Multiparty Communication Complexity: Very Hard Functions ⋆

Ján Maňuch

Department of Mathematics and Turku Centre for Computer Science,
University of Turku, FIN-20014 Turku, Finland
manuch@cs.utu.fi

Abstract. We consider the multiparty communication model defined in [DF89] using the formalism from [Hr97]. First, we correct an inaccuracy in the proof of the fundamental result of [DR95] providing a lower bound on the nondeterministic communication complexity of a function. Then we construct several very hard functions, i.e., functions such that those as well as their complements have the worst possible nondeterministic communication complexity. The problem to find a particular very hard function was proposed in [Du99], where it has been shown that almost all functions are very hard. We also prove that combining two very hard functions by the boolean operation xor gives a very hard function.

Introduction

The multiparty model is a natural extension of the two-party model that has been extensively studied. The two-party communication model assumes that each of two parties has a part of the input, and the aim is to compute a given boolean function on the input with the minimal amount of communication, i.e., minimizing the communication complexity. The study of the communication complexity of the two-party model was inspired by the VLSI circuits complexity. There are many other applications of the communication complexity. An overview of applications can be found in [KN97], and an overview of results on the two-party model in [Hr97].

In the multiparty model the input is distributed among n processors and the goal is the same: to compute a function on the whole input with the minimal 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 with each other. The multiparty model was introduced and investigated in [DF89].

Note that in [CFL83] a different multiparty model was considered. In that model each of the n parties has all the inputs except one, and all parties communicate through a shared "blackboard". This model was investigated in [BNS89], where an interesting relation to time-space tradeoffs and branching programs was discovered. We do not know any connections between these two models.

⋆ Supported by Academy of Finland under Grant No. 44087.

There are only few results known about the multiparty model introduced in [DF89]. In [DF92] an upper bound on the deterministic communication complexity of order $O(\mathrm{k}(f)\,\mathrm{k}^2(1-f)\,\mathrm{ncc}(f)\,\mathrm{ncc}(1-f))$ was established, where $\mathrm{k}(f)$ is the number of processors accessed and $\mathrm{ncc}(f)$ is the nondeterministic communication complexity of the function f. In [DR95] a fundamental tool to prove a lower bound on the communication complexity of a given boolean function was developed. However, there was a small mistake in the proof. In this paper we give a correct proof of this result. Further results proved in [DR95] are the following relations between the nondeterministic communication complexities of a function and its complement and between the deterministic and nondeterministic communication complexities:

$$\mathrm{ncc}(1-f) \leq n\ \left(\ 1+2^{\mathrm{ncc}(f)}\ \right), \quad \mathrm{cc}(f) \leq n\ \left(\ 1+2^{\mathrm{ncc}(f)}\ \right)\ .$$

For the restricted model, where only one communication is allowed for each direction on each link, the following results were established in [DR95]:

$$\mathrm{ncc}_1(f) = \mathrm{ncc}(f)\,, \quad \mathrm{cc}_1(f) \leq \mathrm{cc}(f) \cdot 2^{\mathrm{cc}(f)+1}\,.$$

For all these bounds it is shown that they are optimal.

The main result of this paper is the construction of several very hard functions. A very hard function is a function such that it and its complement have the worst nondeterministic communication complexity. In the deterministic case, of course, the communication complexities of a function f and its complement $1-f$ are equal. In the nondeterministic case there could be even an exponential difference between the communication complexities of f and $1-f$, cf. [DR95]. However, in some sense, functions f and $1-f$ represent the same function, so it is better to switch to the complementary function, if that dramatically lowers the nondeterministic communication complexity. The existence of a very hard function shows that we cannot always use this approach to decrease the nondeterministic communication complexity. The problem to find a particular very hard function was proposed in [Du99], where it has been shown that almost all functions are very hard.

Moreover we prove that combining two very hard functions by the boolean operation *xor* gives also a very hard function. This result extends a similar result from [DR95] that claims that the boolean operation *and* preserves the hard functions and the result from [Du99] that claims that the deterministic communication complexity adds up when combining two functions by xor.

