Electronic Colloquium on Computational Complexity, Report No. 6 (2000)

On operations of geometrical projection and of monotone extension *

E. A. Okol′nishnikova
Sobolev Institute of Mathematics
Kontjuga, 4, Novosibirsk, 630090,
E-mail: okoln@math.nsc.ru

Abstract

Some operations over Boolean functions are considered. It is shown that the operation of the geometrical projection and the operation of the monotone extension can increase the complexity of Boolean functions for formulas in each finite basis, for switching networks, for branching programs, and read-k-times branching programs, where $k \geq 2$.

It is established that if there exists a "big" gap between the complexities of realization of a function in the class of nondeterministic and in the class of deterministic branching programs (read-k-times branching programs), then it is possible to construct a function with a "big" gap between the complexity of realization of this function and the complexity of realization of its projection in the class of deterministic branching programs (read-k-times branching programs).

It is shown that there is some relation between the operation of the geometrical projection and the operation of the monotone extension. Namely, it is shown that if there exists a function with a "big" gap between the complexities of realization of a function and its monotone extension in a class of schemata, then it is possible to construct a function with a "big" gap between the complexity of realization of this function and the complexity of realization of its projection in the same class of schemata.

It is shown that there exist a functions of polynomial complexity and such that their monotone extension is a characteristic function of a NP-complete problem. The similar fact is valid for the operation of the geometrical projection. Namely, it is shown that there exist Boolean functions of polynomial complexity and such that their geometrical projection with respect to some subset of variables (the cardinality of this subset is less than the half of the cardinality of the set of variables) is the characteristic function of an NP-complete problem.

1 Introduction

In the complexity theory it is of interest to define operations causing the complexities of functions to increase. In the present paper the operation of the geometrical projection and the operation of the monotone extension are considered. Besides, the operations of the negation and of the projection [2, 3] are mentioned too.

ISSN 1433-8092

*This research was supported by the Russian Foundation for Basic Research (Grant 97–01–00848) and the Federal Target program "Integration" (Project 473).

The comparison between the complexities of Boolean functions and their geometrical projection is considered first. [1] In the classic mathematics the qualitative possibility of objects to become complicated under the operation of the projection and the complement of sets is well known (e.g., consider A- and B-sets in the descriptive theory of sets). The complexity theory makes it possible to estimate quantitatively similar complication.

The question about comparison of complexities between Boolean functions and their geometrical projections was set up by A.K. Pulatov. In 1976 it was shown by E.A. Okol'nishnikova that there exists a sequence of Boolean functions such that the complexity of the geometrical projection of Boolean functions is C times as great as the complexity of the Boolean functions for switching networks (unpublished). In 1977 A.K. Pulatov in cooperation with E.A. Okol'nishnikova showed that there exists a sequence of Boolean functions such that the complexity of the geometrical projection of a Boolean function is $\ln n$ times as great as the complexity of the Boolean functions for the formulas in the basis $(\&, \vee, \neg)$ [9]. In 1977 E.A. Okol'nishnikova showed that there exists a sequence of Boolean functions such that the complexity of the geometrical projection of the Boolean function is $n/\ln n$ times as great as the complexity of the Boolean functions for the formulas in each finite basis, and is $n/\ln^2 n$ times as great as the complexity of Boolean function for switching networks [8].

In the present paper the similar result is obtained for branching programs. For syntactic nondeterministic branching read-once-only programs there is no increase of complexity for Boolean functions under the operation of the geometrical projection.

The sequence of Boolean functions that is less complicated then their geometrical projection is the sequence considered in [8]. The projections of these functions have many subfunctions. It is possible to apply Nečiporuk's method to these functions and to obtain non-linear bounds on the complexity of these functions. The starting functions are comparatively simple.

The operation of the negation does not lead to the increase of complexities for many kinds of schemata without restrictions. But this operation can increase the complexity of functions for switching networks. It was shown [6, 1] that the negation of a Boolean function can be C times as great as the functions themselves for switching networks. There are many papers where it is shown that the operation of the negation can bring the complexity for schemata with restrictions to increase.

Another operation I would like to write about is the operation of the monotone extension. (The strict definition of this operation will be given below.) It is well known that it may be that this operation can bring the complexity to increase. Compare the complexity of the algorithm of recognition of the fact that a graph on m vertices is the $\lfloor m/2 \rfloor$-clique and the algorithm of recognition of the fact that a graph contains the $\lfloor m/2 \rfloor$-clique. But we have no good bounds for schemata without restrictions for the characteristic functions for these properties of the graph. Therefore, it is of interest to give examples of functions for which it is shown that their monotone extension is more complicated than the functions themselves. It seems that these examples can give a new view on the role of the negation in the complexity theory. In the present paper it is shown that there exists a sequence of functions $f(x_1, \ldots, x_n)$ such that their complexity is equal to Cn for formulas in the basis $(\&, \vee, \neg)$, and its monotone extension is more complicated (nearly quadratic with respect to the number of variables of the functions) for formulas in each finite basis, for switching networks, for branching programs, and for read-k-times only branching programs, where $k \geq 2$.

