Multiparty communication complexity
and very hard functions

Pavol Řuriš*

Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University,
Mlynská dolina, 842 48 Bratislava, Slovakia

Received 23 March 1999; revised 4 March 2002
Available online 20 April 2004

Abstract

A boolean function \( f(x_1, \ldots, x_n) \) with \( x_i \in \{0, 1\}^m \) for each \( i \) is hard if its nondeterministic multiparty communication complexity (introduced in [in: Proceedings of the 30th IEEE FOCS, 1989, p. 428–433]), \( C(f) \), is at least \( nm \). Note that \( C(f) \leq nm \) for each \( f(x_1, \ldots, x_n) \) with \( x_i \in \{0, 1\}^m \) for each \( i \). A boolean function is very hard if it is hard and its complementary function is also hard. In this paper, we show that randomly chosen boolean function \( f(x_1, \ldots, x_n) \) with \( x_i \in \{0, 1\}^m \) for each \( i \) is very hard with very high probability (for \( n \geq 3 \) and \( m \) large enough). In [in: Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science, LNCS 900, 1995, p. 350–360], it has been shown that if \( f(x_1, \ldots, x_k, \ldots, x_n) = f_1(x_1, \ldots, x_k) \cdot f_2(x_{k+1}, \ldots, x_n) \), where \( C(f_1) > 0 \) and \( C(f_2) > 0 \), then \( C(f) = C(f_1) + C(f_2) \). We prove here an analogical result: If \( f(x_1, \ldots, x_k, \ldots, x_n) = f_1(x_1, \ldots, x_k) \oplus f_2(x_{k+1}, \ldots, x_n) \) then \( DC(f) = DC(f_1) + DC(f_2) \), where \( DC(g) \) denotes the deterministic multiparty communication complexity of the function \( g \) and “\( \oplus \)” denotes the parity function.

© 2004 Elsevier Inc. All rights reserved.

1. Introduction

In the two-party communication model, each of two processors has a part (half) of the input, and the goal is to compute a given boolean function on the input minimizing the amount
of communication. The multiparty model (introduced in [4]) generalizes the two-party model in such a way that the input \( (x_1, \ldots, x_n) \) is distributed among \( n \) processors (parties), where party \( i \) knows \( x_i \) and the goal is the same: to compute a given boolean function \( f(x_1, \ldots, x_n) \) on the input minimizing the total amount of communication. It is assumed that there is a coordinator that is allowed to communicate to each party, but the parties are not allowed to communicate (directly) among them.

Note that in [3] a different multiparty model was considered. In that model each of \( n \) parties has all the input expect one, and all parties communicate through a shared “blackboard.” This model was introduced in [2], where an interesting relation to time-space tradeoffs and branching programs was discovered. We do not consider this model here and by the “multiparty model” we denote in this paper the model introduced in [4].

The study of two-party communication was inspired by VLSI complexity. The relative power of determinism, nondeterminism, and randomization were the main studied issues [1,5,7,10–12]. Two-party communication with limited number of exchanged messages have been studied in [5,11].

Dolev and Feder [4] stated a challenge to obtain lower bounds for the multiparty model. For two-party communication, Yao [12] has introduced a method based on a crossing sequence argument (or on a fooling set argument) to bound the amount of information that needs to be exchanged. Šuriš and Rolim [6] have generalized the Yao’s method for multiparty communication model to derive (roughly) optimal lower and upper bounds on the multiparty communication complexity of some (simple) particular boolean functions. For example, it has been shown in [6] that the nondeterministic communication complexity of \( f, C(f) \), is at least \( nm \) for \( f(x_1, \ldots, x_n) = 1 \) iff \( x_1 = \cdots = x_n \), where \( x_i \in \{0,1\}^m \) for each \( i \). On the other hand, we will show here that \( C(g) \leq nm \) for each boolean function \( g(x_1, \ldots, x_n) \) with \( x_i \in \{0,1\}^m \) for each \( i \). Thus, \( f \) is a hard function from the point of view of communication complexity. Hence the following natural questions arise: How many hard functions are there? Is there a hard function of which the complementary function is also hard? (Call such functions very hard.) And, if yes, how many very hard function are there?

For two-party communication model introduced by Papadimitriou and Sipser [11], it has been shown that there are plenty of hard languages (i.e., languages with maximal possible communication complexity) for deterministic communication. Šuriš et al. [5] have generalized this result also for nondeterministic communication; (in fact, one can easily modify the proof of Theorem 4 of [5] to show that there are plenty of hard languages with hard complements).

In this paper, we answer the questions mentioned above (for multiparty communication) by showing that randomly chosen boolean function \( f(x_1, \ldots, x_n) \) with \( x_i \in \{0,1\}^m \) for each \( i \) is very hard with very high probability for \( n \geq 3 \) and \( m \) large enough. Note that some particular very hard functions have been constructed in [9]. Moreover, it has been shown in [9] that combining two very hard functions by \( \oplus \) (where “\( \oplus \)” denotes the parity function) results in a new very hard function.

It has been shown in [6] that if \( f(x_1, \ldots, x_k, \ldots, x_n) = f_1(x_1, \ldots, x_k) \cdot f_2(x_{k+1}, \ldots, x_n) \), where \( C(f_1) > 0 \) and \( C(f_2) > 0 \), then \( C(f) = C(f_1) + C(f_2) \). We prove here an analogical result for the deterministic communication: If \( f(x_1, \ldots, x_k, \ldots, x_n) = f_1(x_1, \ldots, x_k) \oplus f_2(x_{k+1}, \ldots, x_n) \) then \( DC(f) = DC(f_1) + DC(f_2) \), where \( DC(g) \) denotes the deterministic communication complexity of the function \( g \). It is not clear whether such result holds also for the logical “and” or for the logical “or.”

The following basic question about communication complexity is mentioned in [8]: Can we solve two problems simultaneously in a way that is better than to solve each of the two problems separately? Note that our second result solves this question (for the model considered in this paper),
since it is easy to adopt each protocol computing simultaneously \( f_1 \) and \( f_2 \) to a protocol computing \( f = f_1 \oplus f_2 \) without increasing communication complexity.

1.1. Definitions

To state our result more precisely, we first give several definitions. Let \( \epsilon \) be the empty string and let \( w = w_1w_2\ldots w_l \), \( l \geq 1 \), \( w_i \in \{0,1\}^+ \) for every \( i \). We define: \( h(\epsilon) = \epsilon \) and \( h(w) = w_1w_2\ldots w_l \).

Let \( r = (r_1,r_2,\ldots,r_t) \), \( t \geq 1 \), where either \( r_i = r_i^{0}S r_i^{1}S\ldots S r_i^{j_i} \), \( j_i \geq 1 \), or \( r_i = \epsilon \). We define:

\[
h(r) = h(r_1)h(r_2)\cdots h(r_t).
\]

We denote the length of a string \( w \) (the cardinality of a set \( S \)) by \( |w| \) (by \( |S| \)). If \( S \) is a set then by \( h(S) \) we denote the set \( \{h(s) | s \in S \} \).

