Economical formulae for symmetric boolean functions

preprint, 01.05.2025*

Igor S. Sergeev†

1 Introduction

In the present paper we prove upper bounds on the complexity and the depth of symmetric boolean functions. A function is called *symmetric* if it preserves its values under permutations of arguments. In the boolean case, it means that values of the functions depend only on the arithmetic sum of variables.

Problems of efficient implementation of symmetric functions in various computational models always attracted attention of complexity theorists. Applications, both practical and theoretic, interest in methods of computation of certain symmetric functions, as well as of some subclasses (threshold, periodic functions), and sometimes of symmetric functions in general. One of the most popular applications are parallel circuits for integer multiplication based on efficient computation of the arithmetic sum of bits. Some known parallel circuits for integer division and other operations (see e.g. [2, 6]) also involve fast subcircuits for symmetric functions.

In this paper we restrict our consideration to the computational model of formulae over complete finite bases, being interested primarily in binary bases. It is the optimization of the formula depth that is of interest in the problem of fast multiplication of numbers.

Recall that formulae are essentially circuits of functional elements with fan-out 1. More formally, the set of *formulae* over a basis B, the *complexity* of a formula, the *depth* of a formula, and the function *implemented* by the formula are defined inductively as follows: 0) the basis constants are formulae of complexity and depth 0; 1) the symbols of variables are formulae of complexity 1 and depth 0 that implement the corresponding identical functions;

*09.09.2025: a recent result of I. Orzel [22] is cited now.

†e-mail: isserg@gmail.com

2) the expression $G(F_1, ..., F_k)$, where G is a symbol denoting a non-constant k-ary function $g \in B$, and F_i is a formula of complexity L_i and depth D_i implementing the function f_i, is a formula of complexity $L_1 + \ldots + L_k$ and depth $\max\{D_1, ..., D_k\} + 1$, and it implements the function $g(f_1, \ldots, f_k)$. Informally, the complexity of a formula is the number of symbols of variables in it.

The complexity and the depth of a formula F over a basis B will be denoted by $\Phi_B(F)$ and $\mathsf{D}_B(F)$, respectively. The complexity (depth) of a function f is defined as the minimal complexity (depth) of formulae implementing it, and is denoted by $\Phi_B(f)$ ($\mathsf{D}_B(f)$). For a set (class) of functions $\mathcal{F}$, $\Phi_B(\mathcal{F})$, $\mathsf{D}_B(\mathcal{F})$ are understood as the maximal values of complexity and depth of functions from $\mathcal{F}$.

Let S_n denote the class of all symmetric boolean functions of n variables. Let T_n^k denote the *threshold* symmetric function of n variables with threshold k: by definition[1], $T_n^k(x_1, \ldots, x_n) = (x_1 + \ldots + x_n \geq k)$. The function $\mathrm{Maj}_n = T_n^{n/2}$ is also called the *majority function*[2].

The known upper bounds on the complexity and the depth of implementation of symmetric functions by both formulae and circuits over complete bases are related to the efficient implementation of the boolean (n, m)-operator $C_n(x_1, \ldots, x_n) = [C_{n,m-1}, ..., C_{n,0}]$ counting the number of ones in the vector of boolean variables $(x_1, \ldots, x_n)$, where $m = \lceil \log_2(n + 1) \rceil$. Indeed, any symmetric Boolean function $f \in S_n$ is naturally represented as a composition of the operator C_n and some boolean function φ of m variables:

$$f(x_1, \ldots, x_n) = \varphi(C_n(x_1, \ldots, x_n)). \tag{1}$$

Almost all known complexity- and depth-efficient formulae and circuits for C_n are composed from compressors[3]. A binary (k, l)-compressor is a circuit (or formula) implementing a boolean operator $(x_1, \ldots, x_k) \to (y_1, \ldots, y_l)$ defined by the condition $\sum 2^{a_i} x_i = \sum 2^{b_j} y_j$, where $k > l$ and $a_i, b_j \in \mathbb{Z}$. It is easy to construct a circuit from compressors that reduces the summation of n numbers to the addition of $O(1)$ numbers. The simplest example is the (3,2)-compressor[4]. It calculates the sum of three bits according to the rule $x_1 + x_2 + x_3 = 2y_2 + y_1$.

There is evidence that the idea of compressors for reducing the execution time of arithmetic operations arose in the design of electronic devices even

[1]Here and in similar cases below, the value of the function is assumed to be 1 if the expression on the right-hand side is true, and 0, otherwise.

[2]The majority function is often defined as $T_n^{(n+1)/2}$. For further consideration, such violation of definition is not essential.

[3]Another term for compressor in the context of summation is CSA (carry save adder).

[4]Often denoted as FA_3 (full adder).

before 1960 among various groups of engineers (in particular, both in the West and in the USSR). In the early 1960s, in several theoretical papers (for example, [21, 37]) there was proved that, using compressor circuits, multiple addition (and, in particular, the C_n operator) may be implemented with logarithmic depth, i.e. $\mathsf{D}_B(C_n) \asymp \log n$ in a full basis B.

In the same years, an alternative approach was proposed in [1] — to use a notation of numbers with an extended set of digits[5], allowing addition to be performed without carries, i.e. with depth $O(1)$, see also [38, §3.2]. But ultimately this method also allows description in terms of compressors, only not binary ones.

Thus, from (1) it immediately follows[6] that

$$\mathsf{D}_B(S_n) \le \mathsf{D}_B(C_n) + O(\log n) \asymp \log n$$

in an arbitrary complete boolean basis B. The corresponding synthesis method (of boolean circuits) is discussed in [18]. Further, in view of the obvious relation $\Phi_B(f) \le 2^{O(\mathsf{D}_B(f))}$ (in binary bases, simply $\Phi_B(f) \le 2^{\mathsf{D}_B(f)}$) we immediately obtain $\Phi_B(S_n) = n^{O(1)}$. However, about ten years passed before V. M. Khrapchenko [14] published the first proof of the polynomial complexity of formulae for symmetric functions in the early 1970s.

Over the next 50 years, our understanding of efficient ways to compute symmetric functions has improved. The general idea, which has proven itself in circuit-type computational models, is to select a basic set of functions (usually also symmetric) that have relatively simple implementation, and whose values uniquely determine the arithmetic sum of the variables. Then any symmetric function can be computed following a rule similar to (1).

For boolean circuits, the components of the operator C_n are currently considered to be the optimal basic set (at least in binary bases). Note that the record upper bound for the complexity of circuits for symmetric functions was obtained in [7] via special compressors. In the case of switching circuits, the best known means of expressing symmetric functions is a set of periodic functions with small mutually prime periods [19]. However, in the formula model, the optimal choice of an encoding set seems to be less homogeneous. It is advantageous to include in the basic set a part of the digits of several operators $C_n^{(q)}$, calculating the sum of n boolean variables in q-ary number systems, for small prime q (here $C_n^{(q)}$ coincides with C_n for $q = 2$) [31, 32], as well as components of operators calculating the sum of boolean variables approximately, with controlled accuracy [33].

[5]Such a notation is ambiguous.

[6]Due to the trivial bound $O(n)$ for the depth of an arbitrary function of n variables.

The components $C^{(q)}_{n,i}$ of the operator $C^{(q)}_n$ are interpreted as digits from $\mathbb{Z}_q$ given in a suitable binary encoding. The operator is implemented by circuits of compressors, this time q-ary. The definition of a compressor over $\mathbb{Z}_q$ is analogous to the definition of a binary compressor: it is a circuit (or formula) that performs the transformation $\mathbb{Z}^k_q \to \mathbb{Z}^l_q : (x_1, \ldots, x_k) \to (y_1, \ldots, y_l)$ subject to the condition[7] $\sum q^{a_i} x_i = \sum q^{b_j} y_j$, where $k > l$ and $a_i, b_j \in \mathbb{Z}$. The theory of optimal synthesis of formulae from compressors was developed by M. Paterson, N. Pippenger and U. Zwick in [24, 26, 27]. In particular, the method [24] allows one to obtain upper bounds for individual digits of summation operators. A number of techniques for constructing efficient compressors were also proposed in these works.

An economical (but nonconstructive) procedure for constructing formulae for the approximate computation of threshold functions was proposed by L. Valiant[36]. With its help, it is possible to determine the value of the sum of variables with a given accuracy [33] (the higher the accuracy, the more complex the formula). Note that the idea of locating the values of the sum in intervals also plays a key role in the method [35] of implementing threshold symmetric functions by switching circuits.

The methods of formula synthesis discussed below can be applied to any complete basis. However, since binary bases are of a greater importance in theory, we select one sufficiently expressive binary basis from each complexity class to state the results[8]: the basis $\mathcal{B}_2$ of all binary boolean functions and the standard basis $\mathcal{B}_0 = \{\wedge, \vee, {}^-\}$.

First, we note that there remain significant gaps between the known lower and upper bounds for the complexity of symmetric functions. The bounds

$$\Phi_{\mathcal{B}_0}(S_n) \succeq \Phi_{\mathcal{B}_0}(\mathrm{Maj}_n) \succeq n^2, \qquad \Phi_{\mathcal{B}_2}(S_n) \succeq \Phi_{\mathcal{B}_2}(\mathrm{Maj}_n) \succeq n \log n$$

were proved by V. M. Khrapchenko in [13] and by M. Fischer, A. Meyer, and M. Paterson [8] more than 40 years ago, respectively, and have not been improved for a long time. Though the second bound was generalized by D. Yu. Cherukhin [4] to $\Phi_B(\mathrm{Maj}_n) \succeq n \log n$ in any finite boolean basis B. The latter bound is not too far from the truth, since with a special choice of basis, for example, $B = S_l$, it is easy to obtain upper bounds for $\Phi_B(C_n), \Phi_B(\mathrm{Maj}_n)$ of the form $n^{1+\varepsilon(l)}$, where $\varepsilon(l) \to 0$ (see also [15]). In 2025

[7] The summation is performed in $\mathbb{Z}$.

[8] Recall that complete binary boolean bases are divided into two equivalence classes in terms of formula complexity (the order of complexity of any function in equivalent bases is the same). The classification of binary bases by formula depth includes at least four equivalence classes [20] (the depth of a function in equivalent bases is the same up to an additive constant).

I. Orzel [22] established new stronger bounds:

$$\Phi_{\mathcal{B}_2}(S_n) \succeq \Phi_{\mathcal{B}_2}(\mathrm{Maj}_n) \succeq n^2.$$

Apparently, no lower bounds are known for the depth of formulas over the bases under consideration other than those that trivially follow from the complexity estimates above.

In what follows, for the sake of consistency, the constant C in the upper bounds for the complexity $n^{C+o(1)}$ or the depth $(C+o(1))\log_2 n$ of functions of n variables will be called the *index* of complexity and depth, respectively. The known upper bounds for the complexity and the depth of formulae for the operator C_n, the class S_n, and the function Maj_n are given in Table 1 in chronological order. The bounds obtained in this paper are marked with the sign $\star$.