In the number of papers [2, 3] the operation of the projection (not geometrical) was consid-

[1] This definition differs from the operation of the projection given in, i.e., [2, 3].

ered. Namely, a function $f = (f_n)$ is a projection of $g = (g_n)$, $f \leq_{proj} g$,

$$f_n(x_1, \ldots, x_n) = g_{p(n)}(y_1, \ldots, y_{p(n)})$$

for some polynomial $p(n)$ and $y_j \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, 0, 1\}$. The number of j, where $y_j \in \{x_i, \bar{x}_i\}$, is called the multiplicity of x_i. The projection is read-once projection, $f \leq_{rop} g$, if the multiplicity of each x_i is bounded by 1. In the case of schemata without restrictions the considered operation does not cause the complexity to increase. But for some special schemata with restrictions (OBDD, FODD, k-OBDD, k-IBDD) this operation can bring the complexity to the exponential increase. It was also shown in [2] that if $f \leq_{rop} g$, and g has polynomial OBDD (FODD, k-OBDD, k-IBDD) size $s(n)$, then f has polynomial OBDD (FODD, k-OBDD, k-IBDD) size $s(p(n))$, respectively.

2 Geometrical projection

Recall some definitions. A deterministic branching program on the variables $\{x_1, \ldots, x_n\}$ is a directed acyclic graph with one source and sinks labelled with the constants 0 and 1. Each non-sink node is labelled with a variable x_i and has exactly two outgoing edged labelled with 1 and 0, respectively.

Let $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ be an input. A deterministic branching program computes a Boolean function in the following manner. The computation starts at the source. If one reaches a node labelled with x_i, one proceeds to the successor by the edge labelled with 1 if $a_i = 1$, and by the edge labelled with 0 if and $a_i = 0$. Finally, one reaches for the input a the exit $f(a)$. The size of a deterministic branching program $\mathcal{P}$ is the number of its nodes and is denoted by $|\mathcal{P}|$.

A deterministic branching program on the variables $\{x_1, \ldots, x_n\}$ becomes a nondeterministic branching program if we allow unlabelled (nondeterministic) nodes with two outgoing unlabelled edges (free edges).

A nondeterministic branching program $\mathcal{P}$ *computes* a Boolean function $f : \{0,1\}^n \to \{0,1\}$ as follows. For any input tuple $\tilde{a} = (a_1, \ldots, a_n)$, $a_i \in \{0,1\}$ for $i = 1, 2, \ldots, n$, we set $f(\tilde{a}) = 1$ if and only if there exists at least one (oriented) path from the source to the output node labelled with 1 (*accepting path* for $\tilde{a}$) such that all labels along this path don't contradict to an input tuple $\tilde{a}$ (i.e., if there is an edge labelled with d that outgoes from the node labelled with x_i on this path, then $a_i = d$). The size of a nondeterministic branching program $\mathcal{P}$ is the number of its labelled edges and is denoted by $|\mathcal{P}|$.

A branching program is called a read-k-times branching program if and only if for any i, $1 \leq i \leq n$, the nodes labelled with x_i occur at most k times in any computation path.

A switching network on the variables $\{x_1, \ldots, x_n\}$ is an indirected graph with one source and one sink. All edges are labelled with variables or with the negation of the variables. For any input tuple $\tilde{a} = (a_1, \ldots, a_n)$, $a_i \in \{0,1\}$ for $i = 1, 2, \ldots, n$, we set $f(\tilde{a}) = 1$ if and only if there exists at least one path from the source to the output node (*accepting path* for $\tilde{a}$) such that all labels along this path don't contradict to an input tuple $\tilde{a}$ (i.e., if there is an edge labelled with $x_i = d$ on this path, then $a_i = d$). The size of a switching network $\mathcal{S}$ is the number of its edges and is denoted by $|\mathcal{S}|$.

For the Boolean function f denote by $L_B(f)$ and $L_{B_0}(f)$ the complexity of f in the class of formulas in the basis B and $B_0 = (\vee, \&, \neg)$, respectively; by $S(f)$ the complexity of f in the class of switching networks; by $B(f)$ and $Bk(f)$ the complexity of f in the class of deterministic branching programs and deterministic read-k-times branching programs, respectively; by

$NB(f)$ and $NBk(f)$ the complexity of f in the class of nondeterministic branching programs and nondeterministic read-k-times branching programs, respectively.