The full version of this paper appears in [Ma99].

1 Preliminaries

In this section we define the multiparty model, the nondeterministic protocols and the communication complexity. We start with an informal definition of the model.

The multiparty model consists of a coordinator and n parties. The coordinator wishes to evaluate a boolean function $f(x_1, \ldots, x_n)$. The input vector

$x = (x_1, \ldots, x_n)$ is distributed among parties, with $x_i \in \{0,1\}^m$ known to the party i. We allow a communication only between the coordinator and any party. Instead of saying "the communication between the coordinator and party i" we often say "the communication on link i". The computation consists of several phases, where one phase is as follows:
The coordinator sends some nonempty messages to some parties and then, each party that got a message, sends a nonempty message back to the coordinator. After the computation the coordinator announces the result: $\bar{1}$ if accepted or $\bar{0}$ if rejected. In the nondeterministic case we accept an input, if there exists an accepting computation for this input.

Next, we give formal definitions of the nondeterministic protocol. We will use the notation from [Hr97]. Let λ be the empty string.

Definition 1. *A* ***nondeterministic protocol*** *over* $X = \{x_1, \ldots, x_n\}$ *with* $x_i = x_i^1 \ldots x_i^m \in \{0,1\}^m$, $n \geq 1$, $m \geq 2$, *is a pair* $P = \langle \Phi_C, \Phi_P \rangle$, *where*

(a) Φ_C *is a communication relation of the coordinator in*

$$[\{0,1,\$\}^*]^n \times ([\{0,1\}^*]^n \cup \{\bar{0},\bar{1}\}) ;$$

and the projections of Φ_C *are defined as follows: let* $c \in [\{0,1,\$\}^*]^n$, *then*

$$\Phi_C(c) \cap [\{0,1\}^*]^n = (\Phi_{C,1}(c), \ldots, \Phi_{C,n}(c)) ;$$

(b) Φ_P *is a communication relation of parties in*

$$\left(\{1,\ldots,n\} \times \{0,1\}^m \times \{0,1,\$\}^+\right) \times \{0,1\}^+,$$

where

(i) Φ_C *has the* ***prefix freeness property:***
for each $i \in \{1,\ldots,n\}$, $c, c' \in [\{0,1,\$\}^*]^n$, *if* $c_i = c'_i$, *where* c_i *is the* i*-th component of* c *and* $d, d' \in \{0,1\}^+$ *such that* $(c,d), (c',d') \in \Phi_{C,i}$, *then* d *is not a proper prefix of* d'; *if* c_i *is a proper prefix of* c'_i *and* $d \in \Phi_{C,i}(c) \cap \{0,1\}^+$, *then* $c_i d$ *is not a proper prefix of* c'_i *(each party knows when it is the end of a message from the coordinator),*

(ii) Φ_P *has the* ***prefix freeness property:***
for each $i \in \{1,\ldots,n\}$, $c \in \{0,1,\$\}^*$, $d, d' \in \{0,1\}^+$ *and* $x_i, x'_i \in \{0,1\}^m$ *such that* $((i,x_i,c),d), ((i,x'_i,c),d') \in \Phi_P$ d *is not a proper prefix of* d' *(the coordinator knows when it is the end of a message from any party),*

(iii) for each $i \in \{1,\ldots,n\}$, $c, c' \in [\{0,1,\$\}^*]^n$ *such that* $c_i = c'_i \neq \lambda$, *if* $\Phi_C(c) \cap \{\bar{0},\bar{1}\} \neq \emptyset$, *then* $\Phi_C(c') \subseteq \{\bar{0},\bar{1}\}$ *or* $\Phi_{C,i}(c') = \{\lambda\}$ *(if there is a communication on link* i *then party* i *knows when it is finished).*

Let us give an intuitive explanation of the symbols used in the above definition. $0, 1$ represent bits sent through the communication links and \$ is a virtual end mark of messages. *Virtual* means that the symbol is not send neither by the coordinator, nor a party. The properties *(i)–(iii)* ensure that a virtual end mark is not necessary. In fact, the use of such a mark during the communication can lead to very nonintuitive properties of the model: Consider, for example, a node which sends k empty messages and in this way the receiver obtains an arbitrary large information, the number k, with a minimal increase of the communication complexity. Symbols $\bar{0}, \bar{1}$ represent the result value of a computation.

