6	6	12	0	12	0
0	0	3	0	3	9	9	0	0	7	7	13	0	13	0
0	0	2	0	2	8	8	11	2	0	2	8	8	11	11
0	0	3	0	3	12	14	12	3	0	3	12	14	12	14
1	1	2	1	2	9	9	11	2	1	2	9	9	11	11
1	1	2	1	2	13	14	13	2	1	2	13	14	13	14
1	1	3	1	3	12	14	12	3	1	3	12	14	12	14
1	1	3	1	3	8	8	11	3	1	3	8	8	11	11
4	4	4	6	6	9	9	11	4	6	6	9	9	11	11
4	4	4	7	7	8	8	11	4	7	7	8	8	11	11
4	4	4	6	6	12	14	12	4	6	6	12	14	12	14
4	4	4	7	7	13	14	13	4	7	7	13	14	13	14
0	0	0	7	7	8	8	11	0	7	7	8	8	11	11
11	0	2	0	2	9	9	11	2	0	2	9	9	11	11
14	0	3	0	3	13	14	13	3	0	3	13	14	13	14
11	1	2	1	2	8	8	11	2	1	2	8	8	11	11
14	1	2	1	2	12	14	12	2	1	2	12	14	12	14
11	1	3	1	3	9	9	11	3	1	3	9	9	11	11
14	1	3	1	3	13	14	13	3	1	3	13	14	13	14
11	4	4	6	6	8	8	11	4	6	6	8	8	11	11
14	4	4	6	6	13	14	13	4	6	6	13	14	13	14
11	4	4	7	7	9	9	11	4	7	7	9	9	11	11
0	4	4	7	7	9	9	0	4	7	7	9	9	0	0
0	4	4	6	6	12	0	12	4	6	6	12	0	12	0
14	4	4	7	7	12	14	12	4	7	7	12	14	12	14

Figure 5: The matrix of the function M . Its classical communication complexity is 6, while its half-duplex complexity is at most 5.

						Arg. of func. $\varphi \downarrow$	The value of function $\varphi \downarrow$																
						Λ	0				1				0				1				
						0	0				1				0				1				
						1	0				1				0				1				
						Inp. B \rightarrow	r	0φ						1φ									
A. η						Inp. A \downarrow																	
V. η			001	010	101	110	$r\eta$	0	0	2	0	2	8	8	0	0	6	6	12	0	12	0	
			0	0	0	1		0	0	2	0	2	8	8	0	0	6	6	13	0	13	0	
			0	1	1	0		0	0	2	0	2	9	9	0	0	7	7	12	0	12	0	
			1	0	1	1		0	0	0	3	0	3	8	8	0	0	7	7	13	0	13	0
			1	1	0	0		0	0	0	3	0	3	9	9	0	0	6	6	12	0	12	0
		1	1	1	1	1	0	0	3	0	3	9	9	0	0	7	7	13	0	13	0		
A. ψ	0	1	00	01	10	11																	
V. ψ	0	0	0	0	0	1	0ψ	0	0	2	0	2	8	8	11	2	0	2	8	8	11	11	
	0	1	0	1	0	0		0	0	3	0	3	12	14	12	3	0	3	12	14	12	14	
	0	0	1	0	1	1		1	1	2	1	2	9	9	11	2	1	2	9	9	11	11	
	0	1	1	0	1	0		1	1	2	1	2	13	14	13	2	1	2	13	14	13	14	
	0	1	1	1	0	0		1	1	3	1	3	12	14	12	3	1	3	12	14	12	14	
	0	0	1	1	0	1		1	1	3	1	3	8	8	11	3	1	3	8	8	11	11	
	1	0	0	0	1	1		4	4	4	6	6	9	9	11	4	6	6	9	9	11	11	
	1	0	0	1	0	1		4	4	4	7	7	8	8	11	4	7	7	8	8	11	11	
	1	1	0	0	0	0		4	4	4	6	6	12	14	12	4	6	6	12	14	12	14	
	1	1	0	1	1	0		4	4	4	7	7	13	14	13	4	7	7	13	14	13	14	
1	0	1	1	0	1	0	0	0	7	7	8	8	11	0	7	7	8	8	11	11			
V. ψ	0	0	0	0	1	1	1ψ	11	0	2	0	2	9	9	11	2	0	2	9	9	11	11	
	0	1	0	1	1	0		14	0	3	0	3	13	14	13	3	0	3	13	14	13	14	
	0	0	1	0	0	1		11	1	2	1	2	8	8	11	2	1	2	8	8	11	11	
	0	1	1	0	0	0		14	1	2	1	2	12	14	12	2	1	2	12	14	12	14	
	0	0	1	1	1	1		11	1	3	1	3	9	9	11	3	1	3	9	9	11	11	
	0	1	1	1	1	0		14	1	3	1	3	13	14	13	3	1	3	13	14	13	14	
	1	0	0	0	0	1		11	4	4	6	6	8	8	11	4	6	6	8	8	11	11	
	1	1	0	0	1	0		14	4	4	6	6	13	14	13	4	6	6	13	14	13	14	
	1	0	0	1	1	1		11	4	4	7	7	9	9	11	4	7	7	9	9	11	11	
	1	0	0	1	1	0		0	4	4	7	7	9	9	0	4	7	7	9	9	0	0	
1	1	0	0	0	1	0	4	4	6	6	12	0	12	4	6	6	12	0	12	0			
1	1	0	1	0	0	14	4	4	7	7	12	14	12	4	7	7	12	14	12	14			

Figure 6: The matrix of function M with an indication of Alice’s and Bob’s inputs. “A.” means “Argument of”, “V.” means “Value of”, “Inp. A” means “Input of Alice” and “Inp. B” means “Input of Bob”.

identified. The only exception is the rectangle P_{rr} , in which all 2–5-transcripts 0, 5, 10 and 15 can appear. The number in each cell indicates in which part (0-part or 1-part) the respective cell falls in each round. Namely, its leading bit is equal to the bit sent by Bob in the second round, the second bit is equal to the bit sent by Alice in the third round, and so on. In the second round all numbers are partitioned into small ones (less than 8) and large ones (larger than or equal to 8). This partition is indicated by vertical lines. In the third round the numbers are partitioned according to the second bit: red numbers have 0 and blue ones have 1. In the fourth round the numbers are partitioned according to the third bit, those with 1 as the third bit are shown in italic. Finally, in the fifth round the numbers are partitioned into even and odd ones.

It is not hard to verify that this partition corresponds to the protocol described in Section 5.1.1. It is also easy to check that all the rectangles in the partition are monochromatic. For example, all the 5 rectangles consisting of large red slanted odd numbers are monochromatic: all their entries are equal to 11.

5.2 The lower bound of communication complexity of U and M .

It remains to prove that the classical complexity of functions U, M is at least 6. In the proof for M it will be easier to forget that M is a sub-function of U and instead define M as the function whose matrix is shown on Fig. 5.

Theorem 6. *The classical communication complexity of U, M is at least 6.*

Unfortunately, we cannot prove this using partitions into monochromatic rectangles, as both matrices can be partitioned into $30 \leq 2^5$ monochromatic rectangles. These rectangles correspond to the leaves of classical protocols performed by Alice and Bob in 2–5 rounds. We have 9×16 such leaves (9 events in the first rounds are multiplied by the number of leaves in a classical protocol of depth 4). However some of these rectangles can be joined together and after that we obtain 30 monochromatic rectangles. Let us show this partition for the matrix of the function S (for U and M the partition is similar). For any number $i \neq 0$ the part of the matrix consisting of entries i can be partitioned into 2 rectangles. For $i = 1$ and $i = 2$ the partition is shown in Fig. 8. And 0s can be partitioned into 6 rectangles as shown in Fig. 9. This number (30) of monochromatic rectangles cannot be decreased, which can be shown by fooling sets.