Now we give the definition of geometrical projection. By $Pf_{x_{i_1},...,x_{i_k}}(x_{j_1},...,x_{j_{n-k}})$ denote the projection of the Boolean function $f(x_1,...,x_n)$ with respect to variables $x_{i_1},...,x_{i_k}$, where $\{j_1,...,j_{n-k}\} = \{1,2,...,n\} \setminus \{i_1,...,i_k\}$. Without loss of generality it can be assumed that $i_l = l$ for $l = 1,...,k$. By definition, put

$$Pf_{x_1,...,x_k}(x_{k+1},...,x_n) = \bigvee_{\sigma_1,...,\sigma_k} f(\sigma_1,...,\sigma_k,x_{k+1},...,x_n),$$

where $\sigma_i \in \{0,1\}$ for $i = 1,...,k$.

By A_k denote the set of tuples $\alpha = (\alpha_1,\ldots,\alpha_k)$, such that $\sum_{i=1}^{k} \alpha_i = \lfloor k/2 \rfloor$. It is clear that $|A_k| = \binom{k}{\lfloor k/2 \rfloor}$. Let a variable u_α corresponds to a tuple α.

Let us consider the Boolean function $F^{k,m}$ of sets of variables $U = \{u_\alpha, \alpha \in A_k\}$, $Y = \{y_1,...,y_k\}$ and $X_l = \{x_{l,1},...,x_{l,k}\}$, $1 \le l \le m$. By definition, put

$$F^{k,m}(U,Y,X_1,...,X_m) = \bigvee_{\alpha \in A_k} u_\alpha \cdot y_1^{\alpha_1} \cdot ... \cdot y_k^{\alpha_k} \cdot \left(\bigwedge_{l=1}^{m} (x_{l,1} \vee x_{l,1}^{\alpha_1}) \cdot \ldots \cdot (x_{l,k} \vee x_{l,k}^{\alpha_k}) \right).$$

The number of variables of this function is equal to

$$\binom{k}{\lfloor k/2 \rfloor} + k(m+1). \tag{1}$$

The projection of this function with respect to the variables $Y = \{y_1,...,y_k\}$ is equal to the function

$$PF_Y^{k,m}(U,X_1,...,X_m) = \bigvee_{\alpha \in A_k} u_\alpha \cdot \left(\bigwedge_{l=1}^{m} (x_{l,1} \vee x_{l,1}^{\alpha_1}) \cdot \ldots \cdot (x_{l,k} \vee x_{l,k}^{\alpha_k}) \right). \tag{2}$$

Explanation. Let $k = 4$, then the variable u_{0101} corresponds to the tuple $\alpha = (0101)$. In the setting of the function $F^{4,m}(U,Y,X_1,\ldots,X_m)$ the term $u_{0101} \cdot \bar{y}_1 y_2 \bar{y}_3 y_4 \cdot x_{1,2} \cdot x_{1,4} \cdot \ldots \cdot x_{m,2} \cdot x_{m,4}$ corresponds to the tuple $\alpha = (0101)$. Respectively, in the function $PF_Y^{4,m}(U,X_1,...,X_m)$ the term $u_{0101} \cdot x_{1,2} \cdot x_{1,4} \cdot \ldots \cdot x_{m,2} \cdot x_{m,4}$ corresponds to the tuple above.

Lemma 1
. *The formula size in the basis* $(\vee,\&,\neg)$ *for the Boolean function* $F^{k,m}(U,Y,X_1,...,X_m)$ *is less than* $5\binom{k}{\lfloor k/2 \rfloor} + k(m+2)$.

Proof [8]. It is easy to see that the function F can be represented as

$$F^{k,m}(U,Y,X_1,...,X_m) = \bigvee_{\alpha \in A_k} u_\alpha \cdot y_1^{\alpha_1} \cdot ... \cdot y_k^{\alpha_k} \cdot (y_1 x_{1,1} \cdot \ldots \cdot x_{m,1} \vee \bar{y}_1) \cdot \ldots \cdot (y_k x_{1,k} \cdot \ldots \cdot x_{m,k} \vee \bar{y}_k). \tag{3}$$