The relation Φ_C maps a temporary state of a communication on all links to the one of the result values or an n-ary sequences of messages sent from the coordinator to the each party. The empty message means that the coordinator does not communicate with the party. The relation Φ_P maps the number i of a party, the input of the party i and a temporary state of the communication on the link i to a nonempty message sent from the party i back to the coordinator. Since we consider nondeterministic protocols, these maps could be ambiguous.

We proceed with the definition of the computation.

Definition 2. *A* ***computation*** *of P on an input vector $x = (x_1, \ldots, x_n)$ is a communication vector $c = (c_1, \ldots, c_n)$ with $c_i = c_i^1\$c_i^2\$\ldots\$c_i^{2r_i-1}\$c_i^{2r_i}\$$, where*

(i) for all $i \in \{1, \ldots, n\}$ $c_i^1, \ldots, c_i^{2r_i} \in \{0,1\}^+$;
(ii) we can find an r and a sequence of vectors $c_{[0]}, \ldots, c_{[2r]} \in [\{0,1,\$\}^*]^n$, i.e., states of the computation such that

(a) $c_{[0]} = (\lambda, \ldots, \lambda)$,
(b) if l is even, then

$$c_{[l+1]} \in \{(c_{[l]1}d_1\$, \ldots, c_{[l]n}d_n\$) \mid (d_1, \ldots, d_n) \in \Phi_C(c_{[l]})\},$$

(c) if l is odd, then for each $i \in \{1, \ldots, n\}$

$$\begin{aligned} c_{[l+1]i} &= \lambda, && \text{if } c_{[l]i} = \$, \\ c_{[l+1]i} &= c'\$, && \text{if } c_{[l]i} = c'\$\$, \\ c_{[l+1]i} &\in \{c_{[l]i}d\$ \mid d \in \Phi_P(i, x_i, c_{[l]i})\}, && \text{otherwise,} \end{aligned}$$

(d) $c_{[2r]} = c$;

(iii) $\Phi_C(c) \subseteq \{\bar{0}, \bar{1}\}$.

We denote the set of all computations on an input x (resp. all computations) by comp(P, x) *(resp.* comp(P)*). We say that a computation c is* ***accepting****, if $\bar{1} \in \Phi_C(c)$. The smallest r is called the* ***number of rounds*** *of c. P is called an* ***r-round nondeterministic protocol*** *if every computation of P has at most r rounds.*

We say that P ***computes*** *1 for an input vector x, i.e., $P(x) = 1$, if there exists an accepting computation c of P on x, otherwise P computes 0. We say that P* ***computes*** *the boolean function f with the input variables X, if for each $x \in [\{0,1\}^m]^n$ we have $f(x) = P(x)$.*

Now, we illustrate the above definitions with the following example.

Example 1. Consider the function f defined by

$$f(x_1, \ldots, x_n) = 1 \text{ iff there exist } i \neq j \text{ such that } x_i = x_j .$$

We construct a nondeterministic protocol $P = \langle \Phi_C, \Phi_P \rangle$ computing this function:

$$\Phi_C(\lambda, \ldots, \lambda) = \{(b_1, \ldots, b_n) \mid b_i = b_j \in \{0,1\} \text{ and } \forall k \neq i, j : \; b_k = \lambda\},$$

$$\Phi_P(i, x_i, b\$) = \begin{cases} \{x_i^2 \ldots x_i^m\}, & \text{if } x_i^1 = b, \\ \emptyset, & \text{otherwise,} \end{cases}$$

$$\Phi_C(c_1, \ldots, c_n) = \begin{cases} \bar{1}, & \text{if } c_i = c_j = b\$y\$,\ b \in \{0,1\},\ y \in \{0,1\}^{m-1} \\ & \text{and } \forall k \neq i, j\text{: } c_k = \lambda, \\ \bar{0}, & \text{otherwise.} \end{cases}$$