The proofs for functions U and M are almost identical. More specifically, some lemmas that are proved analytically for U can be proved for M using pictures. We will show that for every vertical partition of the matrix into two parts (that is, columns are divided into two parts) at least one part has a fooling set of size 17 (hence its communication complexity is at least 5), and the similar statement holds for horizontal partitions.

5.2.1 The proof for horizontal partitions

We distinguish in the matrix 25 rectangles denoted by

$$R_0, R_1, \dots, R_4, R_6, \dots, R_9, R_{11}, \dots, R_{14}, S_1, \dots, S_4, S_6, \dots, S_9, S_{11}, \dots, S_{14}$$

0	0	2	0	2	8	8	10	5	6	6	12	15	12	15
0	0	2	0	2	8	8	10	5	6	6	13	15	13	15
0	0	2	0	2	9	9	10	5	7	7	12	15	12	15
0	0	3	0	3	8	8	10	5	7	7	13	15	13	15
0	0	3	0	3	9	9	10	5	6	6	12	15	12	15
0	0	3	0	3	9	9	10	5	7	7	13	15	13	15
0	0	2	0	2	8	8	11	2	0	2	8	8	11	11
0	0	3	0	3	12	14	12	3	0	3	12	14	12	14
1	1	2	1	2	9	9	11	2	1	2	9	9	11	11
1	1	2	1	2	13	14	13	2	1	2	13	14	13	14
1	1	3	1	3	12	14	12	3	1	3	12	14	12	14
1	1	3	1	3	8	8	11	3	1	3	8	8	11	11
4	4	4	6	6	9	9	11	4	6	6	9	9	11	11
4	4	4	7	7	8	8	11	4	7	7	8	8	11	11
4	4	4	6	6	12	14	12	4	6	6	12	14	12	14
4	4	4	7	7	13	14	13	4	7	7	13	14	13	14
5	5	5	7	7	8	8	11	5	7	7	8	8	11	11
11	0	2	0	2	9	9	11	2	0	2	9	9	11	11
14	0	3	0	3	13	14	13	3	0	3	13	14	13	14
11	1	2	1	2	8	8	11	2	1	2	8	8	11	11
14	1	2	1	2	12	14	12	2	1	2	12	14	12	14
11	1	3	1	3	9	9	11	3	1	3	9	9	11	11
14	1	3	1	3	13	14	13	3	1	3	13	14	13	14
11	4	4	6	6	8	8	11	4	6	6	8	8	11	11
14	4	4	6	6	13	14	13	4	6	6	13	14	13	14
11	4	4	7	7	9	9	11	4	7	7	9	9	11	11
10	4	4	7	7	9	9	10	4	7	7	9	9	10	10
15	4	4	6	6	12	15	12	4	6	6	12	15	12	15
14	4	4	7	7	12	14	12	4	7	7	12	14	12	14

Figure 7: Partition of the matrix of M into monochromatic rectangles by the protocol II.

0	0	0	2	9	0	0	0	7	7	12	12	0
0	0	0	3	8	0	0	0	7	7	13	13	0
0	0	0	2	8	0	0	0	6	6	13	13	0
0	0	7	7	13	13	0	0	7	7	13	13	0
0	0	6	6	12	12	14	0	6	6	12	12	14
4	4	6	6	12	12	14	4	6	6	12	12	14
4	4	6	6	9	11	11	4	6	6	9	11	11
1	1	1	3	9	0	0	1	1	3	9	0	0
0	0	0	2	8	0	0	0	0	2	8	0	0
0	0	7	7	13	13	0	0	7	7	13	13	0
14	0	6	6	12	12	14	0	6	6	12	12	14
14	4	6	6	12	12	14	4	6	6	12	12	14
11	4	6	6	9	11	11	4	6	6	9	11	11
0	1	1	3	9	0	0	1	1	3	9	0	0
0	0	0	2	8	0	0	0	0	2	8	0	0

Figure 8: Partition of 1s and 2s in the matrix of S into two rectangles. Each rectangle has its own color.

0	0	0	2	9	0	0	0	7	7	12	12	0
0	0	0	3	8	0	0	0	7	7	13	13	0
0	0	0	2	8	0	0	0	6	6	13	13	0
0	0	7	7	13	13	0	0	7	7	13	13	0
0	0	6	6	12	12	14	0	6	6	12	12	14
4	4	6	6	12	12	14	4	6	6	12	12	14
4	4	6	6	9	11	11	4	6	6	9	11	11
1	1	1	3	9	0	0	1	1	3	9	0	0
0	0	0	2	8	0	0	0	0	2	8	0	0
0	0	7	7	13	13	0	0	7	7	13	13	0
14	0	6	6	12	12	14	0	6	6	12	12	14
14	4	6	6	12	12	14	4	6	6	12	12	14
11	4	6	6	9	11	11	4	6	6	9	11	11
0	1	1	3	9	0	0	1	1	3	9	0	0
0	0	0	2	8	0	0	0	0	2	8	0	0

Figure 9: Partition of zeros in the matrix of S into 6 rectangles. Each rectangle has its own color.

($R_5, R_{10}, R_{15}, S_0, S_5, S_{10}, S_{15}$ are skipped). Those rectangles are called *fooling rectangles*. All the entries of rectangles R_i, S_i are equal to i . Besides, for any two cells u, v from different rectangles the set $\{u, v\}$ is a fooling set for the matrix, that is, no monochromatic rectangle includes both u and v . In other words, if we pick any one cell from every fooling rectangle, the resulting set of cells is a fooling set. The existence of such a set of rectangles proves only that the matrix has a fooling set of size 25. However fooling rectangles have the following feature: *for every division of rows into two parts, the rows of one of these parts intersect at least 17 fooling rectangles*. That part thus has a fooling set of size 17.

We will define now fooling rectangles. For the matrix of M the reader can skip the definition and look at Fig. 10 instead, where fooling rectangles are indicated by colors.

Definition 8 (fooling rectangles). Recall that the value of the function is essentially equal to the 2–5-transcript of the computation, which is a 4-bit sequence $abcd$. Therefore it is convenient, in the definition of R_i, S_i , to consider i as a 4-bit sequence $abcd$. We first define rectangles R_i . If $a \neq c$, then

$$R_{abcd} = R_{ab\bar{a}d} = \{* \psi \mid \psi(a) = b, \psi(a\bar{a}) = d\} \times \{\bar{b} \varphi \mid \varphi(\Lambda) = a, \varphi(b) = \bar{a}\}.$$

Here $*$ denotes any bit 0,1 and $\bar{b} = 1 - b$. If $a = c$ and $b \neq d$, then

$$R_{abcd} = R_{aba\bar{b}} = \{\bar{a} \psi \mid \psi(a) = b, \psi(aa) = \bar{b}\} \times \{* \varphi \mid \varphi(\Lambda) = a, \varphi(b) = a\}$$

Finally, if $a = c, b = d$, which happens only if $a = c = b = d = 0$, then

$$R_{abcd} = R_{0000} = R_0 = \{* \psi \mid \psi(0) = 0, \psi(00) = 0\} \times \{* \varphi \mid \varphi(\Lambda) = 0, \varphi(0) = 0\}.$$

We now define rectangles S_i . If $a \neq c$, then

$$S_{abcd} = S_{ab\bar{a}d} = \{\mathbf{r} \eta \mid \eta(bac) = d\} \times \{b \varphi \mid \varphi(\Lambda) = a, \varphi(b) = \bar{a}\}.$$

Otherwise, if $a = c$, then

$$S_{abcd} = S_{aba\bar{b}} = \{a \psi \mid \psi(a) = b, \psi(aa) = \bar{a}\} \times \{\mathbf{r}\}.$$

The first equality holds, since rectangles $S_{0000}, S_{0101}, S_{1010}, S_{1111}$ are not defined. For reader's