	$\Phi_{\mathcal{B}_0}$	$\Phi_{\mathcal{B}_2}$	$\mathsf{D}_{\mathcal{B}_0}$	$\mathsf{D}_{\mathcal{B}_2}$
C_n	4.62 [14] (1972) 4.60 [24] (1990) 4.57 [27] (1993) 4.54 [30] (2012) 4.47 [32] (2014) 3.91 [33] (2016) 3.77 $\star$	3.32 [28] (1978) 3.16 [24] (1990) 3.13 [27] (1993) 3.06 [30] (2012) 3.03 [32] (2014) 2.84 [33] (2016) 2.82 $\star$	5.12 [16] (1978) 5.07 [24] (1990) 4.95 [25] (1991) 4.93 [10] (1993) 4.87 [31] (2013) 4.14 [33] (2016) 3.96 $\star$	4, folklore 3.71 [24] (1990) 3.57 [25] (1991) 3.48 [27] (1993) 3.44 [10] (1993) 3.34 [31] (2013) 3.02 [33] (2016) 2.98 $\star$
S_n	4.93 [14] (1972) 4.85 [24] (1990) 4.82 [30] (2012) 4.48 [32] (2014) 4.01 [33] (2016) 3.77 $\star$	3.55 [29] (1974) 3.42 [23] (1977) 3.37 [28] (1978) 3.30 [24] (1990) 3.23 [30] (2012) 3.04 [32] (2014) 2.95 [33] (2016) 2.85 $\star$	4.88 [31] (2013) 4.24 [33] (2016) 3.96 $\star$	3.81 [24] (1990) 3.34 [31] (2013) 3.10 [33] (2016) 2.98 $\star$
Maj_n	3.64 $\star$	2.77 $\star$	3.81 $\star$	2.91 $\star$

Table 1: Upper bounds for indices of complexity and depth of symmetric functions

The first accurate estimates of the complexity and the depth of the operator C_n, as well as the class of symmetric functions, were obtained by

V. M. Khrapchenko [14, 16] (over the basis $\mathcal{B}_0$). A series of papers [29, 23, 28] by various authors in the 1970s dealt with refining the estimates over the basis $\mathcal{B}_2$. A systematic approach to the synthesis of economical formulas from compressors was proposed in the early 1990s [26]. The author added the idea of exploiting modular arithmetic (non-binary compressors) [31, 32] and approximate summation [33]. As shown by L. Valiant [36], threshold symmetric functions admit sufficiently economical monotone formulas. In the generalized form due to R. Boppana [3] the result has the form $\Phi_{\mathcal{B}_M}(T_n^m) \preceq m^{4.28} n \log n$, where $\mathcal{B}_M = \{\vee, \wedge\}$.

The general scheme for computing symmetric functions, which leads to the bounds obtained in this paper, is illustrated in Fig. 1. Here, σ denotes the arithmetic sum of boolean variables: $\sigma = x_1 + \ldots + x_n$.

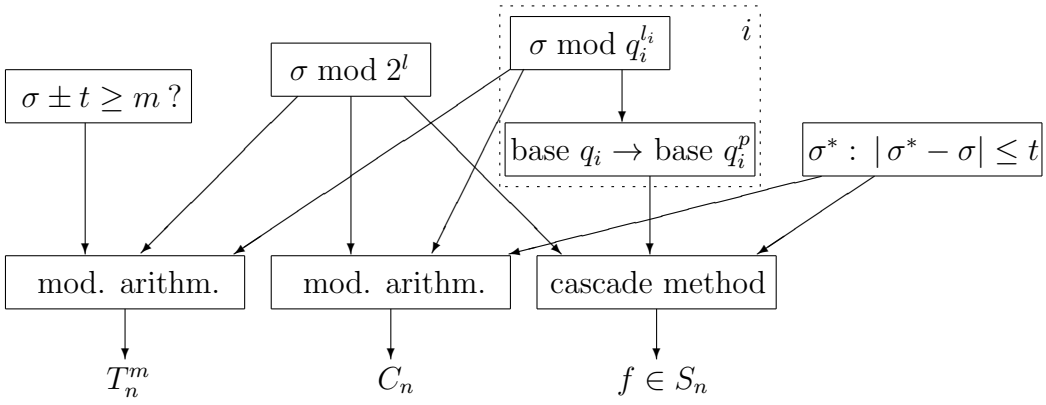

Figure 1: General scheme for computing symmetric functions

Essentially, the computations are as follows. The segment $[0,\ n]$ is covered by intervals of length $2t$. The value σ is determined by the interval number σ' and the remainders $\sigma_2 = \sigma \bmod 2^l$, $\{\sigma_{q_i} = \sigma \bmod q_i^{l_i}\}$, where q_i are the bases of additional number systems, given that $2^l \cdot \prod q_i^{l_i} \geq 2t$. Thus, the least significant digits of the sum are calculated directly, the most significant digits are principally determined by the interval number, and the remaining middle digits may be recovered by means of simple modular arithmetic in the spirit of the Chinese Remainder Theorem. This is how the operator C_n is implemented.

An arbitrary symmetric function $f \in S_n$ may be efficiently computed from a coding set of length approximately $\log_2 n$ by the cascade method. The initial coding set is constituted by the digits of the numbers σ', σ_2, and $\{\sigma_{q_i}\}$. Since a binary code of the number σ_q, written in the q-ary number system, is significantly longer than $\log_2 \sigma_q$ for $q \neq 2$, the number should be

first rewritten in the base q^p with the standard binary encoding of digits. This ensures that the total length of the coding set is asymptotically $\log_2 n$.

Additionally, we note that the threshold symmetric functions T_n^m can be calculated somewhat simpler. If the location of the sum σ with respect to the point m up to an admissible error $\pm t$ is known, then using the remainders σ_2 and $\{\sigma_{q_i}\}$, we can further refine the position of σ within the interval $(m-t,\, m+t)$ and ultimately determine whether $\sigma \geq m$ holds.

Of the author's works [31, 32, 33] on this topic publications [31, 33] are brief notes. The full potential of the method is not revealed in them. This work, in addition to a generally more detailed presentation, includes:

– a more efficient procedure for approximate computation of the arithmetic sum of variables compared to [33];

– the above-mentioned refinement for the complexity and the depth of threshold functions;

– improved designs of ternary (4,2)-compressors compared to [31, 32];

– the use of additional number systems with bases $q > 3$.

All this together leads to an improvement of the results [33], see Table 1.

That additional number systems allow to speed up calculations follows from Table 2. It contains the best known upper bounds on the complexity and the depth of formulae for periodic functions $\sum x_i \bmod q$, that is, for the lower components of the operators $C_n^{(q)}$. Only those results that are not superseded by the bounds in Table 1 are shown.

q	$\Phi_{\mathcal{B}_0}$	$\Phi_{\mathcal{B}_2}$	$\mathsf{D}_{\mathcal{B}_0}$	$\mathsf{D}_{\mathcal{B}_2}$
3	2.59 [19]	2 [20]	2.80 [34]	2 [20]
5	3.22 [34]	—	3.35 [34]	—
7	3.63 [34]	2.59 [17]	3.87 [34]	2.93 [34]

Table 2: Upper bounds for indices of complexity and depth of functions $\sum x_i \bmod q$

The complexity of computing the sum digits in the compressor method smoothly increases from the lower to the higher ones. Therefore, the ability to quickly calculate the sum modulo q is the reason for including part of the components of the operator $C_n^{(q)}$ in the basic set for encoding symmetric functions.

The best parallel algorithms for implementing many arithmetic operations are based on fast computation of symmetric functions. The most popular example is integer multiplication. If Σ_n and M_n denote the operators of

multiplication and addition of n-bit binary numbers, then

$$\mathsf{D}_B(M_n) \le \mathsf{D}_B(C_n) + \mathsf{D}_B(\Sigma_{2n}) + O(\log\log n).$$

Indeed, the multiplication of n-bit numbers in the standard way reduces to n-fold addition of n-bit numbers. If we perform the summation in each digit via the C_n operator, the problem reduces to addition of $\log_2 n$ numbers — by the compressor method consuming depth $O(\log\log n)$ it reduces to the usual addition of at most $2n$-bit numbers. From the Table 1 and the result of V. M. Khrapchenko [12] on the asymptotic depth of addition, in particular, we obtain

$$\mathsf{D}_{\mathcal{B}_0}(M_n) \lesssim 4.96\log_2 n, \qquad \mathsf{D}_{\mathcal{B}_2}(M_n) \lesssim 3.98\log_2 n.$$

The presentation is structured as follows. §2 contains information on auxiliary operations and general ways for computing symmetric functions. Formulae for approximate summation are constructed in §3. The method of compressors is discussed in §4. In §5, designs of efficient compressors are proposed. Numerical bounds on the complexity and the depth of symmetric functions are derived in §6. In §7, a proof of a theorem on the depth of partial threshold functions is given. Some open problems are stated in §8.

2 Auxillary information

This section discusses the auxiliary operations involved in computing symmetric functions — they correspond to the bottom layer in Fig. 1.

First, let us recall the standard facts about the depth of arithmetic operations in complete bases. Addition, subtraction, and comparison of two n-digit binary numbers may be performed with depth $O(\log n)$. Moreover, V. M. Khrapchenko [12] established the asymptotics of the adder depth over some boolean bases; in particular, the depth of addition is asymptotically $\log_2 n$ over $\mathcal{B}_0$. A trivial generalization[9] allows one to obtain a bound $O(\log n)$ on the depth of addition of q-ary n-digit numbers as well. As a consequence, summation of m such numbers may be performed with depth $O(\log(mn))$. First, via a depth $O(\log m)$ tree of (3,2)-compressors (which perform the transform $(x_1, x_2, x_3) \to (y_0, y_1)$ according to the rule $x_1 + x_2 + x_3 = qy_1 + y_0$), m summands is replaced by two preserving the sum. Then, the addition of the latter two $(n + \log_q m)$-digit numbers is performed with depth $O(\log(n + \log m))$.

[9] A parallel circuit implementing q-ary carries can be constructed by analogy with, for example, [21].

Multiplication of binary n-digit numbers reduces to addition of n numbers and therefore has depth $O(\log n)$ [11]. Division with remainder of two n-digit numbers reduces to inversion and multiplication and is also implemented with depth $O(\log n)$ [2]. The transition between binary and q-ary notation of an n-digit number in either direction can be performed by the "divide-and-conquer" method of A. Schönhage with depth $O(\log^2 n)$, see [9, Chapter 14].

As a consequence, the number $x \in [0, AB)$ can be reconstructed from the known remainders $x_A = x \bmod A$ and $x_B = x \bmod B$, where A, B are relatively prime n-digit numbers, by the formula

$$x = x_A + A(\tau(x_B - x_A) \bmod B)$$

with depth $O(\log n)$, where $\tau = A^{-1} \bmod B$ (Chinese remainder theorem).

Finally, the number $x \in [c, c + R)$ with a given remainder $r = x \bmod R$ can be determined by the formula

$$x = \left\lfloor \frac{c}{R} \right\rfloor R + ((c \bmod R) > r) \cdot R + r$$

also with depth $O(\log n)$, where n is the digit length of the numbers R, c.

In the statements below we will use the following notations:

– $\sigma = x_1 + \ldots + x_n$ — arithmetic sum of boolean variables;

– $R = \prod_i q_i^{l_i}$;

– $\sigma_{q_i} = \sigma \bmod q_i^{l_i}$;

– $\sigma' : \sigma \in [\sigma' t, \sigma' t + 2t)$ — pointer to the interval in which the sum σ falls.

Given the remainders σ_{q_i} and the number σ', with a suitable choice of parameters l_i, t, it is easy to compute σ.