Φ_C has the prefix freeness property, since all messages the coordinator sends, have the same length. The same holds for Φ_P. If we have $c, c' \in [\{0,1,\$\}^*]^n$ with $c_i = c'_i \neq \lambda$ and $\Phi_C(c) \cap \{\bar{0}, \bar{1}\} \neq \emptyset$, then $\Phi(c') \subseteq \{\bar{0}, \bar{1}\}$, so P satisfies also condition *(iii)* of Definition 1. Hence it is a nondeterministic protocol.

Informally we can describe the protocol P as follows: The coordinator guesses for which $i \neq j$ the equality $x_i = x_j$ holds. It also guesses the first bit of $x_i = x_j$ and sends it to the parties i and j. It does not communicate with the other parties at all. The party i (resp. j) checks if the coordinator guessed the first bit of x_i (resp. x_j) correctly and in such case sends back the rest of the input x_i (resp. x_j). Finally, the coordinator checks if the rest of x_i equals to the rest of x_j. Only in such case it ends with $\bar{1}$. Clearly, this protocol computes the function f.

Consider an input vector $x = (x_1, \ldots, x_n)$ such that $x_1 = x_2 = a_1 \ldots a_m \neq x_3$ and $a_1 = x_3^1$. Then $c = (a_1\$a_2 \ldots a_m\$, a_1\$a_2 \ldots a_m\$, \lambda, \ldots, \lambda)$ is an accepting computation of P on x. Indeed, we have $r_1 = r_2 = 1$, $r_3 = \cdots = r_n = 0$ and the sequence of states of the computation is:

$$c_{[0]} = (\lambda, \ldots, \lambda), \qquad c_{[1]} = (a_1\$, a_1\$, \$, \ldots, \$),$$
$$c_{[2]} = (a_1\$a_2 \ldots a_m\$, a_1\$a_2 \ldots a_m\$, \lambda, \ldots, \lambda) = c .$$

The number of rounds of c is 1. The computation $c' = (a_1\$a_2 \ldots a_m\$, \lambda, a_1\$x_3^2 \ldots x_3^m\$, \lambda, \ldots, \lambda)$ is not an accepting computation. Obviously, each computation uses exactly 1 round, so P is a 1-round protocol.

We finish this section by the definition of the nondeterministic communication complexity.

Definition 3. *The* ***length of a computation*** *c of P is the total length of all messages $m \in \{0,1\}^+$ in c. For each $x \in [\{0,1\}^m]^n$ such that $P(x) = 1$, let $\mathrm{ncc}(P, x)$ denote the length of the shortest accepting computation of P on x. The* ***nondeterministic communication complexities*** *of the protocol P and of the function f are*

$$\mathrm{ncc}(P) = \max\{\mathrm{ncc}(P, x) \mid P(x) = 1\}, \ \mathrm{ncc}(f) = \min\{\mathrm{ncc}(P) \mid P \textit{ computes } f\} .$$

Similarly, we can define $\mathrm{ncc}(P, x/J)$, $\mathrm{ncc}(P/J)$ *and* $\mathrm{ncc}(f/J)$, *when we measure only the communication on links in* J.

Example 1 (continued). The length of both computations c and c' is $2m$. Clearly this holds for any accepting computation of P. Hence $\mathrm{ncc}(P) = 2m$, , which implies $\mathrm{ncc}(f) \leq 2m$.

Using the idea how the protocol P works, we can construct for any boolean function f a protocol with the nondeterministic communication complexity nm, which, first, sends all the information the parties have to the coordinator and then the coordinator is able to compute f. This implies:

Proposition 1. *For any boolean function* $f : [\{0,1\}^m]^n \to \{0,1\}$, $\mathrm{ncc}(f) \leq nm$.