0	0	2	0	2	8	8	0	0	6	6	12	0	12	0
0	0	2	0	2	8	8	0	0	6	6	13	0	13	0
0	0	2	0	2	9	9	0	0	7	7	12	0	12	0
0	0	3	0	3	8	8	0	0	7	7	13	0	13	0
0	0	3	0	3	9	9	0	0	6	6	12	0	12	0
0	0	3	0	3	9	9	0	0	7	7	13	0	13	0
0	0	2	0	2	8	8	11	2	0	2	8	8	11	11
0	0	3	0	3	12	14	12	3	0	3	12	14	12	14
1	1	2	1	2	9	9	11	2	1	2	9	9	11	11
1	1	2	1	2	13	14	13	2	1	2	13	14	13	14
1	1	3	1	3	12	14	12	3	1	3	12	14	12	14
1	1	3	1	3	8	8	11	3	1	3	8	8	11	11
4	4	4	6	6	9	9	11	4	6	6	9	9	11	11
4	4	4	7	7	8	8	11	4	7	7	8	8	11	11
4	4	4	6	6	12	14	12	4	6	6	12	14	12	14
4	4	4	7	7	13	14	13	4	7	7	13	14	13	14
0	0	0	7	7	8	8	11	0	7	7	8	8	11	11
11	0	2	0	2	9	9	11	2	0	2	9	9	11	11
14	0	3	0	3	13	14	13	3	0	3	13	14	13	14
11	1	2	1	2	8	8	11	2	1	2	8	8	11	11
14	1	2	1	2	12	14	12	2	1	2	12	14	12	14
11	1	3	1	3	9	9	11	3	1	3	9	9	11	11
14	1	3	1	3	13	14	13	3	1	3	13	14	13	14
11	4	4	6	6	8	8	11	4	6	6	8	8	11	11
14	4	4	6	6	13	14	13	4	6	6	13	14	13	14
11	4	4	7	7	9	9	11	4	7	7	9	9	11	11
0	4	4	7	7	9	9	0	4	7	7	9	9	0	0
0	4	4	6	6	12	0	12	4	6	6	12	0	12	0
14	4	4	7	7	12	14	12	4	7	7	12	14	12	14

Figure 10: Fooling rectangles for M . The rectangle R_i consists of red numbers i . The rectangle S_i consists of blue numbers i .

convenience, explicit definitions of fooling rectangles follow:

$$\begin{aligned}
R_0 &= \{*\psi \mid \psi(0) = 0, \psi(00) = 0\} \times \{1\varphi \mid \varphi(\Lambda) = 0, \varphi(0) = 0\} \\
R_1 &= \{1\psi \mid \psi(0) = 0, \psi(00) = 1\} \times \{*\varphi \mid \varphi(\Lambda) = \varphi(0) = 0\} \\
R_4 &= \{1\psi \mid \psi(0) = 1, \psi(00) = 0\} \times \{*\varphi \mid \varphi(\Lambda) = \varphi(1) = 0\} \\
R_{11} &= \{0\psi \mid \psi(1) = 0, \psi(11) = 1\} \times \{*\varphi \mid \varphi(\Lambda) = \varphi(0) = 1\} \\
R_{14} &= \{0\psi \mid \psi(1) = 1, \psi(11) = 0\} \times \{*\varphi \mid \varphi(\Lambda) = \varphi(1) = 1\} \\
R_2 &= \{*\psi \mid \psi(0) = 0, \psi(01) = 0\} \times \{1\varphi \mid \varphi(\Lambda) = 0, \varphi(0) = 1\} \\
R_3 &= \{*\psi \mid \psi(0) = 0, \psi(01) = 1\} \times \{1\varphi \mid \varphi(\Lambda) = 0, \varphi(0) = 1\} \\
R_6 &= \{*\psi \mid \psi(0) = 1, \psi(01) = 0\} \times \{0\varphi \mid \varphi(\Lambda) = 0, \varphi(1) = 1\} \\
R_7 &= \{*\psi \mid \psi(0) = 1, \psi(01) = 1\} \times \{0\varphi \mid \varphi(\Lambda) = 0, \varphi(1) = 1\} \\
R_8 &= \{*\psi \mid \psi(1) = 0, \psi(10) = 0\} \times \{1\varphi \mid \varphi(\Lambda) = 1, \varphi(0) = 0\} \\
R_9 &= \{*\psi \mid \psi(1) = 0, \psi(10) = 1\} \times \{1\varphi \mid \varphi(\Lambda) = 1, \varphi(0) = 0\} \\
R_{12} &= \{*\psi \mid \psi(1) = 1, \psi(11) = 0\} \times \{0\varphi \mid \varphi(\Lambda) = 1, \varphi(1) = 0\} \\
R_{13} &= \{*\psi \mid \psi(1) = 1, \psi(11) = 1\} \times \{0\varphi \mid \varphi(\Lambda) = 1, \varphi(1) = 0\} \\
S_1 &= \{0\psi \mid \psi(0) = 0, \psi(00) = 1\} \times \{r\}, \\
S_4 &= \{0\psi \mid \psi(0) = 1, \psi(00) = 0\} \times \{r\}, \\
S_{11} &= \{1\psi \mid \psi(1) = 0, \psi(11) = 1\} \times \{r\}, \\
S_{14} &= \{1\psi \mid \psi(1) = 1, \psi(11) = 0\} \times \{r\}, \\
S_2 &= \{r\eta \mid \eta(001) = 0\} \times \{0\varphi \mid \varphi(\Lambda) = 0, \varphi(0) = 1\}, \\
S_3 &= \{r\eta \mid \eta(001) = 1\} \times \{0\varphi \mid \varphi(\Lambda) = 0, \varphi(0) = 1\}, \\
S_6 &= \{r\eta \mid \eta(011) = 0\} \times \{1\varphi \mid \varphi(\Lambda) = 0, \varphi(1) = 1\}, \\
S_7 &= \{r\eta \mid \eta(011) = 1\} \times \{1\varphi \mid \varphi(\Lambda) = 0, \varphi(1) = 1\}, \\
S_8 &= \{r\eta \mid \eta(100) = 0\} \times \{0\varphi \mid \varphi(\Lambda) = 1, \varphi(0) = 0\}, \\
S_9 &= \{r\eta \mid \eta(100) = 1\} \times \{0\varphi \mid \varphi(\Lambda) = 1, \varphi(0) = 0\}, \\
S_{12} &= \{r\eta \mid \eta(110) = 0\} \times \{1\varphi \mid \varphi(\Lambda) = 1, \varphi(1) = 0\}, \\
S_{13} &= \{r\eta \mid \eta(110) = 1\} \times \{1\varphi \mid \varphi(\Lambda) = 1, \varphi(1) = 0\},
\end{aligned}$$

Lemma 5. (1) All cells of rectangles R_i, S_i include the number i only.

(2) If u, v are any cells from different fooling rectangles, then the set $\{u, v\}$ cannot be covered by a monochromatic rectangle (in particular, $u \neq v$, that is, fooling rectangles are pair wise disjoint).

Proof. For the function M this can be verified directly by examining Fig. 10. For the function U the proof is in the Appendix. \square

We call two fooling rectangles *horizontally adjacent*, if their first projections intersect (there is a row that intersects both rectangles). Consider the non-oriented graph whose

nodes are fooling rectangles and edges connect horizontally adjacent rectangles. We say that a vertex u is a *neighbor* of a vertex v if u, v are adjacent. In particular, each node is its own neighbor. It turns out that this graph has the following expansion property:

Lemma 6. *Every set of 9 vertices of the graph has at least 17 neighbors.*

The proof of this lemma is deferred to Appendix. Let us derive from this lemma the proof for horizontal partitions. Assume that rows of the matrix are partitioned in sets W and V . Let us denote by \mathcal{R} the set of all fooling rectangles which intersect some row from W . If there are at least 17 such rectangles, then we are done. Otherwise there are at least $25 - 16 = 9$ rectangles outside \mathcal{R} . For those rectangles every row intersecting the rectangle belongs to V . Thus every neighbor y of every vertex x outside \mathcal{R} intersects a row in V . (Indeed there is a row s that intersects both x and y . That row is not in W , as otherwise x would be in \mathcal{R} . Hence $s \in V$ and y intersects a row in V .) Since there are at least 9 rectangles outside \mathcal{R} , the lemma implies that there are at least 17 fooling rectangles intersecting a row in V . The case of horizontal partitions is completed.