Claim 1 ([33])**.** *Let $l_i, t \in \mathbb{N}$, q_i be relatively prime numbers and $R \geq 2t$. Then for an arbitrary complete boolean basis B,*

$$\mathsf{D}_B(C_n) \leq \mathsf{D}_B(\{\sigma_{q_i}\}, \sigma') + O(\log^2 \log n), \quad \Phi_B(C_n) \leq 2^{O(\log^2 \log n)} \cdot \Phi_B(\{\sigma_{q_i}\}, \sigma').$$

Proof. It follows from the above material that the necessary manipulations with $O(\log n)$-digit numbers σ', σ_{q_i} (conversion to the binary number system, modular arithmetic operations) to determine the sum σ can be performed with depth $O(\log^2 \log n)$. □

To calculate threshold functions, slightly less information is required. Let σ_-, σ_+ denote partial[10] boolean functions of variables $x_1, \ldots, x_n$, defined for $0 \leq t \leq k \leq n - t$ as

$$\sigma_- = 1 \iff \sigma \geq k, \qquad \sigma_- = 0 \iff \sigma \leq k - 2t,$$
$$\sigma_+ = 1 \iff \sigma \geq k + 2t, \qquad \sigma_+ = 0 \iff \sigma \leq k.$$

[10]Here and below, a partial (partially defined) boolean function is an arbitrary boolean function that has prescribed values on its domain.

Claim 2. *Let $l_i, t \in \mathbb{N}$, q_i be relatively prime numbers, $t \le k \le n/2$ and $R \ge 4t$. Then for any complete boolean basis B,*

$$\mathsf{D}_B(T_n^k) \le \mathsf{D}_B(\{\sigma_{q_i}\}, \sigma_-, \sigma_+) + O(\log^2 \log n),$$
$$\mathsf{\Phi}_B(T_n^k) \le 2^{O(\log^2 \log n)} \cdot \mathsf{\Phi}_B(\{\sigma_{q_i}\}, \sigma_-, \sigma_+).$$

Proof. As above, $\sigma \bmod R$ can be computed with depth $O(\log^2 \log n)$. It is then easily verified that

$$T_n^k(x_1, \ldots, x_n) = \mathrm{Maj}_3\left(\sigma_-,\ \sigma_+,\ (\sigma \in \{k, k+1, \ldots, k+2t\} \bmod R)\right). \quad (2)$$

Indeed, for $\sigma \le k-2t$ and $\sigma \ge k+2t$, the first two arguments of the majority function (2) take the same correct values ($\sigma \ge k$). For $\sigma \in (k-2t, k+2t)$, only one of these arguments is guaranteed to take the correct value, and in this case the third argument has the deciding vote, which takes the value ($\sigma \ge k$) on the interval $(k-2t, k+2t)$. □

Starting from [14], the following statement is used to derive upper bounds for the complexity (and, similarly, depth, see [24]) of symmetric functions.

Claim 3. *Let K be a class of boolean functions of n variables, and let a boolean (m,n)-operator $\xi(x_1, \ldots, x_n) = (\xi_1, \ldots, \xi_m)$ be such that any function $f \in K$ can be represented as $f = \varphi(\xi_1(x_1, \ldots, x_n), \ldots, \xi_m(x_1, \ldots, x_n))$. Then for any complete finite basis B,*

$$\mathsf{\Phi}_B(K) \preceq \sum_{i=1}^{m} 2^i \mathsf{\Phi}_B(\xi_i) + 2^{2m},$$

and in the case $\mathcal{B}_0 \subset B$,

$$\mathsf{D}_B(K) \lesssim \max_{1 \le i \le m} \{\mathsf{D}_B(\xi_i) + i\}.$$

Proof. Over the basis $\mathcal{B}_0$ the function φ as a function of the components of the operator ξ is realized by the method of decomposition by variables (the cascade method), i.e. recursively by formulas

$$\varphi(y_1, \ldots, y_m) = y_1 \cdot \varphi_1(y_2, \ldots, y_m) \vee \overline{y_1} \cdot \varphi_0(y_2, \ldots, y_m),$$

see e.g. [14], [39, §V.2.3]. To generalize to an arbitrary basis B, the trivial relation $\mathsf{\Phi}_B(f) \le L + O(2^D)$ is used, where L and D are the complexity and the depth of some formula F that implements the function f over the basis $\mathcal{B}_0$ (see e.g. [32]).

To prove the bound on depth, we can exploit a standard multivariate decomposition formula. Let $B = \mathcal{B}_0$. We choose the parameter $k \approx \sqrt{m}$ and apply the representation

$$\varphi(y_1, \ldots, y_m) = \bigvee_{\alpha=(\alpha_1,\ldots,\alpha_k)\in\{0,1\}^k} y_1^{\alpha_1} \cdot \ldots \cdot y_k^{\alpha_k} \cdot \varphi_\alpha(y_{k+1}, \ldots, y_m).$$

We can further apply the decomposition in k variables to the subfunctions φ_α, etc. Since elementary conjunctions has depth $\lceil \log_2 k \rceil$, the depth of the resulting formula with respect to the input y_{ks+j}, $1 \le j \le k$, does not exceed

$$(k+1)s + \log_2 k + 1 = ks + j + O(\sqrt{m}). \qquad \square$$

When computing symmetric functions, the parameters are chosen as in the statement 1, only the encoding of the numbers σ_{q_i} for $q_i > 2$ should be first changed to locally binary in order to reduce the overall code length to $(1+o(1)) \log_2 n$. For this, the q-ary number σ_q is divided into blocks of length $p \approx \sqrt{\log n}$, and the value of each block is rewritten in binary notation. As noted above, the depth of the code components corresponding to one block increases by $O(\log^2 \log n)$.

3 Approximate summation

Let $\Psi_n^{k,t}$ denote a partial threshold function of n variables with threshold k and uncertainty interval of radius $t \le k$:

$$\Psi_n^{k,t}(x_1, \ldots, x_n) = 1 \iff \sum x_i \ge k + t,$$
$$\Psi_n^{k,t}(x_1, \ldots, x_n) = 0 \iff \sum x_i \le k - t.$$

Recall that $\mathcal{B}_M = \{\vee, \wedge\}$ denotes the standard monotone boolean basis. The method [36, 3] proves

Theorem 1. *The following inequality holds:*

$$\mathsf{D}_{\mathcal{B}_M}(\Psi_n^{k,t}) \le \log_{\sqrt{5}-1}(k/t) + \log_2(n/k) + 2\log_2 \log \left(C_n^{k-t} + C_n^{k+t}\right) + O(1).$$

For completeness, the proof is given in §7.

Let $n = 2^r \cdot t - 1$. We split the segment $[0, n]$ into 2^r intervals $I_j = [jt,\, (j+1)t)$. Let $X_j = \Psi_n^{jt+(t-1)/2,\,(t+1)/2}(x_1, \ldots, x_n)$ be a partial threshold function with uncertainty interval I_j.

The number $J = (J_{r-1}, \ldots, J_0)$ of the interval, in which the arithmetic sum $\sigma = x_1 + \ldots + x_n$ falls, can be calculated from X_j with an accuracy

of ± 1 as follows. Note that the sequence $\{X_j\}$ is monotone (meaning that $X_{j+1} \geq X_j$ for any j). The operator $J' = (J'_{r-1}, \ldots, J'_0)$ with components

$$J'_i = \bigvee_{p=1}^{2^{r-i-1}} \overline{X_{2^{i+1}p-2^i-1}} \cdot X_{2^{i+1}p-1} \tag{3}$$

finds the position of the first one. In other words, $J' = \min\{j \mid X_j = 1\}$. Since for the true interval number J it is known that $X_{J-1} = 0$ and $X_{J+1} = 1$, we conclude $J' \in \{J,\ J+1\}$.

It is obvious from (3) that the depth of the evaluation of the component J'_i with respect to the inputs X_j is $r-i$ over the basis $\mathcal{B}_0$. Then from Theorem 1 it follows

Corollary 1. *The depth of the operator $J'(x_1, \ldots, x_n)$ satisfies the inequality*

$$\mathsf{D}_{\mathcal{B}_0}(J') \leq (\log_{\sqrt{5}-1} 2 + 1) \log_2(n/t) + 2 \log_2 n + O(1).$$

This result can be applied to the implementation of the operator C_n. However, to derive more precise bounds on the depth and the complexity of the class of symmetric functions S_n, we have to estimate the depth of individual code bits more accurately. Note that the senior bits of J' are not computed too economically by the described scheme. In this case, it would be desirable to increase the width of the intervals.

It is not clear how to obtain the desired bounds directly for the digits of J', but this can be done in a slightly extended encoding. Let $r = ls$. A number in the interval $[0, 2^r - 1]$ can be written in the form (B, H), where $B = [b_{l-1}, \ldots, b_0]$ is a number composed of s-digit blocks b_i, and $H = (h_{l-1}, \ldots, h_0)$ is an l-bit correction vector. Translation into ordinary binary notation may be performed as follows: in a cycle over i from $l-1$ to 0:

$$[b_{l-1}, \ldots, b_i] := [b_{l-1}, \ldots, b_i] - h_i. \tag{4}$$

Let us formally assume $h_l = 0$.

Lemma 1. *For the components of the code of a number that approximates J with an accuracy of ± 1 and is written in the form (B, H),*

$$\mathsf{D}_{\mathcal{B}_0}(b_i, h_{i+1}) \leq (\log_{\sqrt{5}-1} 2 + 1) \log_2(2^{-is} n/t) + 2 \log_2 n + O(1).$$

Proof. We will compute the bits of the approximation of J in blocks of s pieces, starting from the most significant ones[11]. To compute the next

[11] In fact, the blocks are computed independently and in parallel. But the blocks with larger indices become known earlier, at least from the point of view of the depth estimates that we use.

block b_i, we use the scheme described above with a partition into $2^{(l-i)s}$ intervals of width $2^{is} \cdot t$. We calculate $s+2$ least significant bits of the interval number, thus "overlapping" the previous block by two bits. According to Theorem 1, this computation is performed with depth

$$(\log_{\sqrt{5}-1} 2 + 1) \log_2(2^{-is} n/t) + 2 \log_2 n + O(1).$$

It should be taken into account that the result of each computation can be a group of digits of either the true number J or the adjacent number (by the considered digits). In the process of calculations, we will ensure that the following condition is met: the code $([b_{l-1}, \ldots, b_i], h_{l-1}, \ldots, h_i)$, under rule (4), determines the senior $(l-i)s$ digits of either the number J or the number $J + 2^{is}$.

Initially, we set $h_{l-1} = \ldots = h_0 = 0$. After the first step, when the block b_{l-1} is computed by formulas (3), the required condition is obviously satisfied by the property of the operator J'. Let us consider the step at which b_i is computed.

Let the previously calculated block b_{i+1} have the form $[\rho, a]$, and the new group of bits — $[b, b_i]$, where a, b — respectively the least significant and most significant pair of bits, holding the same positions in the code. Let's consider possible cases.

The case $a - b \equiv 2 \bmod 4$ is impossible, since the blocks with index $i+1$ of the numbers J, $J + 2^{is}$, $J + 2^{(i+1)s}$ can take only two adjacent values.

If $a = b$, then $[b_{i+1}, b_i]$ is a correct pair of blocks of the number J or $J + 2^{is}$. Therefore, we set $h_{i+1} = 0$ and proceed to the next step.

If $a \equiv b + 1 \bmod 4$, then b_{i+1} is a block of $J + 2^{(i+1)s}$. Hence, $[b_{i+1} - 1 \bmod 2^s, b_i]$ is a correct pair of blocks of J or $J + 2^{is}$. Thus, we set $h_{i+1} = 1$ and proceed to the next step.