Suppose a coordinator wishes to evaluate a function \( f(x_1,x_2,\ldots,x_n) \). The input vector \( x = (x_1,x_2,\ldots,x_n) \) is distributed among \( n \) parties (i.e., the processors \( p_1,p_2,\ldots,p_n \)), with \( x_i \) is known only to party \( i \), where \( x_i \) is chosen from \( \{0,1\}^m \) for every \( i \). Suppose there is a deterministic protocol \( P \) that accepts the language defined by \( f \) (when the value of \( f \) is 1). In such a case we will say that \( P \) computes \( f \). Generally, the computation of \( P \) consists of several phases, where one phase is as follows: The coordinator sends some messages (nonempty binary strings) to some parties (not necessary to all parties) and then, each party that got a message, sends a message back to the coordinator. The communication behaviour of \( P \) can be described by a communication vector \( s = (s_1,s_2,\ldots,s_n) \), where either \( s_i = s_i^0S s_i^1S\ldots S s_i^{j_i} \), \( j_i \geq 2 \), \( s_i^j \in \{0,1\}^+ \), or \( s_i = \epsilon \); \( s_i \) is a communication sequence between the coordinator and the party \( i \) (if there is no communication then \( s_i = \epsilon \)). Note that \( j_i \) is an even number (each party must respond after obtaining any nonempty message), and \( s_i^{2j_i-1}S s_i^{2j_i} \) is not necessary the message sent [received] by the coordinator in the phase \( l \) (since the coordinator may have sent no message to the party \( i \) in some previous phase \( k < l \)). We will also say “communication sequence on the link \( i \)” instead of “communication sequence between the coordinator and the party \( i \).” Also we will say “processor \( p_i \)” instead of “party \( i \).”

Formally, a deterministic protocol \( P \) is an \((n+1)\)-tuple of functions \((\phi_0,\phi_1,\ldots,\phi_n)\), for which the following holds:

Let

\[
K = \{0,1,\$\}^* \times \cdots \times \{0,1,\$\}^* \quad (n \text{ times}),
\]

\[
M = \{0,1\}^* \times \cdots \times \{0,1\}^* \quad (n \text{ times}).
\]

(a) \( \phi_0 \) is a function from \( K \) to \( M \cup \{\text{“accept,” “reject”}\} \). Intuitively, behaviour of the coordinator is given by \( \phi_0 \), where the argument of \( \phi_0 \) is a communication vector of all previous messages, with \( \$ \) serving as the delimiter between messages. The result of \( \phi_0 \) are either the next messages sent to the parties or the coordinator stops the communication and accepts/rejects the input.

(b) For \( i = 1,2,\ldots,n \), \( \phi_i \) is a function from \( \{0,1\}^m \times \{0,1,\$\}^* \) to \( \{0,1\}^+ \). Intuitively, behaviour of the party \( i \), \( (1 \leq i \leq n) \), is given by \( \phi_i \), where the first argument of \( \phi_i \) is the local input for the party \( i \) and the second argument of \( \phi_i \) is a sequence of all previous messages on the link \( i \) (delimited by \( \$ \)). The result of \( \phi_i \) is the next message sent by the party \( i \) to the coordinator.

A computation under \( P \) on input \( x = (x_1,\ldots,x_n) \) with \( x_i \in \{0,1\}^m \) for each \( i \) is a communication vector \( s^k = (s_1^k,\ldots,s_n^k) \), where \( k \geq 0 \), \( (k \) is the number of all phases performed on \( x \) under \( P \)), such that for every \( j = 0,1,2,\ldots,k-1 \) there is a communication vector \( s^j = (s_1^j,\ldots,s_n^j) \), \( (s^j \) is the communication vector after completing the \( j \)th phase of the computation on \( x \) under \( P \)), for which (c), (d), and (e) hold.
(c) $s_i^0 = \epsilon$ for $i = 1, 2, \ldots, n$; (the coordinator starts the communication with the empty communication vector $s^0 = (\epsilon, \ldots, \epsilon)$).

(d) For $j = 0, 1, 2, \ldots, k - 1$ it holds: Let $\phi_0(s^j) = (d_1^j, \ldots, d_n^j)$. Then $s_i^{j+1} = s_i^j \mathbin{\text{SD}}_i^j \mathbin{\text{S}} \phi_i(x_i, s_i^j \mathbin{\text{SD}}_i^j)$ if $d_i^j \neq \epsilon$, otherwise $s_i^{j+1} = s_i^j$ for $i = 1, 2, \ldots, n$.

(e) $\phi_0(s^k) \in \{\text{“accept,” “reject”}\}$. If $\phi_0(s^k) = \text{“accept”}$ [if $\phi_0(s^k) = \text{“reject”}$] then $s^k$ is an accepting [rejecting] communication vector under $P$, (or, $s^k$ is an accepting [rejecting] computation under $P$ on $x$).

We require that the nonempty communication sequences on each link are self-delimiting, i.e., if $s_i = s_i^1 \mathbin{\text{SD}}_i^2 \mathbin{\text{S}} \cdots \mathbin{\text{SD}}_i^k r_i$ and $r_i = r_i^1 \mathbin{\text{SD}}_i^2 \mathbin{\text{S}} \cdots \mathbin{\text{SD}}_i^k$ are any two different nonempty communication sequences on the link $i$ under $P$, and if $s_i^j = r_i^j, \ldots, s_i^q = r_i^q$ for some $q \geq 0$, then $q < \min\{j_i, l_i\}$ and $s_i^{j+1}$ is not any proper prefix of $r_i^{j+1}$, or vice versa. (Note that one can easy show that then $h(s_i) \neq h(r_i)$ and $h(s_i)$ is not any proper prefix of $h(r_i)$, or vice versa.)

In fact, we do not need the “end of transmission” symbol, “$\mathbin{\text{S}}$,” because of the self-delimiting property (introduced in [11]). We use this property, since we want to pin down exactly the communication complexity.

Let $f(x_1, \ldots, x_n)$ be a boolean function with $x_i \in \{0, 1\}^m$ for each $i$, and $P$ be a deterministic protocol. We say that $P$ computes $f$ if, for each $x = (x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}^m$ for each $i$, the computation under $P$ on input $x$ is an accepting one iff $f(x) = 1$.

Let $S$ be the set of all accepting and rejecting communication vectors under $P$. By $DC(f)$ we denote the maximum over all $s \in S$ of $|h(s)|$ minimized over all deterministic protocols computing $f$. $DC(f)$ is called the deterministic communication complexity of $f$.

We also consider nondeterministic protocols. In such a case, $\phi_i$’s are “nondeterministic functions,” i.e., they may have several values (and therefore they are not any functions). Moreover, they may be “partial nondeterministic functions,” i.e., they may be not defined for all possible values of arguments; in such a case, the current communication is aborted. We can apply the definitions above also for nondeterministic protocol in such a way that whenever we write $\phi_0(s) [\phi_i(x, s)$ for $i > 0]$ we mean a possible value of $\phi_0(s)$ [of $\phi_i(x, s)$]. We require the self-delimiting property also for nondeterministic protocols.

Let $f(x_1, \ldots, x_n)$ be a boolean function with $x_i \in \{0, 1\}^m$ for each $i$, and $P$ be a nondeterministic protocol. We say that $P$ computes $f$ if, for each $x = (x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}^m$ for each $i$, there is an accepting computation under $P$ on input $x$ iff $f(x) = 1$.

Let $A$ be the set of all accepting communication vectors under a nondeterministic protocol $P$. By $C(f)$ we denote the maximum over all $s \in A$ of $|h(s)|$ minimized over all nondeterministic protocols computing $f$. $C(f)$ is called the nondeterministic communication complexity of $f$.

A protocol is simple if each computation under it on each input consists of at most one phase during which the coordinator sends at most one bit to each party.