Now we can define *hard* and *very hard* functions.

Definition 4. *A boolean function is* ***hard****, if* $\mathrm{ncc}(f) = nm$ *and it is* ***very hard****, if* $\mathrm{ncc}(f) = \mathrm{ncc}(1-f) = nm$.

2 The Fundamental Tool

In this section we correct Theorem 1 and its proof from [DR95]. The mistake in [DR95] occurs actually in Lemma 1. Here is the correct version of Lemma 1.

Lemma 1. *Let* $b_1, b_2, \ldots, b_p$ *be a sequence of nonempty binary strings such that no element is a proper prefix of another and no element of this sequence occurs in it more than* q *times. Then* $\sum_{i=1}^{p} |b_i| \geq p \log \frac{p}{q}$.

Before the proof let us show that Lemma 1 in [DR95] does not hold.

Example 2. In Lemma 1 and its proof of [DR95] the inequalities $\sum_{i=1}^{p} |b_i| \geq p \log \left\lceil \frac{p}{q} \right\rceil$ and $\sum_{i=1}^{p} |b_i| \geq p \left\lceil \log \frac{p}{q} \right\rceil$ are claimed, and the later one is used in the proof of Theorem 1 in [DR95]. However, the following sequence of binary strings $00, 00, 00, 01, 1, 1, 1$ shows that neither of these is true. Indeed, we have $\sum_{i=1}^{p} |b_i| = 11$, $p = 7$ and if we choose $q = 3$, then $p \log \left\lceil \frac{p}{q} \right\rceil \doteq 11{,}09$ and $p \left\lceil \log \frac{p}{q} \right\rceil = 14$.

Proof (Proof of Lemma 1). Let $q' \leq q$ be the maximal number of occurrences of a string in the sequence. For $i = 1, \ldots, q'$ consider sets Q_i containing strings which occur in the sequence at least i times. Clearly, $\sum_{i=1}^{q'} |Q_i| = p$. For each set Q_i there is a corresponding binary tree such that the strings in Q_i encodes the paths of the tree. Since no element in the sequence is a proper prefix of another element, all strings end in the leaves of the tree. The trees $Q_1, \ldots, Q_{q'}$ form together the forest G. Let $T(G)$ denote the total sum of the depths of all leaves of the forest G. Obviously, $T(G) = \sum_{i=1}^{p} |b_i|$. Now we transform the forest G into G' in two steps:

Step (i). If some node in G has only one son, then we delete the node and replace it by its son. In this way we get a binary tree.
Step (ii). Let v_1 (resp. v_2) be a leaf with the maximal (resp. minimal) depth. If $\text{depth}(v_1) \leq \text{depth}(v_2) + 1$, then the forest is balanced. Otherwise, we perform the following operation: if v is the father of v_1, then we cut off both sons of v and connect them to v_2. In this way we get the balanced tree G' with depths of leaves either h or $h+1$.

Note that in both steps we preserve the numbers of both leaves and trees, and the depths of leaves can only decrease, thus $T(G') \leq T(G)$. Let a (resp. b) be the number of leaves in depth h (resp. $h+1$). Clearly, $a + b = p$. It is easy to derive $q' \cdot 2^{h+1} = 2a + b = p + a$, which implies

$$T(G) \geq T(G') = ha + (h+1)b = p \log \frac{p+a}{q'} - a \geq p \log \frac{p}{q'},$$

where the last inequality comes from the inequality $\log(x+1) \geq x$ for $x \in \langle 0, 1 \rangle$, in which we substitute $x = \frac{a}{p}$. □

To state the theorem we need another two definitions.