If $b \equiv a + 1 \bmod 4$, then $b_i = [0 \ldots 0]$ is a block of $J + 2^{is}$. Then $[b_{i+1}, b_i - 1 \bmod 2^s]$ is a correct pair of blocks of J. We set $h_{i+1} = 0$ and $b_i = [1 \ldots 1]$, and proceed to the next step.

These additional steps increase the depth of blocks b_i by $O(1)$. □

As a consequence, the number J' is encoded by $r + o(r)$ bits, the i-th of which is calculated with depth $(\log_{\sqrt{5}-1} 2 + 1)i + (2 + o(1)) \log_2 n$.

4 Formulae composed of compressors

An efficient method that allows computing different digits of a sum with different formula complexity is proposed in [24].

Consider formulae composed of compressors. We can apply the well-known potential method to evaluate the quality of the formulae. The digit

in the l-th place, computed at the depth d, is (implicitly) assigned with a potential $\lambda^d\nu^l$, where λ, ν are suitable constants.

Let a family of q-ary compressors $\mathcal{C}_1, \ldots, \mathcal{C}_s$ over a basis B be given, using t types of input and output encoding. Let the inputs $x_{j,\tau,k,i}$ of the compressor $\mathcal{C}_j$ with encoding type $\tau \in \{1, \ldots, t\}$, related to place k, have depths $d^x_{j,\tau,k,i}$, and the outputs $y_{j,\tau,k,i}$ of the same encoding in the same place have depths $d^y_{j,\tau,k,i}$. Let $k_{\max}$ be the maximum place to which the outputs of compressors belong. For a vector $v = (v_1, \ldots, v_s) \in \mathbb{R}^s_{\geq 0}$ we set

$$a_{\tau,k}(\lambda, v) = \sum_{j=1}^{s} v_j \left(\sum_i \lambda^{d^x_{j,\tau,k,i}} - \sum_i \lambda^{d^y_{j,\tau,k,i}} \right), \tag{5}$$

where the sum over an empty set is assumed to be zero. The vector function

$$a(\lambda, v; x) = (a_1, \ldots, a_t), \qquad a_\tau(\lambda, v; x) = \sum_{k=0}^{k_{\max}} a_{\tau,k}(\lambda, v) x^{k_{\max}-k},$$

is called the *characteristic operator* of the family of compressors. The components of the vector v have the meaning of the distribution of compressors in formulae, so we can assume $||v|| = \sum v_i = 1$. In fact, we just consider a composite compressor $\bigcup v_i \mathcal{C}_i$. In the case of a single compressor in the family ($s = 1$), the argument v is omitted.

Let $\mathbf{0}^m$ (or simply $\mathbf{0}$) denote the zero vector of length m. From [24, 27] we deduce

Lemma 2. *Let for some*[12] $\lambda > 1$, $v \geq \mathbf{0}^s$, $||v|| = 1$ *and* $\nu \geq 1$, *we have* $a(\lambda, v; \nu) \geq \mathbf{0}^t$. *Then for any* $\mu \in [0, 1]$,

$$\mathsf{D}_B \left(C^{(q)}_{n,\, \mu \log_q n} \right) \lesssim (\mu \log_q \nu + 1) \log_\lambda n.$$

Proof. **I.** Assuming $\min\{d^x_{j,\tau,k,i}\} = 0$, denote $d_0 = \max\{d^y_{j,\tau,k,i}\}$.

Let a compressor of type $\mathcal{C}_j$ of place l and depth d receive inputs of place $l + k$ at depths $d^x_{j,\tau,k,i} + d$ and produce outputs of place $l + k$ at depths $d^y_{j,\tau,k,i} + d$ for all admissible τ, k, i. Consider a circuit in which for any l, $-r \leq l \leq \log_q n + 1$, at depth d, $0 \leq d \leq \log_\lambda(c\nu^l n)$, there are $\left\lfloor c v_j \nu^l n \lambda^{-d} \right\rfloor$ compressors of type $\mathcal{C}_j$ of place l, where c is a constant to be chosen later. We assume the non-zero inputs of the circuit to be received in the zero place at depth d_0.

[12]The inequality of vectors $v \geq u$ (or $v > u$) means that the inequalities $v_i \geq u_i$ (respectively $v_i > u_i$) are satisfied for each pair of components.

Let us count the number of inputs and outputs of the circuit at a fixed place l, depending on depth d. By construction, all outputs of the circuit at depth less than d_0 are zero. All inputs in negative places are also zero. The total number of inputs at depth less than d_0 (all of them are zero) is $O(n)$.

The difference between the number of inputs and the number of outputs of type τ at arbitrary place $l \geq 0$ and depth d, $d_0 \leq d \leq \log_\lambda(c\nu^l n)$, of compressors within the circuit takes the form

$$\sum_{j,k,i} \left\lfloor cv_j \nu^{l-k} n \lambda^{d^x_{j,\tau,k,i}-d} \right\rfloor - \sum_{j,k,i} \left\lfloor cv_j \nu^{l-k} n \lambda^{d^y_{j,\tau,k,i}-d} \right\rfloor = \\ = c\nu^{l-r} n \lambda^{-d} a_\tau(\lambda, v; \nu) \pm O(1) \geq -O(1).$$

Thus, the circuit at depth d produces at most $O(1)$ outputs of all types in place l (that is, such outputs that are not connected to the inputs of other compressors). At depths $\log_\lambda(c\nu^l n) + O(1)$ the circuit produces $O(1)$ more outputs of place l. Therefore, the circuit produces $O(\log n)$ nontrivial outputs at each place. A suitable choice of the constant c ensures that there are at least n inputs at place zero[13].

Thus, all non-zero outputs of the circuit can be grouped into $O(\log n)$ summands. By construction, all outputs in places not exceeding $l = \mu \log_q n$ have depth no greater than

$$\log_\lambda(c\nu^l n) + O(1) = (\mu \log_q \nu + 1) \log_\lambda n + O(1). \tag{6}$$

II. Further, with depth $O(1)$ one can reduce the encoding of all digits to a single type. The final summation of $O(\log n)$ l-digit q-ary numbers may be performed with depth $O(\log(l \log n))$, see §2. □

The result for the complexity of formulae is obtained similarly. With a digit in the l-th place, expressed by the formula of complexity L, we assign the potential $L^p \nu^l$, where p, ν are suitable constants.

Let, as above, a family of q-ary compressors $\mathcal{C}_1, \ldots, \mathcal{C}_s$ over a basis B be given, using t types of encoding of inputs $x_{j,\tau,k,i}$ and outputs $y_{j,\tau,k,i}$, where j is the compressor number, τ is the encoding type, and k is the place number. By $L(x)$ we denote the size of the formula[14], implementing a digit x, allowing $L(x)$ to take an arbitrary positive real value.

[13] This refers to the total number of inputs, including those of intermediate compressors.

[14] Here and below, in a number of specific applications, we interpret the concept of complexity broadly as a certain measure that behaves according to the rules of formula complexity in order to cover cases of using multi-bit encoding of digits or joint encoding of groups of digits.

For a vector $v = (v_1, \ldots, v_s) \geq \mathbf{0}$, we set

$$A_{\tau,k}(p, v) = \sum_{j=1}^{s} v_j \left(\sum_i L(x_{j,\tau,k,i})^p - \sum_i L(y_{j,\tau,k,i})^p \right). \tag{7}$$

Due to the given context, we assume a continuous, monotone, and also proportional[15] dependence of the complexity $L(y_{j,\tau,k,i})$ of the outputs on the complexity $L(x_{j,\tau,k,i})$ of the inputs.

The characteristic operator of the compressor family is defined as

$$A(p, v; x) = (A_1, \ldots, A_t), \qquad A_\tau(p, v; x) = \sum_{k=0}^{k_{\max}} A_{\tau,k}(p, v) x^{k_{\max}-k},$$

where $k_{\max}$ is the maximum number of the place to which the outputs belong.

In principle, in the case of formula complexity, it is possible to do without introducing the weight vector v, indirectly setting the relationship between the number of compressors of different types by the size of the input formulae.

Also from the works [24, 26, 27] is extracted

Lemma 3. *Let for some $p > 0$, $L(x_{j,\tau,k,i}) > 0$, $v \geq \mathbf{0}^s$, $||v|| = 1$, and $\nu \geq 1$ we have*

$$A(p, v; \nu) > \mathbf{0}^t. \tag{8}$$

Then for any $\mu \in [0, 1]$,

$$\Phi_B \left(C^{(q)}_{n,\, \mu \log_q n} \right) \preceq n^{(\mu \log_q \nu + 1)/p + o(1)}.$$

Proof. The lemma is essentially reduced to the previous one.

Due to the continuous dependence of the complexity of the outputs on the complexity of the inputs, there exists $\delta > 0$ such that inequality (8) remains valid when substituting into (7) the parameters $L^x_{j,\tau,k,i} \in [L(x_{j,\tau,k,i}) - \delta,\, L(x_{j,\tau,k,i})]$ and $L^y_{j,\tau,k,i} \in [L(y_{j,\tau,k,i}),\, L(y_{j,\tau,k,i}) + \delta]$ instead of $L(x_{j,\tau,k,i})$ and $L(y_{j,\tau,k,i})$, respectively.

Then we can find a (sufficiently small) $\lambda > 1$ such that there exist $d^x_{j,\tau,k,i}, d^y_{j,\tau,k,i} \in \mathbb{Z}$ ensuring $\lambda^{d^x_{j,\tau,k,i}/p} \in [L(x_{j,\tau,k,i}) - \delta,\, L(x_{j,\tau,k,i})]$ and $\lambda^{d^y_{j,\tau,k,i}/p} \in [L(y_{j,\tau,k,i}),\, L(y_{j,\tau,k,i}) + \delta]$ for all j, τ, k, i. Let us call the number $d^x_{j,\tau,k,i}$ (respectively $d^y_{j,\tau,k,i}$) *pseudodepth* of the input $x_{j,\tau,k,i}$ (output $y_{j,\tau,k,i}$).

Therefore, under the substitution of $\lambda^{d^x_{j,\tau,k,i}/p} \to L(x_{j,\tau,k,i})$ and $\lambda^{d^y_{j,\tau,k,i}/p} \to L(y_{j,\tau,k,i})$ expression (7) turns into (5), and the conditions of Lemma 2 are

[15] An increase in the size of all inputs by a times leads to an increase in the size of all outputs by a times.

satisfied. Part I of the proof of Lemma 2 allows the interpretation of depth as pseudodepth, so its conclusion is valid. This means that there exists a circuit for summation of n bits, composed of given q-ary compressors, and circuit outputs are grouped into $O(\log n)$ summands. Moreover, all outputs in places no higher than $l = \mu \log_q n$ have a pseudodepth at most (6), which means: they are expressed by formulae of size

$$O\left((\nu^l n)^{1/p}\right) \preceq n^{(\mu \log_q \nu + 1)/p}.$$

The final addition increases the depth of the formula by $o(\log n)$ (part II of the proof of Lemma 2), and therefore the complexity increases by $n^{o(1)}$ times. □

5 Constructions of compressors

5.1 Binary compressors

The main direction of obtaining more efficient formulae for symmetric functions for a long time remained the improvement (i.e. complication) of binary compressor designs and the efficiency of their utilization, culminating in a series of papers [24]–[27]. In this work, we will simply use the best designs known to date. Below, we will list them with an indication of characteristics, omitting the details of the constructions.