We need to estimate the formula size in the basis $B_0 = (\vee,\&,\neg)$ of the function $\bigvee_{\alpha \in A_k} u_\alpha \cdot y_1^{\alpha_1} \cdot \ldots \cdot y_k^{\alpha_k}$. For the sake of simplicity we will estimate the complexity of the function for the Π-circuits (series-parallel circuits). It is known that the formula size in the basis $B_0 = (\vee,\&,\neg)$ is equal to the size of Π-circuits for Boolean functions. Let $T^i(z_1,\ldots,z_p)$ be a contact tree that realizes all conjunctions $z_1^{\sigma_1},\ldots,z_p^{\sigma_p}$ such that $\sum_{j=1}^{p} \sigma_j = i$. The size of the contact tree $T^i(z_1,\ldots,z_p)$ denote by $\phi^i(p)$. The inductive construction of the contact tree $T^i(z_1,\ldots,z_p)$ is presented in Fig. 1. From this presentation it follows that

$$\phi^i(p) \le \phi^{i-1}(p-1) + \phi^i(p-1) + 2. \tag{4}$$

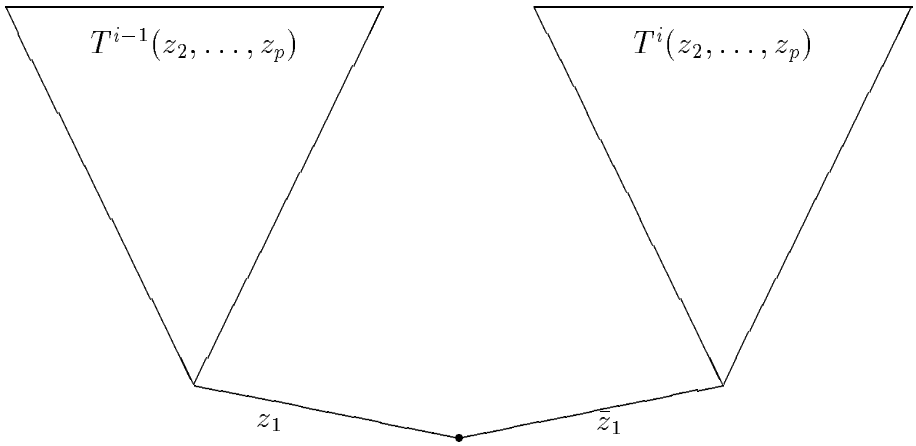


Fig.1. The contact tree $T^i(z_1,\ldots,z_p)$

We prove by induction on i that

$$\phi^i(p) \le \binom{p+2}{1+1} - 2. \tag{5}$$

It is easy to verify that $\phi^0(p) = p$, i. e. $\phi^0(p) = \binom{p+2}{1} - 2 = p$. Let the statement be valid for all $i, i \le i_0$. We prove that it is valid for i_0. From (4) and the inductive conjunction it follows that

$$\phi^{i_0}(p) \le \phi^{i_0-1}(p-1) + \phi^{i_0}(p-1) + 2 \le \binom{p+1}{i_0} - 2 + \binom{p+1}{i_0+1} - 2 + 2 \le \binom{p+2}{i_0+1} - 2.$$

The inequality (5) is proved.

It follows from (5) that

$$\phi^{\lfloor k/2 \rfloor}(k) \le 4\binom{k}{\lfloor k/2 \rfloor}. \tag{6}$$

We add contacts u_α to the ends of the tree, that correspond to the conjunctions $y_1^{\alpha_1} \cdot \ldots \cdot y_k^{\alpha_k}$, $\alpha \in A_k$. We join ends of the obtained circuit. From (6) it follows that the size of the constructed circuit is less that $5\binom{k}{\lfloor k/2 \rfloor}$. From this construction and the representation (3) it follows that the complexity of the function $F^{k,m}(U, Y, X_1, ..., X_m)$ does not exceed $5\binom{k}{\lfloor k/2 \rfloor} + k(m+2)$. Lemma 1 is proved.

Let n, $n \ge 2$, be a natural number. Put

$$k_0(n) = \lfloor \log_2 n + 1/2 \log_2 \log_2 n \rfloor \tag{7}$$

and

$$m_0(n) = \lfloor (n - \binom{k_0(n)}{\lfloor k_0(n)/2 \rfloor}) / k_0(n) \rfloor - 1. \tag{8}$$

From (1) we can obtain that the number of variables of the Boolean function F^{k_0,m_0} is less than n and

$$\binom{k_0}{\lfloor k_0/2 \rfloor} \sim C_0 n, \quad m \sim (1 - C_0) n / \log_2 n, \tag{9}$$

where $C_0 = \sqrt{2/\pi}$.

In what follows we drop subscripts k_0 and m_0 in the notation of F^{k_0,m_0} and PF^{k_0,m_0}.

B. A. Subbotovskaja [12] showed that for each finite basis B there exists a constant C_B that depends of the basis only and such that for each function f we have $L_B(f) \leq C_B \cdot L_{B_0}$.