Now we are able to explain how the set Alice's inputs was reduced when we restricted the function U to get the function M . Each row in the matrix of U yields a clique in the graph, the set of edges of the graph is the union of those cliques. We have chosen a minimal set of cliques whose union covers all the edges of the graph and removed the rows corresponding to the remaining cliques.

5.2.2 Proof for vertical partitions

Let us call two fooling rectangles *vertically adjacent* if their second projections intersect. Unfortunately, the analog of Lemma 6 is not true anymore. Indeed, the connected components of the vertical adjacency graph are the following:

$$\begin{aligned} & \{S_1, S_4, S_{11}, S_{14}\}, \\ & \{R_0, S_2, S_3, S_6, S_7, R_1, R_2, R_3, R_4, R_6, R_7\}, \\ & \{S_8, S_9, S_{12}, S_{13}, R_8, R_9, R_{11}, R_{12}, R_{13}, R_{14}\}. \end{aligned}$$

These components correspond to three sub-matrices into which the matrix M is divided by ordinary vertical lines in Fig. 7. Let us join the first and the second connected components. In this way we obtain a vertical partition of the matrix into two sub-matrices, where the first sub-matrix intersects 15 fooling rectangles, and the second one intersects 10 fooling rectangles.

This obstacle forces to increase the number of fooling rectangles. This can be done by adding rectangles with cells containing 0. To do this, we will reduce the rectangle R_0 and add rectangles called R_5, R_{10}, R_{15}, S_0 . Namely, let

$$\begin{aligned} R_{abab} &= \{*\psi \mid \psi(a) = \psi(aa) = b, \psi(\bar{a}) \neq \psi(\bar{a}\bar{a})\} \\ &\quad \times \{\bar{b}\varphi \mid \varphi(\Lambda) = \varphi(b) = a, \varphi(\bar{b}) = \bar{a}\}, \text{ where } a, b = 0, 1, \\ S_0 &= \{\mathbf{r}\eta \mid \eta \text{ arbitrary}\} \times \{i\varphi \mid \varphi(\Lambda) = \varphi(i)\} \end{aligned}$$

Here are the explicit definitions of rectangles R_0, R_5, R_{10}, R_{15} :

$$\begin{aligned} R_0 &= \{*\psi \mid \psi(0) = 0, \psi(00) = 0, \psi(1) \neq \psi(11)\} \times \{1\varphi \mid \varphi(\Lambda) = 0, \varphi(0) = 0, \varphi(1) = 1\} \\ R_5 &= \{*\psi \mid \psi(0) = 1, \psi(00) = 1, \psi(1) \neq \psi(11)\} \times \{0\varphi \mid \varphi(\Lambda) = 0, \varphi(1) = 0, \varphi(0) = 1\}, \\ R_{10} &= \{*\psi \mid \psi(1) = 0, \psi(11) = 0, \psi(0) \neq \psi(00)\} \times \{1\varphi \mid \varphi(\Lambda) = 1, \varphi(0) = 1, \varphi(1) = 0\} \\ R_{15} &= \{*\psi \mid \psi(1) = 1, \psi(11) = 1, \psi(0) \neq \psi(00)\} \times \{0\varphi \mid \varphi(\Lambda) = 1, \varphi(1) = 1, \varphi(0) = 0\}. \end{aligned}$$

In fact, for the function M , the new version of the rectangle R_0 coincides with the old one, since the difference between them is due to those rows that were removed from the matrix of U . The fooling rectangles for the matrix of M are shown in Fig. 11.

We need an analogue of the Lemma 5 for new fooling rectangles:

Lemma 7. (1) *All cells of the rectangles $R_0, R_5, R_{10}, R_{15}, S_0$ contain 0.*

(2) *Any two cells u, v from different rectangles from the list $R_0, S_0, R_5, R_{10}, R_{15}$ cannot be covered by one monochromatic rectangle (the cells of old fooling rectangles are marked with non-zeros, so they can also be added to this list).*

For M , this lemma is verified directly by looking at Fig. 11. For U , the lemma is proved in Appendix.

To complete the proof of the theorem, we need an analog of Lemma 6 for the vertical adjacency graph. The number of fooling rectangles has increased to 29, so instead of 9 we now have $29 - 16 = 13$.

Lemma 8. *Any set of 13 vertices of the vertical adjacency graph has at least 17 neighbors.*

This lemma is proven in Appendix. As before, it implies that for any vertical division of the matrices of M and U in two parts at least one of the parts has a fooling set of 17 cells.

The case of vertical partitions is completed (modulo Lemmas 7 and 8). Theorems 5 and 6 are proved.

6 Open questions

1. Is there a partial function f on words of length n over a fixed alphabet with a super-logarithmic gap between the local half-duplex and the local classical complexities?

2. Is there a total function with values 0,1 for which the half-duplex complexity is strictly less than the classical complexity?

3. Is there a total function f on words of length n over a fixed alphabet with a super-constant gap between the half-duplex and the classical complexity?

We believe that all our questions answer in positive. Regarding the third question, we think that the appropriate example is the n th power of the function of Section 5 but we do not see any means to prove that.

0	0	2	0	2	8	8	0	0	6	6	12	0	12	0
0	0	2	0	2	8	8	0	0	6	6	13	0	13	0
0	0	2	0	2	9	9	0	0	7	7	12	0	12	0
0	0	3	0	3	8	8	0	0	7	7	13	0	13	0
0	0	3	0	3	9	9	0	0	6	6	12	0	12	0
0	0	3	0	3	9	9	0	0	7	7	13	0	13	0
0	0	2	0	2	8	8	11	2	0	2	8	8	11	11
0	0	3	0	3	12	14	12	3	0	3	12	14	12	14
1	1	2	1	2	9	9	11	2	1	2	9	9	11	11
1	1	2	1	2	13	14	13	2	1	2	13	14	13	14
1	1	3	1	3	12	14	12	3	1	3	12	14	12	14
1	1	3	1	3	8	8	11	3	1	3	8	8	11	11
4	4	4	6	6	9	9	11	4	6	6	9	9	11	11
4	4	4	7	7	8	8	11	4	7	7	8	8	11	11
4	4	4	6	6	12	14	12	4	6	6	12	14	12	14
4	4	4	7	7	13	14	13	4	7	7	13	14	13	11
0	0	0	7	7	8	8	11	0	7	7	8	8	11	11
11	0	2	0	2	9	9	11	2	0	2	9	9	11	11
14	0	3	0	3	13	14	13	3	0	3	13	14	13	14
11	1	2	1	2	8	8	11	2	1	2	8	8	11	11
14	1	2	1	2	12	14	12	2	1	2	12	14	12	14
11	1	3	1	3	9	9	11	3	1	3	9	9	11	11
14	1	3	1	3	13	14	13	3	1	3	13	14	13	14
11	4	4	6	6	8	8	11	4	6	6	8	8	11	11
14	4	4	6	6	13	14	13	4	6	6	13	14	13	14
11	4	4	7	7	9	9	11	4	7	7	9	9	11	11
0	4	4	7	7	9	9	0	4	7	7	9	9	0	0
0	4	4	6	6	12	0	12	4	6	6	12	0	12	0
14	4	4	7	7	12	14	12	4	7	7	12	14	12	14

Figure 11: Fooling rectangles for vertical partitions for the matrix of M . The rectangle R_i for $i \neq 10, 15, 20$ consists of red numbers i . The rectangle S_i for $i \neq 0$ consists of blue numbers i . The rectangles R_5, R_{10}, R_{15} consist of only one cell each: yellow, cyan and green zeros. The rectangle S_0 consists of magenta zeros.