Formula depth over the basis $\mathcal{B}_0$. To construct a circuit over the basis $\mathcal{B}_0$, we use a (7,3)-compressor, which calculates the arithmetic sum of seven bits. It was probably proposed in [10], for a description see also [31]. The characteristic polynomial of the compressor is $a(\lambda; x) = (6+\lambda^2-\lambda^6)x^2 - \lambda^7 x - \lambda^6$.

Bounds on the depth of the components of the operator C_n are obtained by Lemma 2 applied with parameters $\lambda \in [\lambda_0, \sqrt{2})$ and $\nu = \frac{\lambda^7+\lambda^3\sqrt{\lambda^8-4\lambda^6+4\lambda^2+24}}{2(6+\lambda^2-\lambda^6)}$, where $\lambda_0 \approx 1.151$ is a root of the polynomial $\lambda^7 + 2\lambda^6 - \lambda^2 - 6$.

Formula depth over the basis $\mathcal{B}_2$. In [27] a family of three $(3,2)$-compressors with three types of input and output encoding was proposed, with the help of which a record bound for the depth of the operator C_n over the basis $\mathcal{B}_2$ was obtained. The characteristic operator of the family has the form $a(\lambda, v; x) = (a_1, a_2, a_3)$, where

$$a_1(\lambda, v; x) = v_1(2 + \lambda - \lambda^2)x + v_2(2 - \lambda^3)x + v_3(\lambda^2 - \lambda^3)x,$$
$$a_2(\lambda, v; x) = -v_1\lambda^2 + v_2\lambda^2 x + v_3(x - \lambda^3),$$
$$a_3(\lambda, v; x) = -v_2\lambda^3 + v_3\lambda x.$$

It is easy to verify [27] that the maximum value $\lambda = \lambda_0$, for which for some $v \geq \mathbf{0}$ the inequation $a(\lambda, v; 1) \geq \mathbf{0}$ holds, is a root of the polynomial $3\lambda^3 + 2\lambda^2 - 2\lambda - 6$ and is approximately 1.221. It can also be verified that for $\lambda_0 \leq \lambda < 2$ the optimization problem $\nu = \min x$ for $a(\lambda, v; x) \geq \mathbf{0}$, $v \geq \mathbf{0}$ and $x \geq 1$ has a solution

$$\nu = b + \sqrt{b^2 + c}, \qquad b = \frac{\lambda^3 + \lambda^2 - \lambda - 4}{2(2 + \lambda - \lambda^2)}, \qquad c = \frac{2\lambda^3}{2 + \lambda - \lambda^2}.$$

Then, the depth of the C_n components may be bounded with the use of Lemma 2.

In [10] (according to [27]) an even more efficient family of (5,3)-compressors was constructed, apparently, but its description is not available to the author.

Formula complexity over the basis $\mathcal{B}_0$. To estimate the complexity of formulae, we use compressors from [32]. For the basis $\mathcal{B}_0$, a $(17, 6)$-compressor was constructed, operating according to the rule

$$x_1 + \ldots + x_{17} = y_1 + y_2 + y_3 + 2y_4 + 4y_5 + 8y_6.$$

The complexity of the outputs y_i is expressed in terms of the complexity of the inputs as

$$\begin{pmatrix} \Phi(y_1) \\ \Phi(y_2) \\ \Phi(y_3) \\ \Phi(y_4) \\ \Phi(y_5) \\ \Phi(y_6) \end{pmatrix} \leq \begin{pmatrix} 12 & 16 & 0 & 0 & 0 \\ 12 & 16 & 0 & 0 & 0 \\ 0 & 0 & 32 & 4 & 16 \\ 96 & 96 & 96 & 12 & 32 \\ 144 & 144 & 96 & 14 & 40 \\ 72 & 72 & 48 & 7 & 20 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{pmatrix},$$

where

$$\Phi_1 = \Phi(x_1) = \ldots = \Phi(x_6), \qquad \Phi_2 = \Phi(x_7) = \ldots = \Phi(x_{10}),$$
$$\Phi_3 = \Phi(x_{11}) = \ldots = \Phi(x_{14}), \qquad \Phi_4 = \Phi(x_{15}), \qquad \Phi_5 = \Phi(x_{16}) = \Phi(x_{17}).$$

Formula complexity over the basis $\mathcal{B}_2$. For the basis $\mathcal{B}_2$ we choose the $(15, 6)$-compressor [32], which generalizes the construction from [27] and operates according to the rule

$$16x_1 + \sum_{i=1}^{3} 2^{4-i}(x_{3i-1} + x_{3i} + x_{3i+1}) + x_{11} + \ldots + x_{15} = \sum_{i=1}^{6} 2^{i-1} y_i.$$

The complexity of the outputs y_i satisfies the relation

$$\begin{pmatrix} \Phi(y_1) \\ \Phi(y_2) \\ \Phi(y_3) \\ \Phi(y_4) \\ \Phi(y_5) \\ \Phi(y_6) \end{pmatrix} \leq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 3 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 6 & 1 & 2 \\ 0 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 6 & 3 & 3 & 6 & 1 & 2 \\ 1 & 2 & 2 & 3 & 3 & 3 & 6 & 3 & 3 & 6 & 3 & 3 & 6 & 1 & 2 \\ 1 & 4 & 4 & 9 & 3 & 3 & 6 & 3 & 3 & 6 & 3 & 3 & 6 & 1 & 2 \end{pmatrix} \cdot \vec{\Phi},$$

where $\vec{\Phi} = (\Phi(x_1), \ldots, \Phi(x_{15}))^T$.

Bounds on the complexity of the components of C_n are obtained by Lemma 3. For any μ, the minimizing ν values of the input sizes are selected. The minimal values of the exponent p for which $\mu \geq 1$ are $p \approx 0.2194$ for the compressor over $\mathcal{B}_0$ and $p \approx 0.3237$ for the compressor over $\mathcal{B}_2$.

5.2 Ternary compressors

Simple constructions of ternary (4,2)-compressors, i.e. minimally possible complete compressors, are proposed by the author in [31, 32]. Here we construct optimized versions of the circuits (except for the case of complexity of formulae over $\mathcal{B}_2$). Over the standard basis, the direction of improvement consists in approaching to known fast methods for summation modulo 3 [19, 5], see also [34].

The ternary (4,2)-compressor computes the arithmetic sum $[U_1, U_0]$ of four numbers $X_1, X_2, X_3, X_4 \in \mathbb{Z}_3$, that is, $3U_1 + U_0 = X_1 + X_2 + X_3 + X_4$.

To write ternary digits we use two types of encoding. In the *standard* representation ternary digit X is encoded by three bits x^0, x^1, x^2, where $x^k = (X = k)$. In the *monotone* encoding a pair of code bits $x^\wedge = x^2$ and $x^\vee = \overline{x^0}$ is used (i.e. digits 0, 1, 2 have codes 00, 01, 11 respectively).

First, we note that in the monotone encoding, the bits $u_j^\wedge$ and $u_j^\vee$ of the sum are expressed in terms of the inputs by dual formulae: a formula for $u_j^\wedge$ turns into a formula for $u_j^\vee$ via substitution $x_k^\wedge \leftrightarrow x_k^\vee$, $\vee \leftrightarrow \wedge$.

This can be verified if 1) we consider x_k°, u_j°, $\circ \in \{\vee, \wedge\}$ as three-valued logic functions of variables X_i; 2) in the table of values of functions x_k°, u_j° replace 1 with 2, and replace functions of the basis $\mathcal{B}_0$ with similar functions defined on vectors of zeros and twos and taking values from $\{0, 2\}$; 3) apply the ternary version of the duality principle [39, §I.1.3], considering constants k and $2 - k$ as dual, $k \in \{0, 1, 2\}$.

Recall that the sum $Z = X + Y \bmod 3$ in the standard encoding may be implemented by simple (essentially dual) formulae [19][16]

$$z^k = x^0 y^k \vee x^1 y^{k-1} \vee x^2 y^{k-2} = (x^0 \vee \overline{y^k})(x^1 \vee \overline{y^{k-1}})(x^2 \vee \overline{y^{k-2}}), \qquad (9)$$

where index operations are performed modulo 3.

Formula complexity over the basis $\mathcal{B}_0$. Consider a pair of compressors that compute the digit U_0 in the standard encoding, and U_1 — in the monotone. The first compressor uses the standard encoding for inputs, and the second — the monotone.

[16] Which are trivially generalized to the case of sums modulo q.

If the inputs are given in the standard encoding, then, computing $U_0 = (X_1+X_2)+(X_3+X_4) \bmod 3$ directly by formulas (9), we obtain the relation[17]

$$\Phi(U_0) \le 9(\Phi(X_1) + \Phi(X_2) + \Phi(X_3) + \Phi(X_4)).$$

For inputs in the monotone encoding, the same formulas are used, only the "missing" bits x^1 are first calculated as $x^\vee \cdot \overline{x^\wedge}$. The complexity is estimated as

$$\Phi(U_0) \le 12(\Phi(X_1) + \Phi(X_2) + \Phi(X_3) + \Phi(X_4)).$$

The most significant digit U_1 of the sum is computed as in [32]. Due to duality, it is sufficient to specify a formula for $u_1^\vee$. We will construct it in the standard way of expression via threshold functions. Denote $\chi_1^t = (X_1 + X_2 \ge t)$ and $\chi_2^t = (X_3 + X_4 \ge t)$ — these are threshold functions of ternary variables expressed by binary codes. Then

$$u_1^\vee = \chi_1^3 \vee \chi_1^2\chi_2^1 \vee \chi_1^1\chi_2^2 \vee \chi_2^3. \tag{10}$$

The functions χ_i^1 and χ_i^2 are implemented by the formulae

$$\chi_i^1 = x_{2i-1}^\vee \vee x_{2i}^\vee, \qquad \chi_i^2 = x_{2i-1}^\vee x_{2i}^\vee \vee x_{2i-1}^\wedge \vee x_{2i}^\wedge,$$

and the functions χ_i^4 and χ_i^3 can be expressed dually in the sense mentioned above. So we obtain (for each of the compressors)

$$\Phi(U_1) \le 5(\Phi(X_1) + \Phi(X_2) + \Phi(X_3) + \Phi(X_4)).$$

The characteristic operator of the pair of compressors with the complexity of all inputs set to 1 has the form

$$A(p, v; x) = (\,(4x - 36^p)v_1x - 48^pv_2x,\ 4v_2x - 20^p\,).$$

The condition $A(p, v; x) > \mathbf{0}$ is equivalent to

$$v_2 < \frac{4 - 36^p}{48^p - 36^p + 4}, \qquad x > \frac{20^p}{4v_2} > \frac{20^p(48^p - 36^p + 4)}{4(4 - 36^p)}$$

and is applied when $p \in (p_0, \log_6 2)$, where $p_0 \approx 0.2056$ is the root of the equation

$$960^p - 720^p + 4(20^p + 36^p) = 16$$

(the condition providing $A(p, v; x) = \mathbf{0}$ and $x = 1$). As a consequence of Lemma 3, for admissible p and any $\varepsilon > 0$, we obtain

$$\Phi_{\mathcal{B}_0}\left(C^{(3)}_{n,\,\mu\log_3 n}\right) \preceq n^{(\mu\log_3(\nu+\varepsilon)+1)/p+o(1)}, \qquad \nu = \frac{20^p(48^p - 36^p + 4)}{4(4 - 36^p)}.$$

[17] By complexity of a ternary digit we mean the complexity of each code bit.

Note that in this case, all bounds are achieved on a single naturally constructed formula, which can be analyzed without resorting to Lemma 3. The same note applies to all (4,2)-compressors, which will be discussed below.