Definition 5. [DR95] *Let Y be a nonempty subset of $f^{-1}(1)$. We say that the index j is **important for f with respect to** Y, if for every $y = (y_1, \ldots, y_n) \in Y$ there is $y'_j \in \{0,1\}^m$ such that $f(y_1, \ldots, y_{j-1}, y'_j, y_{j+1}, \ldots, y_n) \neq f(y)$.*

Let $J \subseteq \{1, \ldots, n\}$ be a nonempty set of indices and $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$ two input vectors. We denote

$$[x:y]/J = (z_1, \ldots, z_n), \text{ where } z_i = \begin{cases} x_i, & \text{if } i \in J, \\ y_i, & \text{otherwise.} \end{cases}$$

*Let M be a nonempty sets of inputs. Then J**-closure of** M is the set*

$$C\ell_J(M) = \{\{[x:y]/J \mid x, y \in M\}.$$

If we use Lemma 1 instead of incorrect Lemma 1 in [DR95] in the proof of the theorem, we get the following result.

Theorem 1. *Let Y be a nonempty subset of $f^{-1}(1)$ and $J_1, \ldots, J_r$ be pairwise disjoint sets of indices. Assume that every $j \in \cup_{i=1}^{r} J_i$ is a important index for f with respect to Y and let d_i be integers satisfying*

$$d_i \geq \max_{M \subseteq Y} \{|M| \mid C\ell_{J_i}(M) \subseteq f^{-1}(1)\}.$$

Then we have lower bounds on the nondeterministic communication complexity

$$\text{ncc}(f/J_i) \geq \log \frac{|Y|}{d_i}, \qquad \text{ncc}(f) \geq \sum_{i=1}^{r} \log \frac{|Y|}{d_i}.$$

The proof is based on Lemma 1, properties of nondeterministic protocols and the classical crossing sequence argument and we refer interested readers to [DR95]. Note that the above theorem differs from Theorem 1 in [DR95], where it was claimed $\mathrm{ncc}(f) \geq \sum_{i=1}^{r} \left\lceil \log \frac{|Y|}{d_i} \right\rceil$. We apply this theorem in Example 1.

Notation. *For every $a \in \{0,1\}^m$ and every integer d let $\mathrm{val}(a)$ denote the value of the binary string a and let $(a)_d$ denote the string from $\{0,1\}^m$ such that $\mathrm{val}((a)_d) \equiv \mathrm{val}(a) + d \bmod 2^m$.*

Example 1 (continued). Assume that $n \leq 2^{m-1}+1$. Let us choose in Theorem 1: $r = 2$, $J_1 = \{1\}$, $J_2 = \{2\}$ and $Y = \{(a, a, (a)_1, (a)_2, \ldots, (a)_{n-2}) \mid a \in \{0,1\}^m\}$. It is easy to see that indices 1 and 2 are important with respect to Y. Take arbitrary disjoint $a, b \in \{0,1\}^m$. Assume that $f(a, b, (b)_1, (b)_2, \ldots) = f(b, a, (a)_1, (a)_2, \ldots) = 1$. Then we have $a = (b)_c$ and $b = (a)_d$ for some integers $1 \leq c, d \leq n-2 \leq 2^{m-1}-1$. Obviously $c + d = 2^m$, which contradicts the above restrictions for c and d. Hence we can set $d_1 = 1$ and similarly $d_2 = 1$. Using Theorem 1 we obtain $\mathrm{ncc}(f) \geq 2\log(2^m) = 2m$. Thus we can conclude $\mathrm{ncc}(f) = 2m$ for all $n \leq 2^{m-1}+1$.

3 Very Hard Functions

In this section we give the main result of the paper. It can be shown using Theorem 1 that the function $g(x_1, x_2) = 1$ iff $\mathrm{val}(x_1) \leq \mathrm{val}(x_2)$ is a very hard function. We construct several very hard functions for arbitrary n using the function g. We need some more definitions.