Let $P$ be a protocol computing a function $f(x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}^m$ for each $i$. Let $k$ be any integer with $1 \leq k \leq n$ and let $b$ be any bit in $\{0, 1\}$. Let $A$ be the set of all accepting communication vectors under $P$ and let

$$D = \{d|d \in \{0, 1\}^+ \text{ and there is } (s_1, \ldots, s_k, \ldots, s_n) \in A \text{ with } s_k = b \mathbin{\text{SD}}d\}.$$ 

$P$ is nice on the link $k$ for the bit $b$ if $|D| \leq (nm - 2)2^m$. $P$ is nice if it is nice on every link $k = 1, 2, \ldots, n$ for each bit $b \in \{0, 1\}$. 
2. Very hard functions

In this section we prove our main result: Randomly chosen boolean function \( f(x_1, \ldots, x_n) \) with \( x_i \in \{0, 1\}^m \) for each \( i \) is very hard with very high probability (for \( n \geq 3 \) and \( m \) large enough).

To prove the main result, we need two lemmas characterizing and simplifying nondeterministic multiparty communications. However, we first state an optimal upper bound on the nondeterministic multiparty communication complexity.

**Claims 1.** \( C(f) \leq nm \) for each boolean function \( f(x_1, \ldots, x_n) \) with \( x_i \in \{0, 1\}^m \) for \( i = 1, 2, \ldots, n \).

**Proof.** For each \( f \) under consideration, there is a nondeterministic protocol \( P \) computing \( f \) using \( nm \) exchanged bits as follows. Given input \( (x_1, \ldots, x_n) \), the coordinator nondeterministically guesses the first bit of \( x_i \) and sends it to \( p_i \) for \( i = 1, 2, \ldots, n \). Then each \( p_i \) responds the rest of \( x_i \) if the guess was successful, otherwise it aborts the communication. If all guesses were successful then the coordinator knows the input, and hence it can accept the input correctly.

Let \( f(x_1, \ldots, x_n) \) be any boolean function with \( x_i \in \{0, 1\}^m \) for each \( i \) and let \( P \) be any nondeterministic protocol computing \( f \). For \( f \) let us define the sets \( A \), \( C_i \)'s and \( X_i^C \)'s as follows. Let \( A \) be the set of all accepting communication vectors under \( P \). For \( i = 1, 2, \ldots, n \), let

\[
C_i = \{ e | e = c_i \text{ for some } (c_1, \ldots, c_i, \ldots, c_n) \in A \}.
\]

For \( i = 1, 2, \ldots, n \) and for each \( e \in C_i \), let

\[
X_i^e = \{ y | y \in \{0, 1\}^m \text{ and some input } (x_1, \ldots, x_i, \ldots, x_n) \text{ with } y = x_i \text{ is accepted by some } (c_1, \ldots, c_i, \ldots, c_n) \in A \text{ with } e = c_i \}.
\]

**Lemma 1.** Let \( f(x_1, \ldots, x_n) \) be any boolean function with \( x_i \in \{0, 1\}^m \) for \( i = 1, 2, \ldots, n \), let \( P \) be any nondeterministic protocol computing \( f \), and let \( A \) and \( X_i^C \)'s be the sets defined for \( P \) as above. Then

\[
\{(x_1, \ldots, x_n) | f(x_1, \ldots, x_n) = 1 \} = \bigcup_{(c_1, \ldots, c_n) \in A} X_1^{c_1} \times \cdots \times X_n^{c_n}.
\]

**Proof.** To prove the inclusion “\( \subseteq \)”, choose any \( x = (x_1, \ldots, x_n) \) with \( f(x_1, \ldots, x_n) = 1 \). Since \( P \) accepts \( f \), \( x \) has to be accepted by a communication vector \( (c_1, \ldots, c_n) \in A \) under \( P \). But it means that \( x_i \in X_i^{c_i} \) for \( i = 1, 2, \ldots, n \).

To prove the symmetric inclusion, choose any \( c = (c_1, \ldots, c_n) \in A \) and any \( x = (x_1, \ldots, x_n) \in X_1^{c_1} \times \cdots \times X_n^{c_n} \). Let \( x' = (x'_1, \ldots, x'_n) \) be any input accepted by \( c \) under \( P \), let \( x^0 = x' \) and let \( x^i = (x_1, \ldots, x_i, x_{i+1}', \ldots, x_n') \) for \( i = 1, 2, \ldots, n \). Assume that some \( x^i \) \((0 \leq i \leq n - 1) \) is accepted by \( c \) under \( P \). Hence \( c_{i+1} \) is a possible communication between the coordinator and \( p_{i+1} \) owning \( x_{i+1}' \) under \( P \). Since \( x_{i+1} \in X_i^{c_{i+1}} \), there is an input \( (x''_1, \ldots, x''_{i+1}, x_{i+1}', \ldots, x_n') \) that is accepted by a communication vector \( (c'_1, \ldots, c'_i, c_{i+1}, c_{i+2}, \ldots, c''_n) \) under \( P \). Thus, \( c_{i+1} \) is a possible communication between the coordinator and \( p_{i+1} \) owning \( x_{i+1} \) under \( P \). But it means that if we replace \( x_{i+1}' \) of \( x^i \) by \( x_{i+1} \) then the resulting input \( x^{i+1} \) has to be accepted by \( c \) under \( P \), since the coordinator is not able to recognize this replacement because of the same communication (i.e., crossing sequence) \( c_{i+1} \).
between the coordinator and $p_{i+1}$ owning $x'_{i+1}$ and between the coordinator and $p_{i+1}$ owning $x_{i+1}$, (see above). Consequently, one can observe that every input $x^1, x^2, \ldots, x^n$ is accepted by $c$ under $P$, since $x^0 = x'$ is accepted by $c$ under $P$, (see above). Hence $f(x_1, \ldots, x_n) = 1$, since $x^n$ is accepted by $c$ under $P$, where $x^n = x = (x_1, \ldots, x_n)$, and $P$ computes $f$. □

Lemma 2. For each boolean function $f(x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}^m$ for $i = 1, 2, \ldots, n$ there is a non-deterministic simple protocol computing $f$ using at most $C(f)$ exchanged bits.

Proof. Let $f$ be any boolean function under consideration and let $P$ be any non-deterministic protocol computing $f$ using at most $C(f)$ exchanged bits. We can simulate $P$ by a simple non-deterministic protocol $P'$ as follows. The coordinator under $P'$ non-deterministically chooses any communication vector (say $c = (c_1, \ldots, c_n)$) under $P$ and then it sends to $p_i$ the first bit of $c_i$ (if $c_i$ is not any empty communication) for $i = 1, 2, \ldots, n$. Then each $p_i$ (owning some $x_i$) that got a bit non-deterministically chooses any communication (say $c'_i = s'_i \bar{s}_i^2 \bar{s}_i^3 \cdots \bar{s}_i^n$) with the same first bit as $c_i$ that is possible on the link $i$ under $P$ with respect to $x_i$, (if there is no such $c'_i$ then $p_i$ aborts the communication), and finally, $p_i$ sends to the coordinator the string $s_i = z_i^1 z_i^2 \cdots z_i^n$, where $z_i^1$ is the string $s_i^1$ without the first bit. The coordinator is able to restore $c'_i$ from $s_i$, since it knows the first bit of $s_i^1$ and the nonempty communications are self-delimiting on each link. The coordinator accepts the input if $(c'_1, \ldots, c'_n)$ is an accepting communication vector under $P$, where $c'_i = c_i$ ($1 \leq i \leq n$) if $c_i$ is the empty communication. □

Theorem 1. For each integer $n \geq 3$ and for each real number $\alpha$, $0 < \alpha < 1$, there is a positive integer $m$ such that randomly chosen boolean function $f(x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}^m$ for $i = 1, 2, \ldots, n$ is very hard with probability at least $\alpha$.