Formula depth over the basis $\mathcal{B}_0$. We say that an intermediate ternary digit U, computed by a compressor circuit, has $*$-depth d, if for each bit u of the code of the digit U there are formulas $\Phi_0^\vee, \Phi_1^\vee, \Phi_0^\wedge, \Phi_1^\wedge$, such that $u = \Phi_0^\vee \vee \Phi_1^\vee = \Phi_0^\wedge \cdot \Phi_1^\wedge$, while in the standard encoding $\mathsf{D}(\Phi_0^\star) \le d$ and $\mathsf{D}(\Phi_1^\star) \le d-1$, and in the monotone encoding $\mathsf{D}(\Phi_0^\star) \le d$ and $\mathsf{D}(\Phi_1^\star) \le d-2$, where $\star \in \{\vee, \wedge\}$.

In the circuit (actually in the family of circuits) that we are constructing, the inputs X_1 and X_3 have $*$-depth 0, and the inputs X_2 and X_4 have $*$-depth 2; any input can be specified in any encoding. Let us check that in this case the sum bit U_0 is computed with $*$-depth 7 in the standard encoding, and the bit U_1 — with $*$-depth 6 in the monotone encoding.

1. Applying either of two formulas (9), it is easy to check that the sums $E_1 = X_1 + X_2 \bmod 3$ and $E_2 = X_3 + X_4 \bmod 3$ may be computed with depth 5 in any input encoding. The output encoding is standard. If the input X is given in the monotone encoding, then the missing bit x^1 should be pre-computed as $x^\vee \cdot \overline{x^\wedge}$. Therefore, $U_0 = E_1 + E_2 \bmod 3$ is computed with *-depth 7 by formulas (9).

2. To calculate U_1 we use only the monotone part of the input codes. Due to duality (see above) it is sufficient to construct formulas of the type $\Phi_0^\vee \vee \Phi_1^\vee$ only.

The equality $U_1 = 2$ is satisfied when the total sum is not less than 6. This means that among the terms X_i there are either three twos, or there are two twos and no zeros. The first condition is expressed as

$$A_1 \vee A_2, \qquad A_1 = x_1^\wedge x_2^\wedge (x_3^\wedge \vee x_4^\wedge), \qquad A_2 = x_3^\wedge x_4^\wedge (x_1^\wedge \vee x_2^\wedge).$$

The second condition is written as

$$A_3 \cdot A_4, \qquad A_3 = (x_1^\wedge \vee x_2^\wedge)(x_3^\wedge \vee x_4^\wedge) \vee x_1^\wedge x_2^\wedge \vee x_3^\wedge x_4^\wedge, \qquad A_4 = x_1^\vee x_2^\vee x_3^\vee x_4^\vee.$$

Note that the expressions $x_1^i \star x_2^i$ and $x_3^i \star x_4^i$, where $\star \in \{\vee, \wedge\}$, are evaluated with depth 3 regardless of the input encoding type. Therefore, $\mathsf{D}(A_1) = \mathsf{D}(A_2) = \mathsf{D}(A_4) = 4$ and $\mathsf{D}(A_3) = 5$. Finally, the most significant bit U_1 of the code may be computed as

$$u_1^\wedge = A_1 \vee A_2 \vee A_3 \cdot A_4 = \Phi_0^\vee \vee \Phi_1^\vee, \qquad \Phi_0^\vee = A_3(A_1 \vee A_4), \quad \Phi_1^\vee = A_2, \tag{11}$$

where $\mathsf{D}(\Phi_0^\vee) = 6$ and $\mathsf{D}(\Phi_1^\vee) = 4$.

Symmetrically, $U_1 = 0$ iff there are three zeros among the inputs, or there are two zeros and no twos. Expressing these conditions leads to formulas for $u_1^\vee$ of form (11) up to substitutions $x_i^\wedge \leftrightarrow \overline{x_i^\vee}$.

The characteristic polynomial of the constructed circuit is $(2+2\lambda^2-\lambda^7)x-\lambda^6$. Lemma 2 is applied with parameters $\lambda \in [\lambda_0, \lambda_1)$ and $\nu = \frac{\lambda^6}{2+2\lambda^2-\lambda^7}$, where $\lambda_0 \approx 1.1356$ and $\lambda_1 \approx 1.2657$.

Note that in the proposed design, the computation of U_0 just slightly reproduces the fast algorithm of addition modulo 3 from [5] (only in the case when all inputs are given in the standard encoding) and does not in any way imply the even faster algorithm [34]. This leaves some room for optimization of the formula, which can be potentially used when switching to compressors with a larger number of inputs.

Formula depth over the basis $\mathcal{B}_2$. For computations over the basis $\mathcal{B}_2$ we use the standard encoding. The sum $Z = X + Y \bmod 3$ is computed by simple formulas [20]:

$$z^k = \left(x^k \sim y^0\right)\left(x^{k+1} \sim y^2\right) = \left(x^k \vee y^2\right) \sim \left(x^{k+1} \vee y^0\right), \tag{12}$$

where the symbol "$\sim$" denotes the boolean equivalence operation, operations with indices are performed modulo 3.

Now the sum $[U_1, U_0] = X_1 + X_2 + X_3 + X_4$ can be expressed through the intermediate values $Z_1 = X_1 + X_2 \bmod 3$ and $Z_2 = X_3 + X_4 \bmod 3$ as

$$u_0^k = \left(z_1^k \sim z_2^0\right)\left(z_1^{k+1} \sim z_2^2\right), \tag{13}$$

$$u_1^1 = u_0^1 \oplus x_1^1 \oplus x_2^1 \oplus x_3^1 \oplus x_4^1, \qquad u_1^0 = A_3 \star A_2 A_1 = (A_3 \star A_1)A_2, \tag{14}$$

$$A_1 = \overline{(x_1^1 \oplus x_2^1 \oplus x_3^1 \oplus x_4^1)} \cdot \overline{(x_1^2 \vee x_2^2 \vee x_3^2 \vee x_4^2)},$$

$$A_2 = T_4^2(x_1^0, x_2^0, x_3^0, x_4^0) = (x_1^0 \vee x_2^0)(x_3^0 \vee x_4^0) \vee x_1^0 x_2^0 \vee x_3^0 x_4^0,$$

$$A_3 = E_4^3(x_1^0, x_2^0, x_3^0, x_4^0) = x_1^0 x_2^0 (x_3^0 \oplus x_4^0) \oplus (x_1^0 \oplus x_2^0) x_3^0 x_4^0,$$

where any of the signs $\vee, \oplus$ can be substituted for $\star$; E_n^m denotes the elementary symmetric function: $E_n^m(x_1, \ldots, x_n) = (\sum x_i = m)$. The formulas for u_1^2 may be obtained from the formulas for u_1^0 by replacements $x_i^k \leftrightarrow x_i^{2-k}$.

The formula for u_1^1 is verified as follows: $U_1 = 1$ iff either $U_0 = 1$ and there is an even number of ones among the inputs X_i (then the sum is even and equal to 4), or $U_0 \neq 1$ and there is an odd number of ones among the inputs X_i (the sum is odd and equal to 3 or 5). The formula for u_1^0 is explained as follows: $U_1 = 0$ if either exactly three inputs are zeros, or there are at least two zeros, an even number of ones, and no twos among them.

If the depth of all inputs is 0, then the outputs U_0 and U_1 are computed by formulas (13) and (14) with depths of 4 and 5, respectively. These estimates

were used in [31]. A slightly better result can be obtained by exploiting the fact that formulas (14) are not balanced in depth.

Let us say that the intermediate ternary digit U, computed by a compressor circuit, has $*$-depth d, if for the code bit u^1 there are formulae $\Phi_0^{\oplus}, \Phi_1^{\oplus}$, such that $u = \Phi_0^{\oplus} \oplus \Phi_1^{\oplus}$, and for each bit $u \in \{u^0, u^2\}$ there are formulae $\Phi_0^{\vee}, \Phi_1^{\vee}, \Phi_0^{\wedge}, \Phi_1^{\wedge}$, such that

$$u = \Phi_0^{\vee} \vee \Phi_1^{\vee} = \Phi_0^{\vee} \oplus \Phi_1^{\vee} = \Phi_0^{\wedge} \cdot \Phi_1^{\wedge},$$

while $\mathsf{D}(\Phi_0^{\star}) \leq d$ and $\mathsf{D}(\Phi_1^{\star}) \leq d-1$, where $\star \in \{\vee, \wedge, \oplus\}$.

Consider a pair of compressors with two encoding types: in the second, the bits are encoded by paired formulas, as defined above. The first compressor receives inputs X_i at depth 0 and outputs U_0 at depth 4, and U_1 — at $*$-depth 4. The second compressor receives inputs X_1 and X_3 at depth 0, and inputs X_2 and X_4 (in a different encoding) — at $*$-depth 1. It outputs U_0 at depth 5, and U_1 — at $*$-depth 5. Formulas (12), (13), (14) demonstrate the validity of these estimates.

The characteristic operator of the pair of compressors has the form $a(\lambda, v; x) = (a_1, a_2)$, where

$$a_1(\lambda, v; x) = v_1(4 - \lambda^4)x + v_2(2 - \lambda^5)x,$$
$$a_2(\lambda, v; x) = -v_1\lambda^4 + v_2(2\lambda x - \lambda^5).$$

It can be verified that the maximum value $\lambda = \lambda_0$ such that $a(\lambda, v; 1) \geq \mathbf{0}$ holds for some $v \geq \mathbf{0}$ is a root of the polynomial $3\lambda^4 - \lambda^3 - 4$ and is approximately equal to 1.1687. It can also be verified that, under the condition $\lambda_0 \leq \lambda < \sqrt{2}$, the optimization problem $\nu = \min x$ for $a(\lambda, v; x) \geq \mathbf{0}$, $v \geq \mathbf{0}$ and $x \geq 1$ has a solution $\nu = \frac{2\lambda^4 - \lambda^3}{4 - \lambda^4}$. Then, depth estimates may be obtained via Lemma 2.

Formula complexity over the basis $\mathcal{B}_2$. To derive the complexity bounds, we apply the compressor from [32] without modifications. The least significant bit U_0 is computed by formulas (12), (13), and the most significant bit U_1 is computed by formulas (10) for bits u_1^0, u_1^2 and by formula (14) for bit u_1^1. Thus, we obtain

$$\Phi(U_0) \leq 4(\Phi(X_1) + \Phi(X_2) + \Phi(X_3) + \Phi(X_4)),$$
$$\Phi(U_1) \leq 5(\Phi(X_1) + \Phi(X_2) + \Phi(X_3) + \Phi(X_4)).$$

The characteristic function of the compressor is $A(p; x) = (4 - 16^p)x - 20^p$. Lemma 2 is applied with parameters $p \in (p_0, 1/2)$ and $\nu = \frac{20^p}{4 - 16^p}$, where $p_0 \approx 0.2402$ is the solution of the equation $16^p + 20^p = 4$.

5.3 Compressors in number systems with other bases

In this section, we present simple designs of suitable q-ary compressors in order to demonstrate their effect on the depth and the complexity of constructed formulae. Since formula synthesis methods use only the least significant digits of the sums computed by q-ary formulae, the only thing that matters for the efficiency of the compressor is the economical implementation of the sum modulo q in the least significant place.

Formula complexity over the basis $\mathcal{B}_0$. In the basis $\mathcal{B}_0$ we use quinary compressors. A quinary digit U is standardly encoded by five bits $u^0, \ldots, u^4$, where $u^k = (U = k)$. Acceptable results are obtained already using simple formulas (9) (generalized to the case of arbitrary q).