It follows from this fact, Lemma 1, (9), that for the values of $k_0(n)$ and $m_0(n)$ given in (7) and (8) we have

Corollary 1. *$L_B F(U, Y, X_1, ..., X_m) \preceq n$.*

Lemma 2. *For the Boolean function $PF_Y(U, X_1, ..., X_m)$ we have*
(a) *$L_B(PF_Y) \succeq n^2/\log n$ for formulas in the finite basis B;*
(b) *$S(PF_Y) \succeq n^2/\log^2 n$ for switching networks;*
(c) *$B(PF_Y) \succeq n^2/\log^2 n$ for deterministic branching programs;*
(d) *$NB(PF_Y) \succeq n^{3/2}/\log n$ for nondeterministic branching programs.*

Proof (see [8]). If we substitute constants 0 and 1 instead of variables u_α, $\alpha \in A_k$, and constants 1 instead of variables X_i, $i \neq j$, we obtain $2^{\binom{k}{\lfloor k/2 \rfloor}}$ different functions depending on variables X_j, $1 \leq j \leq m$. The proof follows immediately from the proof of Nečiporuk [7] in the case of switching networks and formulas. We can use the remark of P. Pudlak [9] (see also Theorem 3 in the survey of A.A. Razborov [11]) in the case of branching programs. It allows us to apply Nečiporuk's method to the function PF_Y. Lemma 2 is proved.

Theorem 1. *For the Boolean functions PF_Y and F the following relations hold*
(a) *$L_B(PF_Y)/L_B(F) \succeq n/\log n$ for formulas in the finite basis B;*
(b) *$S(PF_Y)/S(F) \succeq n/\log^2 n$ for switching networks;*
(c) *$B(PF_Y)/B(F) \succeq n/\log^2 n$ for deterministic branching programs;*
(d) *$NB(PF_Y)/NB(F) \succeq \sqrt{n}/\log n$ for nondeterministic branching programs;*
For $k \geq 2$ we have
(e) *$Bk(PF_Y)/Bk(F) \succeq n/\log n$ for deterministic branching read-k-times programs;*
(f) *$NBk(PF_Y)/NBk(F) \succeq \sqrt{n}/\log n$ for nondeterministic branching read-k-times programs.*

Proof. From Corollary 1 and Lemma 2 it follows that the statement (a)–(d) are valid.

The representation (3) of the function F in Lemma 1 corresponds to the representation of the function F in the class of read-2-times branching programs. Therefore the claims (e) и (f) are also valid.

Theorem 1 is proved.

Theorem 2. *For nondeterministic read-once-only branching program for the Boolean function $f(X, Y)$ we have*

$$NB1(P_Y(f)) \leq NB1(f).$$

Proof. Let us consider a read-once-only program realizing $f(X, Y)$. There are no null-chains in read-once-only programs. Therefore, we can replace all labelled edges that go out from nodes labelled with y_i by free edges. The obtained branching program computes the projection of f with respect to Y. Theorem 2 is proved.

The following theorem shows, in particular, that if for a Boolean function $f(X)$ there exists a "big" gap (more than quadratic with respect to $NB(f)$) between $NB(f)$ and $B(f)$ ($NBk(f)$

and $Bk(f)$, respectively), then it is possible to give an example of the function $g(X,Y)$ with a "big" gap between the sizes of deterministic branching programs for $g(X,Y)$ and $P_Y(X)$ (of deterministic read-k-times branching program for $g(X,Y)$ and $P_Y(X)$).

And vice versa if for a Boolean function $g(X,Y)$ there exists a "big" gap between the sizes of deterministic read-once-only branching programs for a Boolean function and its projection then we can construct an example of a function with a "big" gap between the sizes of nondeterministic and deterministic read-once-only branching programs.

Theorem 3. (1a) *For any Boolean function $f(X)$ of n variables it is possible to construct a Boolean function $g(X,Y)$ that depends on not more than $n+(NB(f))^2$ variables and such that*

$$B(P_Y g(X,Y)) - B(g(X,Y)) \geq B(f(X)) - NB(f(X)) - 8(NB(f))^2.$$

(1b) *For any Boolean function $f(X)$ of n variables it is possible to construct a Boolean function $g(X,Y)$ that depends on not more than $n+(NBk(f))^2$ variables and such that*

$$Bk(P_Y g(X,Y)) - Bk(g(X,Y)) \geq Bk(f(X)) - NBk(f(X)) - 8(NBk(f))^2.$$

(2) *Let $h(X,Y)$ be a Boolean function. Then*

$$B1(P_Y(h)) - 1/2NBP1(P_Y(h)) \geq BP1(P_Y(h)) - BP1(h).$$

Proof. (1a). The number of unlabelled nodes in the nondeterministic branching program does not exceed $(2NB(f))^2$ (see, e.g., the proof of Theorem 1 in [4]). Let us enumerate all unlabelled nodes from 1 to l and consider the j-th unlabelled node. We label this node with a new variable y_j, label one of free edges going out of this node with 1, label the second edge going out of this node with 0. The obtained program is the deterministic branching program that realizes a function $g(X,Y)$. The Boolean function $g(X,Y)$ depends on no more than $n+(2NB1(f))^2$ variables. The complexity of this function does not exceed $NB(f)+4(NB(f))^2$ in the class of nondeterministic programs. The geometrical projection of the function $g(X,Y)$ with respect to the variables Y is the function $f(X)$. The first part of theorem 3 is proved for branching programs.