Proof. We first explain an idea of the proof. Let $F_{n,m}$ denote the set of all boolean functions $f(x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}^m$ for $i = 1, 2, \ldots, n$ and with $C(f) \leq nm - 1$. For each $f \in F_{n,m}$ we will choose any non-deterministic simple nice protocol $P_f$ computing $f$ with at most $nm - 1$ exchanged bits (there is such $P_f$ for each $f \in F_{n,m}$, see Corollary 1 of Lemma 3 below) and for $P_f$ we will define a characteristic sequence of $f$ (see below) that fully describes $f$. Using the fact that $P_f$'s are simple nice protocols, we will show that the number of characteristic sequences corresponding to all chosen protocols $P_f$ is (for $m$ large enough) only a small fraction of the number of all boolean functions $f(x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}^m$ for all $i$. This result will enable us to complete the proof of Theorem 1 very easily. □

Now let us prove Theorem 1. To do so we need the following lemma and its corollary.

Lemma 3. Let $f(x_1, \ldots, x_n)$ be any boolean function with $x_i \in \{0, 1\}^m$ for $i = 1, 2, \ldots, n$ and with $C(f) \leq nm - 1$. Let $P$ be any non-deterministic simple protocol computing $f$ with at most $nm - 1$ exchanged bits. Let $k$ be any integer with $1 \leq k \leq n$ and let $b$ be any bit in $\{0, 1\}$. Then there is a non-deterministic simple protocol $P'$ computing $f$ with at most $nm - 1$ exchanged bits that is nice on the link $k$ for the bit $b$. Moreover, if $P$ is nice on some link $k'$, ($1 \leq k' \leq n$), for some bit $b' \in \{0, 1\}$ then $P'$ is nice on the link $k'$ for the bit $b'$, too.
Proof. Given $P$, $k$, and $b$, let us define sets $A$, $D$, and $D_i$'s as follows. Let $A$ be the set of all accepting communication vectors under $P$ and let

$$D = \{d | d \in \{0, 1\}^+ \text{ and there is } (c_1, \ldots, c_k, \ldots, c_n) \in A \text{ with } c_k = bSd\}.$$ 

Note that $1 \leq |d| \leq nm - 2$ for each $d \in D$, since $d \in \{0, 1\}^+ \text{, } |b| = 1$, and $\sum_{i=1}^{n} |h(c_i)| \leq nm - 1$ for each $(c_1, \ldots, c_n) \in A$. For $i = 1, 2, \ldots, nm - 2$, let

$$D_i = \{d | d \in \{0, 1\}^i \text{ and } d \text{ is a prefix of a string in } D\}.$$ 

Hence, $D \subseteq \bigcup_{i=1}^{nm-2} D_i$, since $1 \leq |d| \leq nm - 2$ for each $d \in D$ (see above), and each string $d$ is a prefix of $d$.

If $|D| < (nm - 2)2^m$ then we set $P'$ to be $P$ and we have done, since in such a case $P$ is nice on the link $k$ for the bit $b$.

Now let us suppose that $|D| \geq (nm - 2)2^m$. It means that there is an index $j$, $(1 \leq j \leq nm - 2)$, with $|D_j| \geq 2^m$, since otherwise $(nm - 2)2^m \leq |D| \leq \bigcup_{i=1}^{nm-2} D_i \leq \sum_{i=1}^{nm-2} |D_i| < (nm - 2)2^m$, a contradiction. Let $j$ be such minimal index.

Now we are ready to construct the desired protocol $P'$ by the following modification of $P$. We modify only behaviour of $pk$ after receiving the bit $b$ (from the coordinator) so that $pk$ may use for responding (arbitrarily chosen) $2^m$ strings of $D_j$ (we denote the set of these $2^m$ strings by $D'j$) to encode its $2^m$ possible local inputs $x_k \in \{0, 1\}^m$, and hence to enable the coordinator to know a particular local input $x_k$. Then the coordinator (knowing communications on the other links) has enough information to decide correctly whether to accept the input or not. (We will determine below which communication vectors under $P'$ are accepting ones.) Note that $pk$ under $P'$ does not use any other response (excluding $2^m$ strings in $D'j$) of the length at least $j$ after receiving $b$. Moreover, behaviour of $pk$ after receiving $b$ is unchanged by our modification when $pk$ responds messages shorter than $j$.

Let us prove that $P'$ computes $f$ using at most $nm - 1$ exchanged bits. We first determine accepting communication vectors under $P'$ as follows. Each accepting communication vector $c = (c_1, \ldots, c_k, \ldots, c_n)$ under $P$, where $c_k$ is not of the form $bSd$ with $|d| \geq j$, is an accepting one under $P'$. Thus the following holds for each input $x$ and each $c = (c_1, \ldots, c_k, \ldots, c_n)$, where $c_k$ is not of the form $bSd$ with $|d| \geq j$: $x$ is accepted by $c$ under $P'$ if $x$ is accepted by $c$ under $P$, since behaviour of $pk$ under $P$ and $P'$ is the same in such a case, (see modification of $P$ above). A communication vector $c = (c_1, \ldots, c_k, \ldots, c_n)$, where $c_k = bSd$, $d \in D'$, is an accepting one under $P'$ if there is $x_k \in \{0, 1\}^m$ and an accepting communication vector $c' = (c_1, \ldots, c_{k-1}, bSd', c_{k+1}, \ldots, c_n)$ under $P$ with $|d'| \geq j$ such that $d$ encodes $x_k$ and $pk$ owning $x_k$ under $P$ can respond $d'$ after receiving $b$. Moreover, there are no other accepting communication vectors under $P'$. Hence, the following holds for each input $x = (x_1, \ldots, x_k, \ldots, x_n)$: If $x$ is accepted by a $c = (c_1, \ldots, c_{k-1}, bSd, c_{k+1}, \ldots, c_n)$ under $P'$, where $d \in D'$, then $d$ encodes $x_k$ (see modification of $P$ above), and therefore $x$ is accepted by some $c' = (c_1, \ldots, c_{k+1}, bSd', c_{k+1}, \ldots, c_n)$ under $P$ with $|d'| \geq j$, where $h(c) \leq h(c')$, (since $|d| = j \leq |d'|$); moreover, if $x$ is accepted by an $e = (e_1, \ldots, e_{k-1}, bSg, e_{k+1}, \ldots, e_n)$ under $P$, where $|g| \geq j$, then $x$ is accepted by $e' = (e_1, \ldots, e_{k-1}, bSg, e_{k+1}, \ldots, e_n)$ under $P'$, where $d \in D'$, $d$ encodes $x_k$ and $h(e) \geq h(e')$, since $pk$ owning $x_k$ under $P'$ is able to respond $d$ encoding $x_k$ after receiving $b$, (see modification of $P$ above). Consequently, the results above yield that $P'$ computes $f$ using at most $nm - 1$ exchanged bits, since $P$ does so.
Now let us show that \( P' \) is nice on the link \( k \) for the bit \( b \). Let \( c = (c_1, \ldots, c_{k-1}, bSd, c_{k+1}, \ldots, c_n) \) be any accepting communication vector under \( P' \). We first assume that \(|d| < j\). Then \( c \) is an accepting communication vector under \( P \) (see above), i.e., \( c \in A \). Thus \( d \in D \), more precisely, \( d \in \bigcup_{i=1}^{j-1} D_i \) (see above) and \(|d| < j\). Now we assume that \(|d| \geq j\). Then \( d \in D' \), since \( p_k \) does not use under \( P' \) after receiving \( b \) any other response of the length at least \( j \) excluding strings in \( D' \), (see modification of \( P \) above). Consequently, there is at most \((j-1)2^m + 2^m \leq (nm - 2)2^m \) possible \( d \)'s for which there is an accepting communication vector under \( P' \) of the form \((c_1, \ldots, c_{k-1}, bSd, c_{k+1}, \ldots, c_n)\), since each such \( d \) belongs to \( \bigcup_{i=1}^{j-1} D_i \cup D' \), where \(|\bigcup_{i=1}^{j-1} D_i| \leq \sum_{i=1}^{j-1} |D_i| \leq (j-1)2^m \) (because of minimality of \( j \)), and \(|D'| = 2^m \). Thus \( P' \) is nice on the link \( k \) for the bit \( b \).