7 Acknowledgments

The authors are sincerely grateful to Timur Kuptsov and anonymous referees for helpful suggestions.

A Appendix: deferred proofs

Proof of Lemma 5 for U . (1) This is verified directly. Let us perform this verification, say, for the rectangle $R_6 = R_{0110}$: in the first round, both players send bits that are lost and have no effect on the 2-5-transcript. In the second round Bob sends 0 (because $\varphi(\Lambda) = 0$). Alice then sends 1 (since $\psi(0) = 1$). Then Bob sends 1 (since $\varphi(1) = 1$). Finally, in the last round Alice sends 0, since $\psi(01) = 0$. We get the 2-5-transcript 0110 = 6.

(2) Let u, v belong to different fooling rectangles.

Case 1. These rectangles have different subscripts. Then $U(u) \neq U(v)$ due to item (1). Hence u and v cannot be covered by a monochromatic rectangle.

Case 2. Rectangles containing u, v have the same subscript.

Assume first that $u \in R_{abcd}$ and $v \in S_{abcd}$ where $a \neq c$. Then $u = (*\psi, \bar{b}\varphi)$, $v = (\mathbf{r}\eta, b\varphi')$. Consider the input pair $(\mathbf{r}\eta, \bar{b}\varphi)$. First, the 2-5-transcript t' for this pair is different from $t = abcd$. Indeed, since for this input pair in the first round Alice receives \bar{b} from Bob, she sends in the third round \bar{b} , and not b . Second, the values of the output function p on t and t' are different, since otherwise in both t, t' the first and third bits coincide, which is not the case for t .

Assume now that $u \in R_{abcd}$ and $v \in S_{abcd}$ where $a = c$ and $b \neq d$. Then $u = (\bar{a}\psi, *\varphi)$, $v = (a\psi', \mathbf{r})$. Consider the input pair $(\bar{a}\psi, \mathbf{r})$. First, the transcript t' on this pair is different from $abcd$. Indeed, since in the first round Bob receives \bar{a} from Alice, he sends in the second round \bar{a} . Therefore, t' begins with \bar{a} and hence differs from $t = abcd$. And again the values of the output function on t and t' differ because $b \neq d$. \square

Proof of Lemma 6. We need an explicit description of edges of the graph.

Lemma 9. *The graph of horizontally adjacent fooling rectangles is a union of the following graphs (see Fig. 12):*

1. the complete 4-partite graph whose first part is $\{S_2, S_3\}$, the second part is $\{S_6, S_7\}$, the third part is $\{S_8, S_9\}$ and the fourth part is $\{S_{12}, S_{13}\}$.
2. the complete bipartite graph whose first part is $\{R_0, R_1, S_1\}$ and second part is $\{R_2, R_3\}$
3. the complete bipartite graph whose first part is $\{R_4, S_4\}$ and second part is $\{R_6, R_7\}$
4. the complete bipartite graph whose first part is $\{R_{11}, S_{11}\}$ and second part is $\{R_8, R_9\}$
5. the complete bipartite graph whose first part is $\{R_{14}, S_{14}\}$ and second part is $\{R_{12}, R_{13}\}$

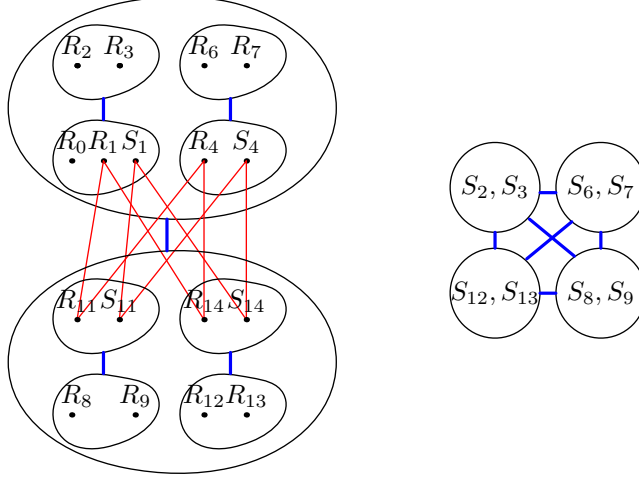


Figure 12: The graph of horizontally adjacent fooling rectangles. Each oval (circle) encircles vertices from one part of a bipartite or many-partite graph. Blue lines between ovals represent edges connecting nodes from different parts of those bipartite graphs. For instance, each blue line from the right represents 4 edges. Red lines represent edges that are excluded from the complete bipartite graph.

6. the complete bipartite graph whose first part is $\{R_0, R_1, R_2, R_3, R_4, R_6, R_7, S_1, S_4\}$ and second part is $\{R_8, R_9, R_{11}, R_{12}, R_{13}, R_{14}, S_{11}, S_{14}\}$, except all edges of the complete bipartite graph whose first part is $\{R_1, R_4, S_{11}, S_{14}\}$ and second part is $\{R_{11}, R_{14}, S_1, S_4\}$.

Proof. Actually, we only need to show that all listed above edges are indeed present in the graph. For the function M this can be verified by contemplating Fig. 10. For the function U this is proved as follows.

Recall that two rectangles are connected by an edge if their first projections intersect. We will call the second character of a word from the first projection of a rectangle its ψ - or η -projection.

In the first item, we listed all the fooling rectangles whose first projections contains words of the form $\mathbf{r}*$. Their first projections have the form $\{\mathbf{r}\eta \mid \eta(bac) = d\}$. Such a set does not intersect another set $\{\mathbf{r}\eta \mid \eta(\tilde{b}\tilde{a}\tilde{c}) = \tilde{d}\}$ of this form only if $a = \tilde{a}, b = \tilde{b}, c = \tilde{c}$, but $d \neq \tilde{d}$. That is, among the rectangles $S_2, S_3, S_6, S_7, S_8, S_9, S_{12}, S_{13}$ there are no edges only between pairs $(S_{0010}, S_{0011}), (S_{0110}, S_{0111}), (S_{1000}, S_{1001}), (S_{1100}, S_{1101})$.

The first projections of all the other fooling rectangles contain words of the form $*\psi$. Their first character, except for rectangles $R_1, R_4, R_{11}, R_{14}, S_1, S_4, S_{11}, S_{14}$, can be arbitrary. For these exceptional rectangles, the first character is 1 for rectangles R_1, R_4, S_{11}, S_{14} and is 0 for rectangles R_{11}, R_{14}, S_1, S_4 . This explains the exclusion of edges in the sixth item. It is not hard to verify that, in items 2–5, no edges are excluded for this reason: in these items, in one of the parts, there are no rectangles from the list $R_1, R_4, R_{11}, R_{14}, S_1, S_4, S_{11}, S_{14}$. It remains to verify that in items 2–6 ψ - or η -projections intersect.

The edges in items 2–5 connect the rectangles R_{abad} and S_{abad} with rectangles of the

form $R_{ab\bar{a}^*}$. We need to prove that ψ -projections of rectangles R_{abad} and S_{abad} intersect with ψ -projections of rectangles of the form $R_{ab\bar{a}^*}$. Note that ψ -projections of R_{abad} and S_{abad} coincide and are equal to

$$A = \{\psi \mid \psi(a) = b, \psi(aa) = d\}.$$

On the other hand, the ψ -projection of $R_{ab\bar{a}e}$ is equal to

$$C = C_e = \{\psi \mid \psi(a) = b, \psi(a\bar{a}) = e\}.$$

Therefore the intersection $A \cap C$ includes all ψ , with $\psi(a) = b, \psi(aa) = d, \psi(a\bar{a}) = e$. It is easy to see that these requirements are compatible.