The case (1b) for read-k-times branching programs can be considered similarly.

(2). Let us consider a deterministic read-once-only program computing $g(X,Y)$. There are no null-chains in read-once-only programs . Therefore we can replace all nodes labelled with y_i by unlabelled nodes and replace edges outgoing from these nodes by free edges. The obtained nondeterministic branching program computes the projection of the function g with respect to the variables Y. So, we have $NBP1(P_Y(h)) \leq BP1(h)$. From this fact it follows that the statement (2) of Theorem is also valid.

It is known that there exist Boolean functions with exponential gap between deterministic and nondeterministic complexities in the class of read-once-only branching programs (see, e.g. [5]). Therefore the following lemma follows from the first part of theorem 3.

Corollary 2 *There exist Boolean functions with the exponential gap between the complexity of the functions and their geometrical projections in the class of deterministic read-once-only branching programs.*

Remark 1. For almost all Boolean functions the complexity of the projection is less than the complexity of the function. It is valid for formulas, switching networks, branching programs,

schemata of functional elements and so on. Moreover, there exist $2^{2^{n-1}+1}-1$ functions of n variables such that their projection with respect to each subset of variables is equal to 1. In fact, let us consider a set of functions that are equal to 1 on the odd levels of the n-cube, and accept arbitrary values at the vertices of the even levels. Similarly we can consider a set of functions that are equal to 1 on even levels of the n-cube, and accept arbitrary values at the vertices of odd levels. The union of these sets gives the estimate we need.

3 Monotone extension

A Boolean function $g(x_1,\dots,x_k)$ is said to be a monotone extension of the function $f(x_1,\dots,x_k)$ if $g(\beta_1,\dots,\beta_k)=1$ iff there exists a tuple $(\alpha_1,\dots,\alpha_k)$, $(\alpha_1,\dots,\alpha_k)\preceq(\beta_1,\dots,\beta_k)$ such that $f(\alpha_1,\dots,\alpha_k)=1$. (Here, as usual, $(\alpha_1,\dots,\alpha_k)\preceq(\beta_1,\dots,\beta_k)$ denotes that $\alpha_1\le\beta_1,\dots,\alpha_k\le\beta_k$.)

Below we give an example of Boolean function that is less complicated than its monotone extension.

Let us consider the Boolean function $H^{k,m}$ of sets of variables $U=\{u_\alpha,\alpha\in A_k\}$ and $X_l=\{x_{l,1},...,x_{l,k}\}$, $l=\overline{1,m}$. By definition, put

$$H^{k,m}(U,X_1,...,X_m)=\bigvee_{\alpha\in A_k}u_\alpha\cdot\left(\bigwedge_{l=1}^{m}x_{l,1}^{\alpha_1}\cdot\ldots\cdot x_{l,k}^{\alpha_k}\right). \tag{10}$$

The number of variables of this function is equal to $\binom{k}{\lfloor k/2\rfloor}+mk$. The monotone extension of this function is the function $PF_Y^{k,m}$ considered in the section 2 (see (2)), i.e.,

$$MH^{k,m}(U,X_1,...,X_m)\equiv PF_Y^{k,m}(U,X_1,...,X_m).$$

Lemma 3. *The formula size in the basis $(\vee,\&,\neg)$ for the function $H^{k,m}(U,X_1,...,X_m)$ is less than $5\binom{k}{\lfloor k/2\rfloor}+2km$.*

Proof. It is easy to see that

$$H(U,X_1,...,X_m)\equiv\bigvee_{\alpha\in A_k}u_\alpha\cdot x_{1,1}^{\alpha_1}\cdot...\cdot x_{1,k}^{\alpha_k}$$

$$\&(x_{1,1}x_{2,1}\cdot\ldots\cdot x_{m,1}\vee\bar{x}_{1,1}\bar{x}_{2,1}\cdot\ldots\cdot\bar{x}_{m,1})\cdot\ldots\cdot(x_{1,k}x_{2,k}\cdot\ldots\cdot x_{m,k}\vee\bar{x}_{1,k}\bar{x}_{2,k}\cdot\ldots\cdot\bar{x}_{m,k}).$$

Therefore the complexity of this function does not exceed $5\binom{k}{\lfloor k/2\rfloor}+2km$. The proof of this fact is similar to the proof of Lemma 1. Lemma 3 is proved.