Finally, if \( P \) is nice on some link \( k' \) for some bit \( b' \) then \( P' \) is nice on the link \( k' \) for the bit \( b' \), too, since behaviour of \( p_k \) after receiving \( b' \) has not been changed by our modification for \( k' \neq k \) or \( b' \neq b \), and we have shown above that \( P' \) is nice on the link \( k \) for the bit \( b \).

This completes the proof of Lemma 3. \( \square \)

**Corollary 1.** Let \( f(x_1, \ldots, x_n) \) be any boolean function with \( x_i \in \{0,1\}^m \) for \( i = 1, 2, \ldots, n \) and with \( C(f) \leq nm - 1 \). Then there is a nondeterministic simple nice protocol computing \( f \) with at most \( nm - 1 \) exchanged bits.

**Proof.** By Lemma 2, there is a nondeterministic simple protocol \( P_f \) computing \( f \) with at most \( nm - 1 \) exchanged bits. Let \( P_0 = P_f \). Applying Lemma 3 \( 2n \) times, one can prove that there are protocols \( P_1, P_2, \ldots, P_{2n} \) (note that \( P, P' \) of Lemma 3 are \( P_{i-1} \) and \( P_i \), respectively, when Lemma 3 is applied the \( i \)th times) satisfying the following properties for \( i = 1, 2, \ldots, 2n \). \( P_i \) is a nondeterministic simple protocol computing \( f' \) with at most \( nm - 1 \) exchanged bits, \( P_i \) is nice on the link \( \lceil i/2 \rceil \) for the bit \( b \), where \( b = 0 \) if \( i \) is odd and \( b = 1 \) if \( i \) is even, and if \( P_{i-1} \) is nice on some link \( j \) for some bit \( b' \) then \( P_i \) is nice on the link \( j \) for the bit \( b' \), too.

It means that if \( P_i \) is nice on some link \( k \) for some bit \( b \), then each \( P_i \) with \( i \geq l \) is nice on the link \( k \) for the bit \( b \), too. Hence \( P_{2n} \) is our desired simple nice protocol computing \( f \) with at most \( nm - 1 \) exchanged bits. \( \square \)

Now we are ready to complete the proof of Theorem 1. Let \( F_{n,m} \) be the set of all boolean functions \( f(x_1, \ldots, x_n) \) with \( x_i \in \{0,1\}^m \) for \( i = 1, 2, \ldots, n \) and with \( C(f) \leq nm - 1 \). Let us bound \(|F_{n,m}|\). For each \( f \in F_{n,m} \) choose any nondeterministic nice simple protocol \( P_f \) computing \( f \) with at most \( nm - 1 \) exchanged bits. (By Corollary 1, there is such \( P_f \) for each \( f \in F_{n,m} \).) For each chosen \( P_f \) let us define a characteristic sequence \( B, C_0^0, \ldots, C_0^0, g_1, \ldots, g_n, Z_1, \ldots, Z_n \) of \( f \) (we will see that the characteristic sequence of \( f \) fully describes \( f \)) as follows.

Let \( A \) and \( X_i^e \)'s be the sets from Lemma 1 defined for \( P_f \). Let

\[
A_0 = \{(c_1, \ldots, c_n)| (c_1, \ldots, c_n) \in A \text{ and } h(c_i) \neq e \text{ for } i = 1, 2, \ldots, n\},
\]

\[
B = \{w0^{nm-1-|w|}|w = h(c_1) \cdots h(c_n) \text{ for some } (c_1, \ldots, c_n) \in A_0\}.
\]

For \( i = 1, 2, \ldots, n \), let

\[
C_i^0 = \{e| e = c_i \text{ for some } (c_1, \ldots, c_i, \ldots, c_n) \in A_0\}.
\]
For \( i = 1, 2, \ldots, n \), let us define a function
\[ g_i : \{1, 2, \ldots, |C_i^0|\} \to \{X | X \subseteq \{0, 1\}^m\}, \]
so that \( g_i(j) = X_i^j \) if the \( j \)th element (lexicographically) of \( C_i^0 \) is \( e, (1 \leq j \leq |C_i^0|) \). Partition arbitrarily the set \( A - A_0 \) into \( n \) subsets \( A_1, A_2, \ldots, A_n \) so that \( \bigcup_{i=1}^{n} A_i = A - A_0 \) and if \( (c_1, \ldots, c_n) \in A_i \) then \( h(c_i) = e \) for \( i = 1, 2, \ldots, n \). For \( i = 1, 2, \ldots, n \), let
\[ Z_i = \bigcup_{(c_1, \ldots, c_n) \in A_i} X_1^{c_1} \times \cdots \times X_n^{c_n}. \]
By Lemma 1,
\[ \{(x_1, \ldots, x_n) | f(x_1, \ldots, x_n) = 1\} = \bigcup_{(c_1, \ldots, c_n) \in A_0} X_1^{c_1} \times \cdots \times X_n^{c_n} \cup \bigcup_{i=1}^{n} Z_i. \]
(1)

Now one has to realize the following important fact. If we know the sets \( B, C_1^0, \ldots, C_n^0 \) then we are able to construct the set \( A_0 \) as follows. It is easy to see that \( A_0 \) is the empty set iff the sets \( B \) and \( C_i^0 \)'s are empty. Hence, it is easy to construct \( A_0 \) if the sets \( B \) and \( C_i^0 \)'s are empty. Now let us suppose that \( B \) and \( C_i^0 \)'s are nonempty. Thus \( A_0 \) is nonempty. In such a case, we can construct \( A_0 \) as follows. One can show (by induction on \( j \)) that for every \( j = 1, 2, \ldots, n \) and for each \( y \in B \) there is exactly one sequence \( c_1, c_2, \ldots, c_j \) with \( c_i \in C_i^0 \) for \( i = 1, 2, \ldots, j \) such that \( h(c_1)h(c_2) \cdots h(c_j) \) is a prefix of \( y \), since no set \( h(C_i^0) \) contains any empty string (see the definitions of the sets \( A_0 \) and \( C_i^0 \) above) and the nonempty strings of each set \( h(C_i^0) \) satisfy the prefix-free property (see above). Hence
\[ A_0 = \{(c_1, \ldots, c_n) | c_i \in C_i^0 \quad \text{for} \quad i = 1, 2, \ldots, n \quad \text{and} \quad h(c_1) \cdots h(c_n) \text{ is a prefix of some string in } B \}. \]

But it means that if we know \( B, C_1^0, \ldots, C_n^0, g_1, \ldots, g_n, Z_1, \ldots, Z_n \) then we are able to construct the set \( \bigcup_{(c_1, \ldots, c_n) \in A_0} X_1^{c_1} \times \cdots \times X_n^{c_n} \cup \bigcup_{i=1}^{n} Z_i \). Thus, by (1), \( f \) is fully described by the characteristic sequence \( B, C_1^0, \ldots, C_n^0, g_1, \ldots, g_n, Z_1, \ldots, Z_n \) of \( f \).

Hence to bound \( |F_{n,m}| \) it is enough to bound the number of all possible different characteristic sequences of all functions in \( F_{n,m} \). Now let us bound this number.

There is at most \( 2^{2^{nm-1}} \) possible different sets \( B \), since each of them is a subset of \( \{0, 1\}^{nm-1} \).