In the sixth item, the rectangles in the first part have the form $R_{0bcd}, S_{0b0\bar{b}}$, and in the second part rectangles have the same form, only 0 and 1 are swapped: $R_{1b'c'd'}, S_{1b'1\bar{b}'}$. Their ψ -projections are equal, respectively, to

$$\begin{aligned} & \{\psi \mid \psi(0) = b, \psi(0c) = d\}, \\ & \{\psi \mid \psi(0) = b, \psi(00) = \bar{b}\}, \\ & \{\psi \mid \psi(1) = b', \psi(1c') = d'\}, \\ & \{\psi \mid \psi(1) = b', \psi(11) = \bar{b}'\}. \end{aligned}$$

Obviously, the first and second sets intersect with the third and fourth sets. □

Now we are able to prove Lemma 6 by considering several cases. The following sets of 9 vertices have minimal number of neighbors (17):

- The set of nodes in the top left oval. All the vertices from both left ovals are their neighbors, 17 vertices in total.
- The set of nodes from four circles on the right (8 nodes) plus any node from the top left oval which is incident to a red edge (e.g. R_1). The extra node has 9 neighbors thus we get 17 nodes in total.

Let us show that this are indeed the worst case. Assume that a set A of vertices contains k vertices from the left component of connectivity and l vertices from the right component, $k + l = 9$. Since there are only 8 vertices in the right component, we know that $k \geq 1$. We distinguish three cases: $l = 0, l \geq 2$ and $l = 1$.

Case $k = 9, l = 0$. It is not difficult to verify that any vertex u from the left component has at least 9 neighbors. So the number of non-neighbors of u in the left component does not exceed $17 - 9 = 8$. Therefore, u is a neighbor of any set of 9 vertices from the left component. Since this is true for any vertex u , every node from the 17 nodes of the left component is a neighbor of each set of 9 vertices from the left component, and we are done.

Case $l \geq 2$. Then one left vertex of A has 9 or more neighbors in the left component, and the two right vertices of A have 8 neighbors from the right component (if these vertices are from the same circle, then they are neighbors of themselves, otherwise they are neighbors of

each other, and the remaining 6 vertices will be their neighbors, since they belong to another part comparing to one of these two vertices). In total, we get 17 neighbors.

Case $k = 8, l = 1$. One A 's vertex on the right has 7 neighbors and it is enough to verify that any set B of 8 vertices of the left component has at least 10 neighbors. Note that there are only four vertices in the left component that have 9 neighbors, all the other have at least ten neighbors. Therefore, B contains a vertex with 10 neighbors. \square

Proof of the Lemma 7 for the function U . (1) For rectangles R_i , this is verified directly. For the rectangle S_0 : let the pair $(r\eta, i\varphi)$ belong to S_0 . Let j denote $\varphi(\Lambda) = \varphi(i)$. In the first round, Alice receives i from Bob. Therefore, Alice sends i in the third round. Thus, the transcript in 2–4 rounds looks like this: jij . Finally, in the fifth round, Alice sends i again, since the bits she received in the second and fourth rounds coincide. We get the transcript $jijji$ hence $U(r\eta, i\varphi) = 0$.

(2) Let u, v belong to different fooling rectangles. First, let us assume that both rectangles have the form R_{abab} . Say the first one has the subscript $abab$, and the second the subscript $a'b'a'b'$, Let $u = (*\psi, \bar{b}\varphi)$, $v = (*\psi', \bar{b}'\varphi')$. Then we know that

$$\psi(a) = \psi(aa) = b \tag{1}$$

$$\psi(\bar{a}) \neq \psi(\bar{a}\bar{a}) \tag{2}$$

$$\varphi'(\Lambda) = \varphi'(b') = a' \tag{3}$$

$$\varphi'(\bar{b}') \neq a'. \tag{4}$$

Consider the input pair $(*\psi, \bar{b}'\varphi')$. We claim that $U(*\psi, \bar{b}'\varphi') \neq 0$. For the sake of contradiction, assume that the 2–5-transcript on this input pair is a multiple of 5, that is, it has the form $\alpha\beta\alpha\beta$. Then we additionally know that

$$\psi(\alpha) = \psi(\alpha\alpha) = \beta \tag{5}$$

$$\varphi'(\Lambda) = \varphi'(\beta) = \alpha. \tag{6}$$

We claim that from (1)–(6) it follows that $a = a' = \alpha$ and $b = b' = \beta$, and therefore the rectangles $R_{abab}, R_{a'b'a'b'}$ coincide. From (5) and (2) it follows that $\alpha \neq \bar{a}$, that is, $\alpha = a$. On the other hand, it follows from (3) and (6) that $\alpha = \varphi'(\Lambda) = a'$. From (6) it follows that $\varphi'(\beta) = \alpha = a$. Now from (4) and (6) it follows that $\varphi'(\bar{b}') \neq a' = \alpha = \varphi'(\beta)$, which means that $\bar{b}' \neq \beta$, that is, $\beta = b'$. Finally, from (5) and (1) we conclude that $\beta = \psi(\alpha) = \psi(a) = b$.

It remains to consider the case when one rectangle is S_0 , and the other one has the form R_{abab} . In this case we have $u = (r\eta, i\varphi)$, $v = (a'\psi', \bar{b}'\varphi')$ and we know the equalities (3) and (4). We want to prove that $U(r\eta, \bar{b}'\varphi') \neq 0$. For the sake of contradiction, assume that the 2–5-transcript for this input pair has the form $\alpha\beta\alpha\beta$. Then additionally we know (6), and besides $\beta = \bar{b}'$, since Alice sends the bit \bar{b}' received from Bob in the third round. From (6) we derive that $\alpha = \varphi'(\beta) = \varphi'(\bar{b}')$. And on the other hand, from (3) and (6) it follows that $\alpha = \varphi'(\Lambda) = a'$. Therefore, $a' = \varphi'(\bar{b}')$, which contradicts (4). \square

Proof of Lemma 8. Again, we need a more explicit description of edges of the graph.

Lemma 10. *The vertical adjacency graph is equal to the graph shown in Fig. 13 (for both functions U, M).*

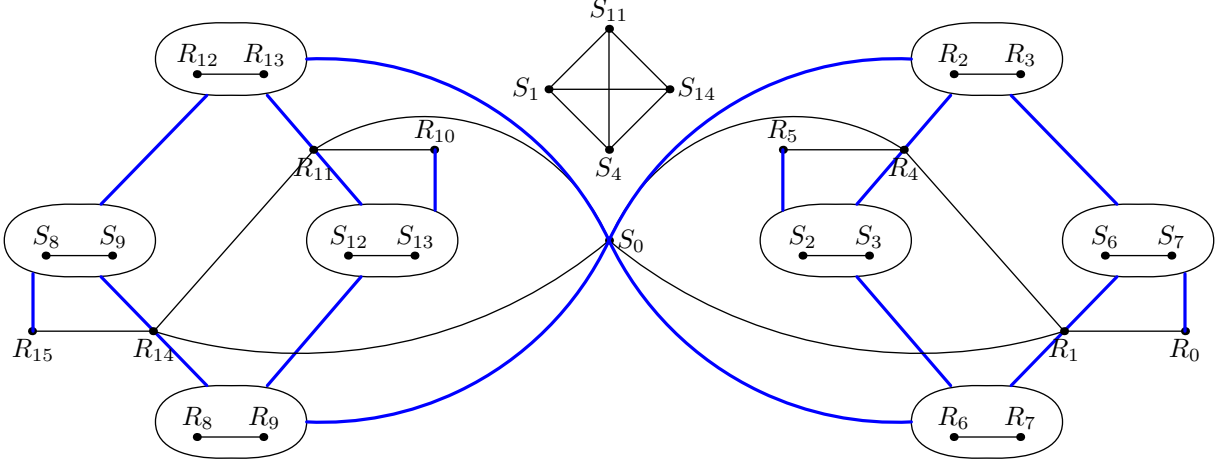


Figure 13: The graph of vertically adjacent fooling rectangles. Blue lines represent multiple edges.

Proof. This lemma for M is verified directly by looking at the matrix of the function. For U the proof is as follows.