An improved summation method is proposed in [34]. It involves an extended encoding $\{u^S | S \subset \mathbb{Z}_q\}$, where $u^S = (U \in S)$. For $Z = X + Y \bmod q$, a simple generalization of (9) holds:

$$z^S = \bigvee_{i=0}^{q-1} x^i y^{S-i} = \bigwedge_{i=0}^{q-1} \left(\overline{x^i} \vee y^{S-i} \right), \tag{15}$$

where $S - i = \{s - i \bmod q \mid s \in S\}$. However, for some sets S, shorter formulae can be produced. In particular, in the case $q = 5$ and $|S| = 4$ we have [34]

$$z^S = \bigvee_{i=0}^{3} x^{A_i} y^{B_i}, \qquad |A_i|, |B_i| \in \{2, 3\}. \tag{16}$$

Fig. 2 illustrates formula (16) for the case $S = \{0, 1, 2, 3\}$. A circulant matrix whose rows and columns are numbered by the values of the variables X and Y, with ones placed exactly at the positions (X, Y) for which $X + Y \in S$, is covered by four rectangles (all-ones submatrices) $A_i \times B_i$.

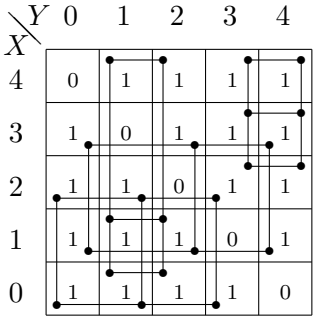

Figure 2: Explanation of formula (16)

In the extended encoding, the quinary sum of two numbers $[w, Z] = X + Y$, where w is the carry to the next digit, is calculated according to the following rules. Bits z^S are implemented by formulas (15) in the case $2 \le |S| \le 3$ and by formulas (16) — in the case $|S| \in \{1, 4\}$ (for $|S| = 1$, negated formulas are applied). The carry can be computed following the formula

$$\overline{w} = x^0 \vee y^0 \vee x^1\overline{y^4} \vee \overline{x^4}y^1 \vee x^2y^2. \tag{17}$$

Let us construct an (incomplete) quinary (4,2)-compressor. The sum $[U, V] = X_1 + X_2 + X_3 + X_4$ is computed by a tree of three additions in the least significant place, and using the carries w_1, w_2, w_3 (17) that appear in the process, we determine the code of the digit U according to the rules

$$\begin{gathered} u^2 = \overline{w_3} \cdot w_1 \cdot w_2 \vee w_3(w_1 \oplus w_2), \quad u^3 = w_1 \cdot w_2 \cdot w_3, \quad u^4 = 0, \\ u^{2,3} = w_3(w_1 \vee w_2) \vee w_1w_2, \quad u^{1,3} = w_1 \oplus w_2 \oplus w_3, \quad u^{0,3} = u^0 \vee u^3. \end{gathered} \tag{18}$$

The missing bits of the code are expressed in a dual way with respect to the written ones[18]. Instead of linear operations, one should substitute formulas implementing them over the basis $\mathcal{B}_0$.

By $l_1(X)$ and $l_2(X)$ we denote the maximal complexity of bits x^S of the code of the number X for $|S| \in \{1, 4\}$ and, respectively, for $|S| \in \{2, 3\}$. Setting $l_i(X_j) = a_i$ for all j, by (15)–(18) we establish

$$\begin{gathered} l_1(X_1 + X_2), l_1(X_3 + X_4) \le 8a_2, \quad l_2(X_1 + X_2), l_2(X_3 + X_4) \le 5(a_1 + a_2), \\ \Phi(w_1), \Phi(w_2) \le 8a_1, \\ l_1(V) \le 40(a_1 + a_2), \qquad l_2(V) \le 25a_1 + 65a_2, \qquad \Phi(w_3) \le 64a_2, \\ l_1(U) \le 2\Phi(w_3) + 3\Phi(w_1) + 3\Phi(w_2) \le 48a_1 + 128a_2, \\ l_2(U) \le 2\Phi(w_3) + 4\Phi(w_1) + 4\Phi(w_2) \le 64a_1 + 128a_2. \end{gathered}$$

Let us define the mixed complexity $L(X) = \max\{l_1(X), l_2(X)/\beta\}$, where $\beta = \frac{5+\sqrt{185}}{16}$ (the optimal value of the parameter for the modulo addition operation [34]). Finally, setting $L(X_1) = \ldots = L(X_4) = 1$ we obtain

$$L(V) \le 40(1 + \beta) < 87, \qquad L(U) \le (48 + 128\beta) < 197. \tag{19}$$

The characteristic function of the compressor is $(4 - 87^p)x - 197^p$. Lemma 2 is applied with parameters $p \in (p_0, \log_{87} 4)$ and $\nu = \frac{197^p}{4-87^p}$, where $p_0 \approx 0.1419$ is the solution of the equation $197^p + 87^p = 4$.

[18] I.e. $u^{\mathbb{Z}_5 \setminus S} = \overline{u^S}$ and for $S \subset \{0, 1, 2, 3\}$: if $u^S = f(w_1, w_2, w_3)$, then $u^{3-S} = f(\overline{w_1}, \overline{w_2}, \overline{w_3})$.

Formula depth over the basis $\mathcal{B}_0$. We are going to describe a (5,2)-compressor, which is obtained from the depth-efficient compressor [34] for computing the sum modulo 5 by adding a formula computing the most significant digit. A quinary digit X, as above, is encoded by a set of bits x^S. Let X have $*$-depth d if each code bit is expressed as both a conjunction and a disjunction of formulae of depth d and $d-2$. The $*$-depth functional is denoted by D^*.

The compressor calculates the sum $[U,V] = X_1 + \ldots + X_5$, where the inputs $X_1, \ldots, X_5$ have $*$-depth 0, 1, 0, 1, 6 respectively. The order of computations is:

$$[w_1, Z_1] = X_1 + X_2, \qquad [w_2, Z_2] = X_3 + X_4, \qquad [w_3, Z_3] = Z_1 + Z_2,$$
$$[w_4, V] = Z_3 + X_5, \qquad U_0 = w_1 + w_2 + w_3, \qquad U = U_0 + w_4.$$

It is easy to check the bounds for depth:

$$\mathsf{D}(z_1^r), \mathsf{D}(z_2^r) \overset{(16)}{\leq} 4, \ \ r \in \mathbb{Z}_5, \qquad \mathsf{D}^*(z_3^r) \overset{(15)}{\leq} 7, \ \ r \in \mathbb{Z}_5, \qquad \mathsf{D}^*(V) \overset{(15)}{\leq} 10,$$
$$\mathsf{D}(w_1), \mathsf{D}(w_2) \overset{(17)}{\leq} 4, \qquad \mathsf{D}(w_3) \overset{(17)}{\leq} 7, \qquad \mathsf{D}(w_4) \overset{(17)}{\leq} 10, \qquad \mathsf{D}(U_0) \overset{(18)}{\leq} 9.$$

Finally, in view of $u^S = \overline{w_4} u_0^S \vee w_4 u_0^{(S \setminus \{0\})-1}$, we obtain $\mathsf{D}^*(U) \leq \mathsf{D}(U) \leq 12$.

The compressor has characteristic polynomial $(2 + 2\lambda + \lambda^6 - \lambda^{10})x - \lambda^{12}$. Lemma 2 is applied with parameters $\lambda \in [\lambda_0, \lambda_1)$ and $\nu = \frac{\lambda^{12}}{2+2\lambda+\lambda^6-\lambda^{10}}$, where $\lambda_0 \approx 1.105$ and $\lambda_1 \approx 1.2299$.

Formula complexity over the basis $\mathcal{B}_2$. For the basis $\mathcal{B}_2$ we construct septenary compressors. For the code bits of the digit U we keep the notation $u^S = (U \in S)$, where $S \subset \mathbb{Z}_7$.

Economical formulae for summation modulo 7 are proposed in [17]. Let $T = \{0,1,2,5\} \subset \mathbb{Z}_7$. It can be verified that for $Z = X + Y$,

$$z^{T+r} = \bigoplus_{k \in \{0,1,3\}} x^{T+k-3r} \cdot y^{T+k-3r}, \tag{20}$$

where index operations are performed modulo 7. The set $\{x^{T+r} \mid r \in \mathbb{Z}_7\}$ constitutes the code of the digit X.

Let's denote $T' = \mathbb{Z}_7 \setminus T = \{3,4,6\}$. The carry w to the next digit can be calculated by the formula

$$w = x^{3,4,6} y^{3,4,6} \oplus x^3 y^3 \oplus x^1 y^6 \oplus x^6 y^1 \oplus \left(x^{5,6} y^{2,3,4,5,6} \vee x^{2,3,4,5,6} y^{5,6}\right) =$$
$$x^{T'} y^{T'} \left(x^{T+4} \vee y^{T+4}\right) \oplus x^{T'+2} y^{T'+2} \left(x^{T'+4} y^{T'} \oplus x^{T'} y^{T'+4}\right) \oplus$$
$$\oplus \left(x^{T'+2} x^{T+4} (y^{T'} \vee y^{T+4}) \vee (x^{T'} \vee x^{T+4}) y^{T'+2} y^{T+4}\right). \tag{21}$$

Now we describe an (incomplete) septenary (4,2)-compressor. As above, the sum $[U,V] = X_1+X_2+X_3+X_4$ is implemented by a tree of three additions in the least significant digit, and with the help of the carries w_1, w_2, w_3 (21) that appear in the process, the code of the digit U is determined by the rules[19]

$$u^{T'} = u^3 = w_1 \cdot w_2 \cdot w_3, \quad u^{T+1} = u^{1,2,3} = w_1 \vee w_2 \vee w_3,$$
$$u^{T'+2} = u^1 = \overline{w_3}(\overline{w_1} \vee \overline{w_2}) \oplus \overline{w_1} \cdot \overline{w_2}, \quad u^{T+3} = u^{1,3} = w_1 \oplus w_2 \oplus w_3,$$
$$u^{T+4} = u^2 = w_3(w_1 \vee w_2) \oplus w_1 w_2, \quad u^{T+5} = u^{0,3} = \overline{w_3}(\overline{w_1} \sim \overline{w_2}) \oplus w_1 w_2,$$
$$u^{T'+6} = u^{2,3} = w_3(w_1 \vee w_2) \oplus w_1 w_2. \quad (22)$$

Denoting by $L(X)$ the maximal complexity of the code bits of a number X, for $L(X_1) = \ldots = L(X_4) = 1$ by (20)–(22) we obtain

$$L(X_1 + X_2), L(X_3 + X_4) \le 6, \quad \Phi(w_1), \Phi(w_2) \le 18, \quad \Phi(w_3) \le 108,$$
$$L(V) \le 36, \qquad L(U) \le \Phi(w_3) + 2\Phi(w_1) + 2\Phi(w_2) \le 180.$$

The characteristic function of the compressor is $(4 - 36^p)x - 180^p$. Lemma 2 is applied with parameters $p \in (p_0, \log_6 2)$ and $\nu = \frac{180^p}{4-36^p}$, where $p_0 \approx 0.1562$ is the solution of the equation $180^p + 36^p = 4$.