Let \( q = (nm - 2)2^{m+1} \). Since each chosen \( P_f \) is a simple nice protocol, each possible set \( C_i^0 \) contains strings of the form \( bSd \) with at most \( (nm - 2)2^m \) possible different strings \( d \) for each \( b \in \{0, 1\} \), i.e., \( |C_i^0| \leq (nm - 2)2^m + 1 \). Moreover, there is less than \( 2^{nm} \) possible different strings of the form \( bSd \) occurred in all possible sets \( C_i^0 \), since \( b \in \{0, 1\} \) and each chosen \( P_f \) accepts inputs with at most \( nm - 1 \) exchanged bits, i.e., \( |d| \leq nm - 1 - |b| = nm - 2 \). Hence, there is less than \( \sum_{i=0}^{q} 2^m \) possible different sets \( C_i^0 \) for each \( i \), and there is at most \( \sum_{i=0}^{q} 2^m \) possible different functions \( g_i \) for each \( i \).

If an input \( x = (x_1, \ldots, x_i, \ldots, x_n) \) is accepted by a communication vector \( c = (c_1, \ldots, c_i, \ldots, c_n) \) without any communication on the link \( i \) (i.e., \( h(c_i) = e \)) then clearly each input \( x' = (x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n) \) with \( x'_i \in \{0, 1\}^m \) will be accepted by \( c \), too; hence if \( x \in Z_i \) then each such \( x' \in Z_i \). But it means that there is at most \( 2^{2^{(n-1)m}} \) possible different sets \( Z_i \) for each \( i \).
Since each \( f \in F_{n,m} \) is fully described by the characteristic sequence of \( f \) (see above), the bounds above yield that
\[
|F_{n,m}| < 2^{2^{n-1}} \left( 2^{\lceil \log_2 m \rceil^2} 2^{\lceil \log_2 m \rceil + 1} \right)^n \left( 2^{mnm+1} \right)^n \left( 2^{|\phi|} \right)^n.
\]
(2)

Let \( G_{n,m} \) denote the set of all boolean functions \( f(x_1,\ldots,x_n) \) with \( x_i \in \{0,1\}^m \) for \( i = 1,2,\ldots,n \). If neither \( f \) nor \( 1-f \) belongs to \( F_{n,m} \), where \( f \in G_{n,m} \), then \( f \) is very hard. Thus there is at least \( |G_{n,m}| - 2|F_{n,m}| \) very hard functions in \( G_{n,m} \). By (2), \( \lim_{m \to \infty} (|G_{n,m}| - 2|F_{n,m}|)/|G_{n,m}| = 1 \) for \( n \geq 3 \), since \( |G_{n,m}| = 2^{2nm} \).

This completes the proof of Theorem 1. \( \square \)

3. Deterministic complexity of parity composition

It has been shown in [6] that if \( f(x_1,\ldots,x_k,\ldots,x_n) = f_1(x_1,\ldots,x_k) \cdot f_2(x_{k+1},\ldots,x_n) \), where \( C(f_1) > 0 \) and \( C(f_2) > 0 \), then \( C(f) = C(f_1) + C(f_2) \). In this section we prove an analitical result for the deterministic communication.

**Theorem 2.** Let \( f(x_1,\ldots,x_k,\ldots,x_n) = f_1(x_1,\ldots,x_k) \oplus f_2(x_{k+1},\ldots,x_n) \). Then \( DC(f) = DC(f_1) + DC(f_2) \).

**Proof.** The inequality “\( \leq \)” is obvious. To prove the symmetric inequality assume to the contrary that there is a deterministic protocol \( P \) computing \( f \) with less than \( DC(f_1) + DC(f_2) \) exchanged bits and derive a contradiction by showing that then there is a deterministic protocol \( P_1 \) computing \( f_1 \) with less than \( DC(f_1) \) exchanged bits or there is a deterministic protocol \( P_2 \) computing \( f_2 \) with less than \( DC(f_2) \) exchanged bits or \( P \) uses on some input at least \( DC(f_1) + DC(f_2) \) exchanged bits. \( \square \)

To derive the desired contradiction we need to introduce the following notions. Let \( x = (x_1,\ldots,x_n) \) be any input for \( P \) and let \( u_i \) [let \( v_i \)] be the communication vector over the links 1,2,\ldots,k [the links \( k+1,k+2,\ldots,n \)] during the \( i \)th phase of the computation under \( P \) on \( x \). Thus the communication vector under \( P \) on \( x \) is fully described by the sequence \( (u_1,v_1), (u_2,v_2), \ldots, (u_l,v_l) \), where \( l \) is the number of all phases performed under \( P \) on \( x \). A prefix of a sequence \( s = (u_1,v_1), (u_2,v_2), \ldots, (u_l,v_l) \) is a sequence \( (u_1,v_1), (u_2,v_2), \ldots, (u_i,v_i) \) for any \( 0 \leq i \leq l \) (note that \( s \) is the empty sequence for \( i = 0 \)), and by the sequence \( s, (u_{l+1},v_{l+1}) \) we denote the sequence \( (u_1,v_1), (u_2,v_2), \ldots, (u_l,v_l), (u_{l+1},v_{l+1}) \). Let \( S \) be the set of all sequences of the form \( (u_1,v_1), (u_2,v_2), \ldots, (u_l,v_l) \) corresponding to all possible inputs \( (x_1,\ldots,x_n) \) for \( P \) (note that \( l \) depends on a particular input for \( P \) and it denotes the number of all phases performed under \( P \) on the particular input), and let \( Q \) be the set of all possible prefixes of all sequences of \( S \).

Now let as classify each sequence of \( Q \) by the values 1-easy or 1-hard and 2-easy or 2-hard as follows. If the empty sequence belongs to \( S \) then it is 1-hard and also 2-hard. Let \( s = (u_1,v_1), \ldots, (u_l,v_l) \) be any sequence in \( S \) with \( l \geq 1 \). If \( \sum_{j=1}^l h(u_j) < DC(f_1) \) then \( s \) is 1-easy, otherwise \( s \) is 1-hard, and if \( \sum_{j=1}^l h(v_j) < DC(f_2) \) then \( s \) is 2-easy, otherwise \( s \) is 2-hard. Having classified all sequences of \( S \), we can classify (inductively) each sequence of \( Q - S \) as follows. Suppose we have classified all sequences of \( Q \) with at least \( i \) couples for some \( i \geq 1 \). Let \( s = (u_1,v_1), \ldots, (u_{l-1},v_{l-1}) \) be any sequence in \( Q - S \).
(Note that \(s\) is the empty sequence for \(i = 1\).) If for each sequence \(s, (u_i, v_i)\) in \(Q\) there is an 1-easy sequence of the form \(s, (u_i, v_i)\) then \(s\) is 1-easy, otherwise \(s\) is 1-hard. If there is a sequence \(s, (u_i, v_i)\) in \(Q\) such that each sequence of the form \(s, (u_i, v_i)\) in \(Q\) is 2-easy then \(s\) is 2-easy, otherwise \(s\) is 2-hard.

To derive the desired contradiction mentioned above we have to consider the following three cases.

**Case 1.** The empty sequence is 1-easy. In such a case we derive a contradiction by showing that there is a deterministic protocol \(P_1\) computing \(f_1\) with less than \(DC(f_1)\) exchanged bits. First we describe behaviour of \(P_1\). Behaviour of \(p_j\) is the same under \(P_1\) and under \(P\) for \(j = 1, 2, \ldots, k\). Let \(x = (x_1, \ldots, x_k)\) be any input for \(P_1\). The main strategy of the coordinator under \(P_1\) on \(x\) is to perform a communication described by a sequence \(u_1, u_2, \ldots, u_t\) (one \(u_i\) per a phase) and to select a sequence \(v_1, v_2, \ldots, v_t\) (one \(v_i\) per a phase) so that the sequence \((u_1, v_1), (u_2, v_2), \ldots, (u_t, v_t)\) may be 1-easy for \(i = 1, 2, \ldots, t\) and the sequence \((u_1, v_1), (u_2, v_2), \ldots, (u_t, v_t)\) may belong to \(S\).