In what follows we shall consider functions H^{k_0,m_0} and MH^{k_0,m_0}, where the values of k_0 and m_0 are given in (7) and (8). We set $H\equiv H^{k_0,m_0}$ and $MH\equiv MH^{k_0,m_0}$.

From Lemma 3, (9), and the result of B. A. Subbotovskaya it follows

Corollary 3. *For each finite basis we have $L_BH^{k,m}(U,Y,X_1,...,X_m)\preceq n$.*

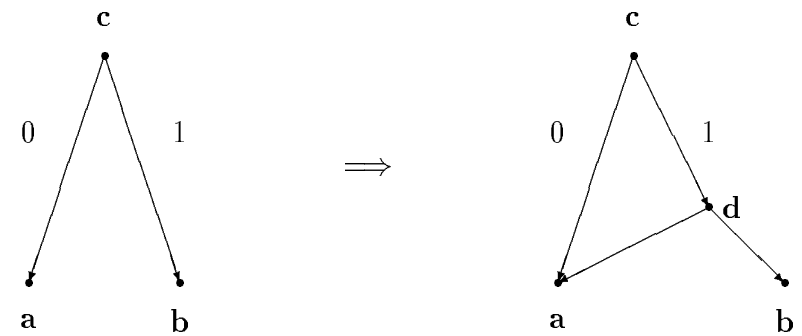
Theorem 4.
(а) *$L_B(MH)/L_{B(H)}\succeq n/\log n$ for formulas in the finite basis B;*
(б) *$S(MH)/L(H)\succeq n/\log^2 n$ for switching networks;*
(в) *$B(MH)/B(H)\succeq n/\log^2 n$ for deterministic branching programs;*
(г) *$NB(MH)/NB(H)\succeq\sqrt{n}/\log n$ for nondeterministic branching programs.*
For $k\ge 2$ we have
(д) *$Bk(MH)/Bk(H)\succeq n/\log^2 n$ for deterministic branching read-k-times programs;*
(е) *$NBk(MH)/NBk(H)\succeq\sqrt{n}/\log n$ for nondeterministic read-k-times branching programs.*

Proof of this theorem is similar to the proof of Theorem 1.

Theorem 5. *For nondeterministic read-once-only branching programs we have*

$$NB1(M(f)) \leq NB1(f).$$

Proof. We can transform the nondeterministic read-once-only branching program for the function f into the nondeterministic read-once-only branching program for the function $M(f)$. Let us consider a node, for example, $\mathbf{c}$, labelled with x that has two edges going out of it: the edge $(\mathbf{c}, \mathbf{b})$ labelled with 1 and the edge $(\mathbf{c}, \mathbf{a})$ labelled with 0 (see Fig. 2).



program for f program for $M(f)$

Рис. 2

Let $\mathbf{d}$ be a new unlabelled node with two unlabelled edges $(\mathbf{d}, \mathbf{a})$ and $(\mathbf{d}, \mathbf{b})$ that go out of this node. In spite of the edge $(\mathbf{c}, \mathbf{b})$ labelled with 1 we consider an edge $(\mathbf{c}, \mathbf{d})$ labelled with 1. Theorem is proved.

Remark 2. For almost all Boolean functions the complexity of their monotone extension does not exceed the complexity of the function. It is valid for formulas, switching networks, branching programs, and so on. Moreover, there exist many functions of n variables ($\sim 2^{2^n-1}$) functions of n variables, namely functions that are equal to 1 at the vertex $(0, \ldots, 0)$, and such that their monotone extension is the function identically equaled to 1.

The following theorem shows that there is some relation between the operation of the geometrical projection and the operation of the monotone extension.

Theorem 6. *Let $f(X)$ be a Boolean function of n variables. There exists a Boolean function $g(X, Y)$ of $2n$ variables that corresponds to $f(X)$, and such that*

(a) the difference between the complexity of the function $g(X, Y)$ and the complexity of the function $f(X)$ does not exceed Cn for a number of classes of schemata, in particular, for schemata with functional elements, for formulas, branching programs and so on;

(b) the projection $Pg_Y(X)$ of f with respect to variables Y coincides with the monotone extension $Mf(X)$ of the function f.

Proof. It is easy to check that the Boolean function

$$g(X, Y) = f(Y) \cdot (x_1 y_1 \vee \bar{y}_1) \cdot \ldots \cdot (x_n y_n \vee \bar{y}_n)$$

satisfies the condition (a) of the Theorem.