Let us write out the second projections of fooling rectangles:

$$\begin{aligned}
 \text{pr}_2 R_{ab\bar{a}d} &= \{\bar{b}\varphi \mid \varphi(\Lambda) = a, \varphi(b) = \bar{a}\} \\
 \text{pr}_2 R_{aba\bar{b}} &= \{*\varphi \mid \varphi(\Lambda) = a, \varphi(b) = a\} \\
 \text{pr}_2 R_{abab} &= \{\bar{b}\varphi \mid \varphi(\Lambda) = a, \varphi(b) = a, \varphi(\bar{b}) = \bar{a}\} \\
 \text{pr}_2 S_{ab\bar{a}d} &= \{b\varphi \mid \varphi(\Lambda) = a, \varphi(b) = \bar{a}\} \\
 \text{pr}_2 S_{aba\bar{b}} &= \{\mathbf{r}\} \\
 \text{pr}_2 S_0 &= \{i\varphi \mid \varphi(\Lambda) = \varphi(i)\}.
 \end{aligned}$$

To prove the lemma, we will make several simple observations, immediately following from the definitions:

1. The graph splits into two connected components: $\{S_1, S_4, S_{11}, S_{14}\}$ (their second projection contains only \mathbf{r}) and all other vertices. The first component is a complete graph on vertices $\{S_1, S_4, S_{11}, S_{14}\}$.
2. Rectangle S_0 is adjacent to all vertices R_i , except R_0, R_5, R_{10}, R_{15} .
3. If we remove the rectangle S_0 , then the second component again splits into two connected components. One of them consists of all rectangles of the form R_{0bcd}, S_{0bcd} , and the other one of all rectangles of the form R_{1bcd}, S_{1bcd} . We will call these components

the second and *the third* ones. The second and third components are isomorphic, they are obtained from each other by swapping zero and one. Therefore, it is sufficient to describe only the second component.

4. Second projections of rectangles $R_{ab\bar{a}0}, R_{ab\bar{a}1}$ coincide. Therefore these rectangles are adjacent. Besides, from them the edges lead to the same vertices. The same applies to pairs of the form $S_{ab\bar{a}0}, S_{ab\bar{a}1}$. In particular, the second component contains the edges $(R_0, R_1), (R_2, R_3), (S_2, S_3), (R_4, R_5), (R_6, R_7), (S_6, S_7)$.
5. Finally, the second component contains all the edges of the complete bipartite graph with the first part $R_0, R_1, R_2, R_3, S_2, S_3$ and the second part $R_4, R_5, R_6, R_7, S_6, S_7$ with the exception of the edges $(R_0, R_4), (R_0, R_5), (R_1, R_5)$ and all edges of a complete bipartite graph with the first part R_0, R_2, R_3, S_6, S_7 and the second part R_5, R_6, R_7, S_2, S_3 .

In the first part, we listed all rectangles R_{abcd} from the second component with $a = b = 0$. In the second part, those with $a = 0$ and $b = 1$. Let us verify first that φ -projections of all the listed pairs intersect, that is, the corresponding conditions on φ are compatible. Let $R(S)_{00cd}$ and $R(S)_{01c'd'}$ be rectangles from the second component. Then requirements for $\varphi(\Lambda)$ in φ -the projections for both rectangles coincide. And because the second bits in $00cd$ and $01c'd'$ are different, the conditions on $\varphi(b)$ are compatible. Finally, if the first of these rectangles is R_{0000} or the second rectangle is R_{0101} (or both), then there is an additional requirement $\varphi(\bar{b}) = \bar{a}$. These conditions may be incompatible for rectangles $R(S)_{00cd}$ and $R(S)_{01c'd'}$. It is not difficult to verify that this happens only for the pairs $(R_0, R_4), (R_0, R_5), (R_1, R_5)$. For this reason, these edges were excluded.

It remains to verify that the first symbols can also be the same. For rectangles of the form $R_{aba\bar{b}}$, the first characters can be arbitrary. For rectangles R_0, R_2, R_3, S_6, S_7 the first symbol is equal to 1, and for rectangles R_5, R_6, R_7, S_2, S_3 the first symbol is equal to 0. Because of this, edges between rectangles of the first and the second lists were excluded.

□

The graph in Fig. 13 consists of a complete graph on vertices S_1, S_{11}, S_4, S_{14} and two isomorphic graphs on 12 vertices together with the vertex S_0 connecting them. In the sequel, we will call these two parts of the 12 vertices *the components* of the graph. The smallest set of neighbors of a set of 13 vertices is attained, for example, if we take all 12 vertices from the left component and the vertex R_0 from the right component, or any 9 vertices from the left component and vertices S_1, S_4, S_{11}, S_{14} .

Let us prove that this is indeed the minimal neighborhood of any set of 13 nodes. To do this, we need the following auxiliary notion. Let A be an arbitrary set of vertices from one component of the graph. Denote by $\Gamma(A)$ the set of all neighbors of A , and by $\Gamma'(A)$ the set of all neighbors, not counting the vertex S_0 , that is, the set of all neighbors inside that component. By $\Gamma(n)$ we denote the minimum $|\Gamma(A)|$ for all n -element sets inside one

n	$\Gamma'(n)$	$\Gamma(n)$
1	4	4
2	5	6
3	6	6
4	7	8
5	7	8
6	9	10
7	10	11
8	11	12
9	12	13
10	12	13
11	12	13
12	12	13

Table 1: The table of values of functions Γ and Γ' expressing the expansion properties of components of the vertical adjacency graph.

component. $\Gamma'(n)$ is defined in a similar way. The values of these functions are given in Table 1, they were found by hand. In fact, we will need the values of these functions only for $n = 1, 4, 6, 7, 9$.

Lemma 11. (a) $\Gamma'(1) \geq 4$

(b) $\Gamma'(4) \geq 7$

(c) $\Gamma'(6) \geq 9$,

(d) $\Gamma'(7) \geq 10$,

(e) $\Gamma'(9) \geq 13$.

Let us finish the proof of Lemma 8 assuming Lemma 11. Let a set A consist of 13 vertices of the adjacency graph. We need to prove that A has at least 17 neighbors. Let us consider two cases.

Case 1: the set A contains the vertex S_0 . First, assume that the set A contains at least one of the vertices S_1, S_4, S_{11}, S_{14} . Then there must be at least $13 - 1 - 4 = 8$ vertices in the left and right components together. Therefore, one of the components must contain at least $8/2 = 4$ vertices. Thus the number of neighbors of A is not less than $5 + \Gamma'(4) + 6 \geq 5 + 7 + 6 = 18$ neighbors (vertices $S_0, S_1, S_4, S_{11}, S_{14}$ plus all neighbors inside the component containing 4 vertices, plus the neighbors of vertex S_0 inside the other component).

Otherwise the set A does not contain any of the vertices S_1, S_4, S_{11}, S_{14} . Then, there should be at least $13 - 1 = 12$ vertices in the left and right components in total. If there are 6 vertices in both components, then A has at least $1 + 2\Gamma'(6) \geq 1 + 2 \cdot 9 = 19$ neighbors. Otherwise, one of the components has at least $12 - 5 = 7$ vertices, and the number of neighbors is not less $1 + \Gamma'(7) + 6 \geq 1 + 10 + 6 = 17$ (the vertex S_0 plus all the neighbors inside the component containing 7 vertices, plus the neighbors of S_0 inside the other component).

Case 2: $S_0 \notin A$. Assume first that A has at least one of the vertices S_1, S_4, S_{11}, S_{14} . In total, both components have at least $13 - 4 = 9$ vertices from the set A . If all these vertices are in one component, then the total number of A 's neighbors is at least $\Gamma(9) + 4 \geq 13 + 4 = 17$ (neighbors of these 9 vertices plus vertices S_1, S_4, S_{11}, S_{14}). If each component has at least one vertex from A and one of the components has at least 6 vertices from A , then the number of A 's neighbors is at least $\Gamma(6) + \Gamma'(1) + 4 \geq \Gamma'(6) + \Gamma'(1) + 4 \geq 9 + 4 + 4 = 17$. Otherwise, there are 5 vertices in one component, and 4 in the other, and the number of A 's neighbors is at least $\Gamma(5) + \Gamma'(4) + 4 \geq \Gamma'(4) + \Gamma'(4) + 4 \geq 7 + 7 + 4 = 18$.