More particularly, behaviour of \(P_1\) on \(x\) during the \(i\)th phase is as follows. Suppose that \(P_1\) on \(x\) has performed a communication described by a sequence \(u_1, u_2, \ldots, u_{i-1}\) and the coordinator under \(P_1\) on \(x\) has selected a sequence \(v_1, v_2, \ldots, v_{i-1}\) after completing the \((i-1)\)st phase, where \(s = (u_1, v_1), (u_2, v_2), \ldots, (u_{i-1}, v_{i-1})\) is an 1-easy sequence. (Note that \(s\) is the empty sequence for \(i = 1\).) If \(s \in S\) then the coordinator stops the communication, (i.e., \(t = i - 1\)), and it accepts (if \(f_1(x) = 1\)) or rejects (if \(f_1(x) = 0\)) the input \(x\). (However, we have to guarantee the following property (i.e., correctness) of \(P_1\): If there is any other input \(x'\) with the same communication vector (described by the sequence \(u_1, u_2, \ldots, u_{i-1}\) then it must hold \(f_1(x) = f_1(x')\), since the coordinator cannot see the inputs \(x, x'\)—it only knows the same communication vector—and hence it accepts both \(x\) and \(x'\) or rejects them. We will show this property below.) Now suppose that \(s \in Q - S\). The coordinator under \(P_1\) on \(x\) sends the same messages via the links \(1, 2, \ldots, k\) during the \(i\)th phase as it sends via these links under \(P\) with respect to the communication vector described by \(s\). (Note that these messages are fully determined only by \(P\) and \(s\).) Then \(p_1, \ldots, p_k\) respond messages under \(P_1\) on \(x\) during the \(i\)th phase. (Recall that behaviour of \(p_j\) is the same under \(P_1\) and under \(P\) for \(1 \leq j \leq k\).) Let \(u_i\) denote the communication vector under \(P_1\) on \(x\) during the \(i\)th phase. Finally the coordinator under \(P_1\) on \(x\) selects such \(v_i\) during the \(i\)th phase that the sequence \(s, (u_i, v_i)\) is 1-easy. (We will show below (see (b) and (c) of Lemma 4) that there is at least one such \(v_i\); note that the coordinator selects always the same \(v_i\) given \(s\) and \(u_i\) if there are several possible \(v_i\)‘s, and this guarantees that \(P_1\) is a deterministic protocol.)

To show the existence of at least one desired \(v_i\) mentioned above and to prove correctness of \(P_1\), we need the following lemma.

**Lemma 4.** Let \(x = (x_1, \ldots, x_k)\) be any input for \(P_1\) and let \(s = (u_1, v_1), \ldots, (u_l, v_l)\) be any 1-sequence in \(Q\), where \(l \geq 1\), such that \(u_i\) is performed and \(v_i\) is selected during the \(i\)th phase of the computation under \(P_1\) on \(x\) for \(i = 1, 2, \ldots, l\). Let \(s'\) be any sequence in \(S\) with the prefix \(s\) and let \(x' = (x'_1, \ldots, x'_n)\) be any input for \(P\) with the communication vector described by \(s'\). Then (a), (b), and (c) hold.

(a) The communication vector over the links \(1, 2, \ldots, k\) \(\{\text{over the links } k + 1, k + 2, \ldots, n\}\) during the \(i\)th phase of the computation under \(P\) on the input \(x'' = (x_1, \ldots, x_k, x'_{k+1}, \ldots, x'_n)\) is \(u_i\) \(\{\text{is } v_i\}\) for \(i = 1, 2, \ldots, l\).
(b) If \(s \in Q - S\) then there is \(v_{i+1}\) such that the sequence \(s, (u_{i+1}, v_{i+1})\) is 1-easy, where \(u_{i+1}\) is the communication vector during the \((l + 1)\)st phase of the computation under \(P_1\) on \(x\).
(c) If the empty sequence is 1-easy and it belongs to $Q - S$ then there is $v$ such that the couple $(u, v)$ is 1-easy, where $u$ is the communication vector during the first phase of the computation under $P_1$ on $x$.

**Proof.** (a) By a contradiction. Assume to the contrary that (a) does not hold for some index $i$, $1 \leq i \leq l$. Let $\hat{i}$ be such minimal index. We first show that $u_{\hat{i}}$ is the communication vector over the links $1, 2, \ldots, k$ during the $\hat{i}$th phase of the computation under $P$ on $x''$. By minimality of $\hat{i}$, the coordinator sends the same messages via the links $1, 2, \ldots$, since these messages are fully determined in both cases only by $P$ and by the sequence $(u_1, v_1), \ldots, (u_{\hat{i} - 1}, v_{\hat{i} - 1})$, (see behaviour of $P_1$ during a phase described above). Moreover, $p_j$ owning $x_j$ $(1 \leq j \leq k)$ responds the same message during the $\hat{i}$th phase of the computation under $P_1$ on $x$ and under $P$ on $x''$, since behaviour of $p_j$ under $P_1$ and under $P$ is the same for $1 \leq j \leq k$ (see above), and, by minimality of $\hat{i}$, the communication between the coordinator and $p_j$ owning $x_j$ is the same under $P_1$ on $x$ and under $P$ on $x''$ after completing the $(\hat{i} - 1)$st phase for $1 \leq j \leq k$; (these communications are described in both cases by the sequence $u_1, \ldots, u_{\hat{i} - 1}$). But it means that the communication vector over the links $1, 2, \ldots, k$ is the same (i.e., $u_{\hat{i}}$) during the $\hat{i}$th phase of the computation under $P_1$ on $x$ and under $P$ on $x''$.

Now let us prove that $v_{\hat{i}}$ is the communication vector over the links $k + 1, k + 2, \ldots, n$ during the $\hat{i}$th phase of the computation under $P$ on $x''$. Let $c$ denote the communication vector described by the sequence $(u_1, v_1), \ldots, (u_{\hat{i} - 1}, v_{\hat{i} - 1})$. Since $s$ is a prefix of $s'$ and because of minimality of $\hat{i}$, the communication vector after completing the $(\hat{i} - 1)$st phase of the computation under $P$ on $x'$ and under $P$ on $x''$ is $c$. Therefore, the coordinator sends the same messages via the links $k + 1, k + 2, \ldots, n$ during the $\hat{i}$th phase of the computation under $P$ on $x''$ and under $P$ on $x'$, since these messages are fully determined in both cases only by $P$ and $c$. Moreover, $p_j$ owning $x_j$ $(k + 1 \leq j \leq n)$ responds the same message during the $\hat{i}$th phase of the computation under $P$ on $x'$ and under $P$ on $x''$, since the communication between the coordinator and $p_j$ owning $x_j$ after completing the $(\hat{i} - 1)$st phase of the computation under $P$ on $x'$ and under $P$ on $x''$ is the same for $k + 1 \leq j \leq n$, by minimality of $\hat{i}$; (these communications are described in both cases by the sequence $v_1, \ldots, v_{\hat{i} - 1}$). But it means that the communication vector over the links $k + 1, k + 2, \ldots, n$ is the same (i.e., $v_{\hat{i}}$) during the $\hat{i}$th phase of the computation under $P$ on $x'$ and under $P$ on $x''$.

Hence the results above contradict our assumption that (a) does not hold for $\hat{i}$.