Let g and h be Boolean functions that depend of the same set of variables. We say that $g \leq h$ if for each tuple $\tilde{a}$ the relation $g(\tilde{a}) \leq h(\tilde{a})$ holds. Without loss of generality we can assume that for a conjunction $\mathcal{K} = x_1, \ldots, x_p, \bar{x}_{p+1} \ldots, \bar{x}_q$ the inequality $\mathcal{K} \leq f$ holds. Then $\mathcal{K}(x_1 y_1 \vee \bar{y}_1) \cdot \ldots \cdot (x_n y_n \vee \bar{y}_n) \leq g(X, Y)$, i.e.,

$$x_1 y_1, \ldots, x_p y_p, \bar{y}_{p+1} \ldots, y_q(x_{q+1} y_{q+1} \vee \bar{y}_{q+1}) \ldots (x_n y_n \vee \bar{y}_n) \leq g(X, Y).$$

Then $x_1, \ldots, x_p \leq P g_Y(X, Y)$. It is clear that $x_1, \ldots, x_p \leq Mf$. It is easy to see that from this fact follows that $P g_Y(X) \equiv Mf(X)$. The statement (b) of Theorem is also proved.

Remark 3. Let us consider Boolean function on the set of variables $X_m = \{x_{i,j}\}$, where $1 \leq i < j \leq m$. Let CLIQUE_m ($\mathrm{CLIQUE} - \mathrm{ONLY}_m$) be the characteristic property of the graph on m vertices to contain an $\lfloor m/2 \rfloor$-clique (to be exactly an $\lfloor m/2 \rfloor$-clique, respectively). The function CLIQUE_m is the monotone extension of the function ($\mathrm{CLIQUE} - \mathrm{ONLY}_m$). The function CLIQUE_m has a polynomial complexity. The function ($\mathrm{CLIQUE} - \mathrm{ONLY}_m$) is a characteristic function of NP-complete problem. Therefore, it is possible to suppose the possibility of the existence of the exponential gap between Boolean functions and their monotone extensions. It follows from Claim 4 that there exist Boolean functions of polynomial complexity and such that their projection with respect to some subset of variables (the cardinality of this subset is less than the half of the cardinality of the set of variables) is the characteristic function of an NP-complete problem.

References

[1] Avgustinovich, S.V. (1980) On an approach to obtaining lower estimates of the complexity for Boolean functions (in Russian) *Metody Diskret. Anal.* **35**, 3–9.

[2] Bollig, B., and Wegener, I. (1996) Read-once projections and formal circuit verification with binary decision diagrams *Proc. of the 13th Ann, Symp. on Theoretical Aspects of Computer science (STACS'96)*, LNCS, 1996, **1046**, 491–502, Springer-Verlag, Berlin/New York

[3] Bollig, B., and Wegener, I. (1998) Completeness and noncompleteness results with respect to read-once projections *Information and Computation*, **143**, No. 1, 24–33.

[4] A. B. Borodin, A. A. Razborov, and R. Smolensky (1993) On lower bounds for read-k-times branching programs, *Comput. Complexity* **3**, No. 1, 1–18.

[5] S. Jukna, A. Razborov, P. Savicky, and I. Wegener (1997) On P versus $NP \bigcap co-NP$ for decision trees and read-once branching programs *Proc. of the 22nd Int. Symp. on MFCS*, LNCS, **1295**, 319–326, Springer-Verlag, Berlin/New York.

[6] A.V. Kuznetsov (1958) On one property of functions realized with nonplanar schemata without repetitions (in Russian) *Trudy Mat. Inst. Steklov* **51**, 174–185.

[7] E.I. Nečiporuk (1966) On a Boolean function *D.A.N. SSSR* **169** No. 4, 765–766. (English translation in *Soviet Math. Dokl.* **2**, No. 4, 999–1000).

[8] E.A. Okol'nishnikova (1977) On the comparison of complexities of Boolean functions and their projections (in Russian) *Metody Diskret. Anal.* **31**, 76–80.

[9] P. Pudlák (1987) The hierarchy of Boolean circuits. *Comput. Artificial Intelligence* **6**, No. 5, 449–468.

[10] A.K. Pulatov (1977) On the influence of zero-chains on the complexity of realization of Boolean functions by contact schemes (in Russian) *Metody Diskret. Anal.* **30**, pp. 30–37.

[11] A. A. Razborov (1991) Lower bounds for deterministic and nondeterministic branching programs, *Fundamentals of Computation Theory* (Gosen, 1991), Lecture Notes in Comput. Sci. Vol. 529, Springer-Verlag, Berlin, pp. 47–60.

[12] B.A. Subbotovskaya (1963) Comparison of bases in the realization by formulas of functions of the algebra of logic *D.A.N. SSSR* **149** No. 4, 784–787. (English translation in *Soviet Math. Dokl.* **2**, No. 4, 478–481).

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'