Definition 6. *Let $f(x_1, \ldots, x_n)$ be a boolean function, $x_i \in \{0,1\}^m$ and let $J_i = \{i\}$ for all $i \in \{1, \ldots, n\}$. If there exists a nonempty set $Y \subseteq f^{-1}(1)$ such that for all $i \in \{1, \ldots, n\}$ the index i is important with respect to Y and there exist integers d_i such that for every $i \in \{1, \ldots, n\}$*

$$d_i \geq \max_{M \subseteq Y} \{|M| \mid C\ell_{J_i}(M) \subseteq f^{-1}(1)\}, \text{ and} \tag{1}$$

$$\log \frac{|Y|}{d_i} = m\,, \tag{2}$$

*then the function f is **strongly hard**. We will say that the function f is **strongly very hard** when both the function f and its complement $1-f$ are strongly hard.*

According to Theorem 1 it is obvious that any strongly (very) hard function is also a (very) hard function. In what follows we take a strongly very hard function $f(x_1, x_2)$ and using the operation *xor* on boolean strings we construct a function $g(x_1, \ldots, x_n)$. We show that the function g is very hard. But let us first define the operation xor on boolean strings and the *xor constructor* $F^{\oplus}$ which constructs the function g.

Definition 7. *Let* $\oplus$ *denote the boolean operation* ***xor****. Let* $x = x^1 \dots x^m$*,* $y = y^1 \dots y^m$ *be* m*-bits long boolean strings. We define the xor operation on strings as follows:* $x \oplus y = x^1 \oplus y^1 \dots x^m \oplus y^m$*. Let* $n \geq 2$ *be an integer,* $J \subsetneq \{1, \dots, n\}$ *be a nonempty set and* $f(x_1, x_2)$ *be a boolean function. We define the* ***xor constructor*** $F^{\oplus}$ *as* $F^{\oplus}(J, n, f)\,(x_1, \dots, x_n) = f\Big(\bigoplus_{i \in J} x_i, \bigoplus_{i \in \{1,\dots,n\}-J} x_i\Big)$*.*

The proof of the following theorem can be found in [Ma99].

Theorem 2. *Let* $f(x_1, x_2)$ *be a strongly very hard function,* n *be an integer and* $J \subsetneq \{1, \dots, n\}$ *be a nonempty set. Then the function* $g = F^{\oplus}(J, n, f)$ *is very hard.*

Theorem 2 gives us the method how to construct a very hard function for any n. But first we have to find a strongly very hard function $f(x_1, x_2)$. We improve the function g to be strongly very hard. Consider the function g' defined by

$$g'(x_1, x_2) = 1 \text{ iff } \operatorname{val}(x_1) \leq \operatorname{val}(x_2) \wedge (x_1, x_2) \neq (0^m, 1^m) \vee (x_1, x_2) = (1^m, 0^m)\,.$$

Theorem 3. *The function* g' *is strongly very hard.*

Proof. First, consider the function g'. Set $J_1 = \{1\}$, $J_2 = \{2\}$, $Y = \{(x_1, x_2) \mid x_1 = x_2\}$. Take any vector $x = (a, a) \in Y$. We have $g'((a)_{+1}, a) = g'(a, (a)_{-1}) = 0$, hence both indices are important with respect to Y. Consider now $a \neq b$. Clearly, exactly one of the values $g'(a, b)$ and $g'(b, a)$ is equal to 0. Hence the integers $d_1 = d_2 = 1$ satisfy the condition (1) of Definition 6 and since $|Y| = 2^m$, they satisfy also (2). So the function g' is strongly very hard.