(b) Assume that $s \in Q - S$. Let $s''$ be the sequence in $S$ corresponding to the input $x''$. By (a), $s$ is a prefix of $s''$. There is the $(l + 1)$st phase of the computation under $P$ on $x''$, since $s'' \in S$, $s$ is a prefix of $s''$ and $s \in Q - S$. Let $u_{l+1}$ [let $w_{l+1}$] be the communication vector over the links $1, 2, \ldots, k$ [over the links $k + 1, k + 2, \ldots, n$] during the $(l + 1)$st phase of the computation under $P$ on $x''$. Hence, the sequence $s, (u_{l+1}, w_{l+1})$ belongs to $Q$. Thus, there is $v_{l+1}$ such that the sequence $s, (u_{l+1}, v_{l+1})$ is 1-easy, since $s$ is 1-easy. Finally, one can show (by a similar manner as in the first part of the proof of (a) above—formally, replace $\hat{i}$ by $l + 1$) that the communication vector over the links $1, 2, \ldots, k$ is the same (i.e., $u_{l+1}$) during the $(l + 1)$st phase of the computation under $P_1$ on $x$ and under $P$ on $x''$.

(c) The proof is very similar to the proof of (b) and hence we omit details here; (formally, replace $s$ in the proof of (b) by the empty sequence and realize that the empty sequence is a prefix of $s''$).

This completes the proof of Lemma 4. □
Corollary 2. Let \( x \) be any input for \( P_1 \). Then there is an 1-easy sequence \( s = (u_1, v_1), \ldots, (u_t, v_t) \) in \( S \) with \( t \geq 1 \) such that \( u_i \) is performed and \( v_i \) is selected during the \( i \)th phase of the computation on \( x \) under \( P_1 \) for \( i = 1, 2, \ldots, t \).

Proof. By the assumption of Case 1, the empty sequence is 1-easy. Hence, by the definition of the 1-easy sequences, the empty sequence cannot belong into \( S \), i.e., it belongs into \( Q - S \). Therefore, one can construct the desired 1-sequence \( s \) in \( S \) first by applying (c) of Lemma 4 and then (if necessary) by applying (b) of Lemma 4 repeatedly until the constructed 1-easy sequence \( (u_i, v_i), \ldots, (u_{i+1}, v_{i+1}) \) belongs to \( S \). \( \blacksquare \)

Now we complete the proof of Case 1 as follows. We derive the desired contradiction by showing that \( P_1 \) computes \( f_1 \) correctly using less than \( DC(f_1) \) exchanged bits. Let \( x = (x_1, \ldots, x_k) \) and \( x' = (x'_1, \ldots, x'_k) \) be any two inputs for \( P_1 \) with the same communication vector under \( P_1 \), say \( c = (c_1, \ldots, c_k) \). To guarantee correctness of \( P_1 \) it is enough to show that \( f_1(x) = f_1(x') \), since the coordinator knowing only \( c \) either accepts both inputs \( x \) and \( x' \) or rejects them. Now let us prove \( f_1(x) = f_1(x') \).

Hence \( c \) is described by \( u_1, \ldots, u_t \). Note that \( c \) is uniquely determined by \( u_1, \ldots, u_t, P_1 \) is a deterministic protocol, and \( v_i \) is uniquely determined by \( (u_1, v_1), \ldots, (u_{t-1}, v_{t-1}), u_t, \) and \( P_1 \) (see behaviour of \( P_1 \) during a phase described above). Let \( s' \) be the corresponding 1-easy sequence in \( S \) for the input \( x' \) according to Corollary 2. Hence \( s' = s \), because of the same communication vector \( c \) for the inputs \( x \) and \( x' \); recall that \( P_1 \) and \( c \) uniquely determine \( s \) (and in our case also \( s' \)). Let \( (x''_1, \ldots, x''_n) \) be any input for \( P \) with the communication vector described by \( s \). By (a) of Lemma 4, the communication vector under \( P \) on the input \( \tilde{x} = (x_1, \ldots, x_k, x''_{k+1}, \ldots, x''_n) \) is described by \( s \). Since \( s' = s \) and again by (a) of Lemma 4, the same holds also for the input \( \tilde{x} = (x'_1, \ldots, x'_k, x''_{k+1}, \ldots, x''_n) \). Therefore, \( f(\tilde{x}) = f(\tilde{x}) \), since \( P \) computes \( f \) and the coordinator knowing only the communication vector described by \( s \) either accepts both inputs \( \tilde{x} \) and \( \tilde{x} \) or rejects them. It means that \( f_1(x) = f_1(x) \), since \( f_1(x_1, \ldots, x_k) \neq f_2(x''_{k+1}, \ldots, x''_n) = f(\tilde{x}) = f(\tilde{x}) = f(\tilde{x}) \). Hence \( P_1 \) computes \( f_1 \) correctly.

Finally, we derive the desired contradiction mentioned above as follows. Choose arbitrary input \( x \) for \( P_1 \). By Corollary 2, for \( x \) there is an 1-easy sequence \( s = (u_1, v_1), \ldots, (u_t, v_t) \) in \( S \) such that \( u_i \) is performed and \( v_i \) is selected during the \( i \)th phase of the computation under \( P_1 \) on \( x \) for \( i = 1, 2, \ldots, t \). But it means that the arbitrarily chosen input \( x \) is recognized by using \( \sum_{i=1}^t h(u_i) < DC(f_1) \) exchanged bits, a contradiction.

Case 2. The empty sequence is 2-easy. In such a case one can derive a contradiction (by a similar way as in Case 1) by showing that there is a deterministic protocol \( P_2 \) computing \( f_2 \) using less than \( DC(f_2) \) exchanged bits. Note that the main strategy of \( P_2 \) on an input is to perform a communication described by a sequence \( v_1, v_2, \ldots, v_t \) (one \( v_i \) per a phase) and to select a sequence \( u_1, u_2, \ldots, u_t \) (one \( u_i \) per a phase) so that the sequence \( (u_1, v_1), \ldots, (u_t, v_t) \) may be 2-easy for \( i = 1, 2, \ldots, t \) and the sequence \( (u_1, v_1), \ldots, (u_t, v_t) \) may belong to \( S \). Also one has to modify Lemma 4 and Corollary 2 to be usable for Case 2 by interchanging the role of the links 1, 2, \ldots, \( k \) and the links \( k + 1, k + 2, \ldots, n \), by interchanging the role of \( u_i \)'s and \( v_i \)'s, and by interchanging the role of 1-easy and 2-easy sequences. We omit the details here, since the proof is very similar to the proof of Case 1.
Case 3. The empty sequence is 1-hard and 2-hard, too. One can observe that if there is a sequence 
$$(u_1, v_1), \ldots, (u_{i-1}, v_{i-1})$$ in $Q - S$ that is 1-hard and also 2-hard, then there is a couple $(u_i, v_i)$ such 
that the sequence $$(u_1, v_1), \ldots, (u_i, v_i)$$ is 1-hard and also 2-hard. Consequently, there is a sequence 
$s = (u_1, v_1), \ldots, (u_i, v_i)$ in $S$ that is 1-hard and also 2-hard, since it is assumed that the empty sequence 
is 1-hard and also 2-hard. It means that $P$ recognizes each input with the communication vector de-
scribed by $s$ using $\sum_{i=1}^{t} h(u_i) + \sum_{i=1}^{t} h(v_i) \geq DC(f_1) + DC(f_2)$ exchanged bits. But this contradicts 
our assumption above, that $P$ computes $f$ using less than $DC(f_1) + DC(f_2)$ exchanged bits.

This completes the proof of Theorem 2. \(\Box\)

References

94–99.
433.