It remains to consider the case when $S_0, S_1, S_4, S_{11}, S_{14} \notin A$. Since there are only 12 vertices in each component, each component must have at least one vertex from A . If one of them has at least 9 vertices from A , then we get at least $\Gamma(9) + \Gamma'(1) \geq 13 + 4 = 17$ neighbors. Otherwise, the division of vertices between components is $5 + 8$ or $6 + 7$. In both cases the number of neighbors is not less than $\Gamma(5) + \Gamma'(7) \geq \Gamma'(4) + \Gamma'(7) \geq 7 + 10 = 17$. Lemma 8 is proved (modulo Lemma 11). \square

Proof of Lemma 11. (a) This can be directly verified by hand.

(b) Assume that there is a set A of 4 vertices of the right component that has less than 7 neighbors inside that component. We want to derive a contradiction.

Let us first assume that A does not contain any vertices from the ovals. Then $A = \{R_0, R_1, R_4, R_5\}$ and has $12 \geq 7$ neighbors inside the component. So A must contain a vertex from one of the ovals and without loss of generality another vertex from the same oval (indeed, adding this other vertex to A does not increase the number of neighbors, so it can replace any other vertex in A). Identifying symmetric variants, we can consider only two cases: $R_2, R_3 \in A$ and $S_2, S_3 \in A$.

First case: $R_2, R_3 \in A$. These two vertices together already have 5 neighbors R_2, R_3, R_4, S_6, S_7 . It is easy to verify that adding to R_2, R_3 any other vertex increases the number of neighbors by at least 2.

The second case: $S_2, S_3 \in A$. These two vertices together already have neighbors $S_2, S_3, R_4, R_5, R_6, R_7$. It is easy to see that adding to S_2, S_3 any vertex, except R_5 , increases the number of neighbors. Since we need to add two vertices, we get a contradiction.

(c) To reduce the number of cases, we will consider the complements. Let us state this technique in general. We claim that the inequality $\Gamma'(m) \geq n$ implies the inequality $\Gamma'(13 - n) \geq 13 - m$. Indeed, assume that $\Gamma'(13 - n) < 13 - m$. That is, there is a set A of cardinality $13 - n$ having at most $12 - m$ neighbors. Then consider the set B consisting of non-neighbors A , $|B| \geq 12 - (12 - m) = m$. Then all the neighbors of B lie in the complement of A , so $|\Gamma'(B)| \leq 12 - (13 - n) = n - 1$. By removing $|B| - m$ vertices from B , we get a set of B' of cardinality exactly m with $|\Gamma'(B')| \leq n - 1$.

It is worth to apply this technique when $13 - n$ is closer to 6 than m , because in this case the number of $(13 - n)$ -element subsets of a component is less than the number of its m -element subsets. For example, the statement (c) follows from statement (b), and we do not have to look over all 6-element subsets of a component.

(d) Using the same technique, it suffices to prove the inequality $\Gamma'(3) \geq 6$. Assume that there is a set A of 3 vertices of a component that has less than 6 neighbors inside that

component. We want to derive a contradiction.

First, assume that A does not contain any vertices from the ovals. Identifying symmetric variants, there are only two such sets: $\{R_0, R_1, R_4\}$ and $\{R_0, R_1, R_5\}$. In the first case, all vertices of the components are neighbors, and in the second case all vertices except R_2, R_3 .

Now assume that A contains a vertex from an oval. Without loss of generality, we can assume that the other vertex of the oval also lies in A .

Identifying symmetrical cases, only two cases can be considered: $A \ni R_2, R_3$ and $A \ni S_2, S_3$. In the first case, the vertices R_2, R_3 already have in total 5 neighbors R_2, R_3, R_4, S_6, S_7 , so the third vertex of A should be in this list and should not have neighbors out of the list. By a simple search we can see that there are no such vertices. In the second case, the vertices S_2, S_3 already have 6 neighbors in total.

(e) Using our technique, we can deduce from item (a) that $\Gamma'(9) \geq 12$. In addition, the vertex S_0 is a neighbor of any set of $12 - 6 + 1 = 7$ vertices inside one component, since it has 6 neighbors inside one component. Therefore, for all n starting from $n = 7$, we have $\Gamma'(n) = \Gamma(n) + 1$. \square

References

- [1] Dementiev, Y., Ignatiev, A., Sidelnik, V., Smal, A., Ushakov, M. New Bounds on the Half-Duplex Communication Complexity // SOFSEM 2021. LNCS v. 12607. Springer, 2021. pp. 233–248. URL: <https://doi.org/10.1007/978-3-030-67731-2>
- [2] Dementiev, Y., Ignatiev, A., Sidelnik, V., Smal, A., Ushakov, M. New bounds on the half-duplex communication complexity // ECCC TR20-117 (2020). URL: <https://eccc.weizmann.ac.il/report/2020/117>.
- [3] Gladyshev, T. Personal communication to A. Smal (2021).
- [4] Hoover, K., Impagliazzo, R., Mihajlin, I., Smal, A. V. Half-Duplex Communication Complexity // ISAAC 2018. V. 123. — Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. — 10:1—10:12. — (LIPIcs). — <https://doi.org/10.4230/LIPIcs.ISAAC.2018.10>.
- [5] Hoover, K., Impagliazzo, R., Mihajlin, I., Smal, A. Half-duplex communication complexity // ECCC TR18-089 (2018). URL: <https://eccc.weizmann.ac.il/report/2018/089>.
- [6] Mauricio Karchmer and Avi Wigderson. Monotone Circuits for Connectivity Require Superlogarithmic Depth. In Janos Simon, editor, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, pages 539–550. ACM, 1988. doi:10.1145/62212.62265.
- [7] Kogan, A., Moskalenko, T., Prozorova, L., Shpraidun, L. Personal communication to A. Smal (2021).
- [8] Kuptsov, T. Personal communication (2023).

- [9] Artur Ignatiev, Ivan Mihajlin, Alexander Smal: Super-Cubic Lower Bound for Generalized Karchmer-Wigderson Games. ISAAC 2022 <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ISAAC.2022.66>
- [10] Or Meir: Toward Better Depth Lower Bounds: A KRW-like theorem for Strong Composition. FOCS 2023 <https://eccc.weizmann.ac.il/report/2023/078/>
- [11] Hao Wu: An Improved Composition Theorem of a Universal Relation and Most Functions via Effective Restriction. <https://eccc.weizmann.ac.il/report/2023/151/>
- [12] Meir, O. Toward Better Depth Lower Bounds: Two Results on the Multiplexor Relation // *Comput. Complex.* 29, 4 (2020). URL: <https://doi.org/10.1007/s00037-020-00194-8>.
- [13] Alexandre Nolin. Communication complexity : large output functions, partition bounds, and quantum nonlocality. *Computational Complexity [cs.CC]*. Université Paris Cité, 2020. English. NNT: 2020UNIP7201. tel-03342472. Ph.D. thesis (2021) <https://theses.hal.science/tel-03342472/document>
- [14] Eyal Kushilevitz, Noam Nisan, *Communication Complexity*, Cambridge University Press (2006)
- [15] Alexander Vladimirovich Smal. Proofs of lower bounds on the size of formulas for Boolean functions using communication complexity. Ph.D. thesis (2022).
- [16] Yao, A. C.-C. Some Complexity Questions Related to Distributive Computing (Preliminary Report) // *STOC 1979*. — ACM, 1979. — P. 209—213. — <http://doi.acm.org/10.1145/800135.804414>.