For the function $1 - g'$ we set $J_1 = \{1\}$, $J_2 = \{2\}$, $Y = \{((a)_{+1}, a) \mid a \in \{0,1\}^m\}$. Since, for all $a \in \{0,1\}^m$, $(1 - g')(a, a) = 0$, both indices $1, 2$ are important with respect to Y. Take any vectors $((a)_{+1}, a), ((b)_{+1}, b) \in Y$ where $a \neq b$. We want to show that at least one of the values $(1 - g')((a)_{+1}, b)$, $(1 - g')((b)_{+1}, a)$ is equal to 0. Assume the contrary that both the values are equal to 1. Then we have $(a)_{+1} > b$ and either $((a)_{+1}, b) \neq (1^m, 0^m)$ or $((a)_{+1}, b) = (0^m, 1^m)$. In the second case both a, b are equal to 1^m, which is a contradiction. In the first case we obtain $2^m - 1 > \operatorname{val}(a) + 1 > \operatorname{val}(b) > 0$ and similarly $2^m - 1 > \operatorname{val}(b) + 1 > \operatorname{val}(a) > 0$. These together imply the inequality $\operatorname{val}(a) + 1 > \operatorname{val}(b) > \operatorname{val}(a) - 1$. Thus again $a = b$, a contradiction. We have shown that at least one of values $(1 - g')((a)_{+1}, b), (1 - g')((b)_{+1}, a)$ is equal to 0. The cardinality of the set Y is 2^m. Therefore as in the previous case the integers $d_1 = d_2 = 1$ satisfy both conditions (1) and (2). Hence the function $1 - g'$ is strongly hard and the function g' is strongly very hard. $\square$

As an immediate consequence of Theorems 2 and 3 we have

Corollary 1. *Let* $n \geq 2$ *be an integer and* J *be a nonempty subset of* $\{1, \dots, n\}$*. The function* $F^{\oplus}(J, n, g')$ *is very hard.*

Next, we analyze the functions created by combining two boolean functions by the boolean operation xor. More precisely:

Definition 8. *Let* $g_1 : [\{0,1\}^m]^{n_1} \to \{0,1\}$, $g_2 : [\{0,1\}^m]^{n_2} \to \{0,1\}$ *be two boolean functions. Let* $n = n_1 + n_2$. *We define the boolean function* $g_1 \oplus g_2$ *as* $\forall x \in [\{0,1\}^m]^n : (g_1 \oplus g_2)(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_{n_1}) \oplus g_2(x_{n_1+1}, \ldots, x_n)$.

Now we can state our last theorem. Its proof can be found in [Ma99].

Theorem 4. *For all boolean functions* g_1, g_2

$$\max(\mathrm{ncc}(g_1) + \mathrm{ncc}(1 - g_2), \mathrm{ncc}(1 - g_1) + \mathrm{ncc}(g_2)) \leq \mathrm{ncc}(g_1 \oplus g_2) .$$

In particular, if both functions g_1 *and* g_2 *are very hard, then the function* $g_1 \oplus g_2$ *is also very hard.*

As a consequence we have an another very hard function.

Corollary 2. *For all even* n *the function* $h(x_1, \ldots, x_n) = (x_1 \leq x_2) \oplus \ldots \oplus (x_{n-1} \leq x_n)$ *is very hard.*

Acknowledgment

I would like to thank Pavol Ďuriš for many useful discussions.

References

[BNS89] Babai, L., Nisan, N., Szegedy, M., Multiparty protocols and logspace-hard pseudorandom sequences, Proceedings, 21st ACM STOC, 1–1, 1989.

[CFL83] Chandra, A. K., Furst, M. L., Lipton, R. J., Multi-party protocols, Proceedings, 15th ACM STOC, 94–99, 1983.

[DF89] Dolev, D., Feder, T., Multiparty communication complexity, Proceedings, 30th IEEE FOCS, 428–433, 1989.

[DF92] Dolev, D., Feder, T., Determinism vs. nondeterminism in multiparty communication complexity, SIAM Journal Computing 21, 889–893, 1992. Proceedings, 30th IEEE FOCS, 428–433, 1989.

[DR95] Ďuriš, P., Rolim J. D. P., A lower bound on the multiparty communication complexity, STACS'95, LNCS 900, Springer–Verlag, 350-360, 1995.

[Du99] Ďuriš, P., Multiparty communication complexity and very hard functions, 1999 (to appear).

[Hr97] Hromkovič, J., Communication complexity and parallel computing, An EATCS Series, Springer, 1997.

[KN97] Kushilevitz, E., Nisan, N., Communication complexity, Cambridge Univ. Press. xiii, 1997.

[Ma99] Maňuch, J., Multiparty Communication Complexity: Very Hard Functions, TUCS Report 282, 1999.