6 Results

To derive specific numerical bounds on the indices of complexity and depth of formulae for symmetric functions, we apply the claims from §2. Approximate summation is performed by method §3, where Corollary 1 is used to implement the operator C_n, Theorem 1 — to compute threshold functions, Lemma 1 — to compute general symmetric functions. Remainders σ_{q_i} are computed by the method of compressors §4 (Lemmas 2, 3) exploiting constructions from §5. In all cases, binary and ternary compressors are employed, and additionally — quinary compressors over the basis $\mathcal{B}_0$ and septenary — to estimate the complexity of formulae[20] in the basis $\mathcal{B}_2$.

The results for the complexity and the depth of the operator C_n and the class S_n are given above in Table 1. The new bounds are

$$\Phi_{\mathcal{B}_0}(C_n) \preceq n^{3.77}, \qquad \mathsf{D}_{\mathcal{B}_0}(C_n) \lesssim 3.96 \log_2 n,$$
$$\Phi_{\mathcal{B}_2}(C_n) \preceq n^{2.82}, \qquad \mathsf{D}_{\mathcal{B}_2}(C_n) \lesssim 2.98 \log_2 n,$$
$$\Phi_{\mathcal{B}_0}(S_n) \preceq n^{3.77}, \qquad \mathsf{D}_{\mathcal{B}_0}(S_n) \lesssim 3.96 \log_2 n,$$
$$\Phi_{\mathcal{B}_2}(S_n) \preceq n^{2.85}, \qquad \mathsf{D}_{\mathcal{B}_2}(S_n) \lesssim 2.98 \log_2 n.$$

[19] The complexity of formulas for u^{T+r} and $u^{T'+r}$ is the same: $u^{T'+r} = \overline{u^{T+r}}$.

[20] The effect of septenary compressors in other cases is assumed to be negligible.

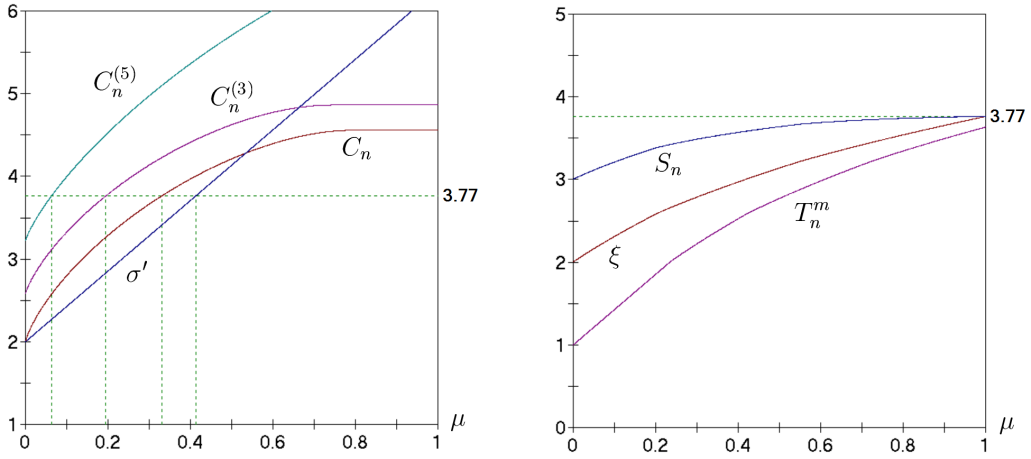

Figure 3: Graphs of complexity indices of formulae over $\mathcal{B}_0$

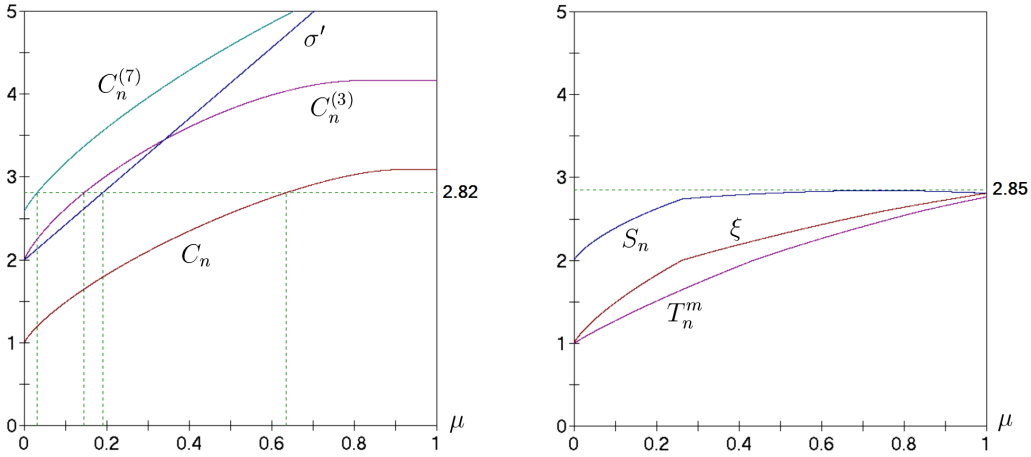

Figure 4: Graphs of complexity indices of formulae over $\mathcal{B}_2$

Their derivation is illustrated by the graphs in Fig. 3–6. Let us comment, for example, the complexity bounds for formulae over the basis $\mathcal{B}_0$, see Fig. 3. The graphs on the left side show indices of complexity of digits $C_{n,\mu\log_2 n}$ (graph C_n), $C^{(3)}_{n,\mu\log_3 n}$ (graph $C_n^{(3)}$), $C^{(5)}_{n,\mu\log_5 n}$ (graph $C_n^{(5)}$) obtained via Lemma 3, as well as the complexity of the operator J' for $t = n^{1-\mu}$ provided by Corollary 1 or, which is the same, the complexity of bits of the approximation σ', starting from the highest ones, established in Lemma 1 (graph σ'). On the right side of the figure, the graph ξ shows the complexity indices of the symmetric function code bits, sorted in ascending order: in this case, the code includes about $0.331\log_2 n$ low-order components of C_n, $(0.195 + o(1))\log_3 n$ and $(0.063 + o(1))\log_5 n$ low-order components of $C_n^{(3)}$ and $C_n^{(5)}$ in locally binary encoding (as explained in §2), respectively, and $(0.413 + o(1))\log_2 n$ high-order bits of the approximate value σ'. The graph S_n is obtained by adding the graph of $y = 1 - \mu$ to the graph ξ. The maximum of this graph provides a bound on the complexity of the class of symmetric functions according to Claim 3.

Additionally, the figures show graphs T_n^m of complexity and depth indices

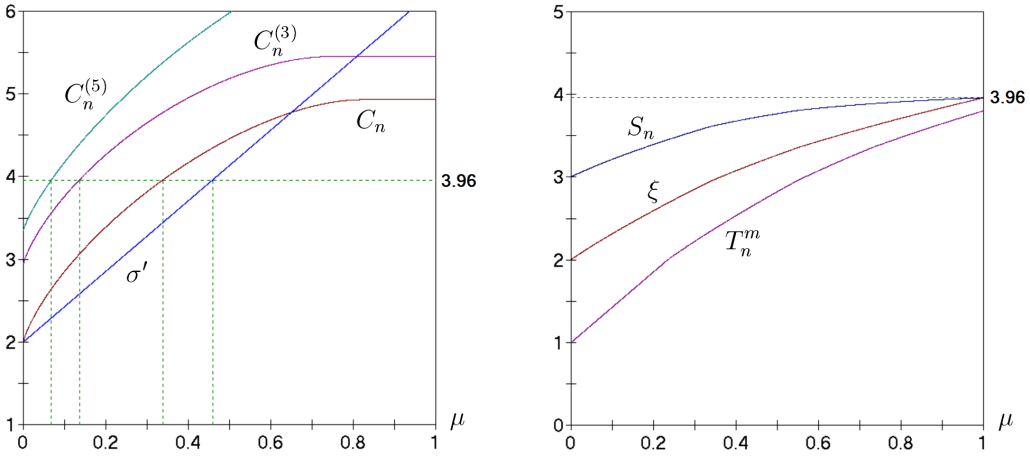

Figure 5: Graphs of depth indices of formulae over $\mathcal{B}_0$

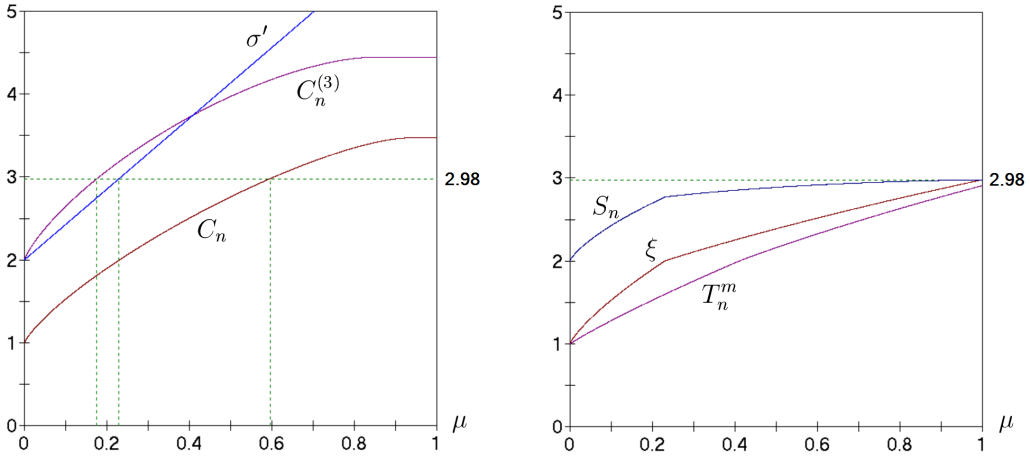

Figure 6: Graphs of depth indices of formulae over $\mathcal{B}_2$

of threshold functions T_n^m, $m = n^\mu$, obtained via Claim 2. Upper bounds for some threshold symmetric functions, including the majority function, are given separately in Table 3.

All calculations were performed with the computer support. However, the estimates (in the rounded form in which they are given) can also be derived analytically — examples of such a derivation can be found in [31, 32].

7 Proof of Theorem 1

Proof of Theorem 1. Recall that we are proving the inequality

$$\mathsf{D}_{\mathcal{B}_M}(\Psi_n^{k,t}) \leq \log_{\sqrt{5}-1}(k/t) + \log_2(n/k) + 2\log_2\log\left(C_n^{k-t} + C_n^{k+t}\right) + O(1).$$

W.l.o.g. assume $2k \leq n$. Let $m = \lfloor n/k \rfloor$. Denote $\alpha = \frac{3-\sqrt{5}}{2} \approx 0.38$.

We will define a series of probability distributions $\Delta_0, \Delta_1, \ldots$ on special sets of monotone formulae. The distribution Δ_d involves formulae of depth $\lceil \log_2 m \rceil + 2d$. The distribution Δ_0 includes formulas of the form $x_{i_1} \vee \ldots \vee x_{i_m}$,

m	$\Phi_{\mathcal{B}_0}$	$\Phi_{\mathcal{B}_2}$	$\mathsf{D}_{\mathcal{B}_0}$	$\mathsf{D}_{\mathcal{B}_2}$
$n^{0.1}$	1.43	1.28	1.43	1.28
$n^{0.2}$	1.86	1.51	1.86	1.53
$n^{0.3}$	2.23	1.73	2.23	1.76
$n^{0.4}$	2.53	1.94	2.54	1.98
$n^{0.5}$	2.78	2.11	2.83	2.17
$n^{0.6}$	2.99	2.27	3.08	2.33
$n^{0.7}$	3.18	2.41	3.30	2.49
$n^{0.8}$	3.35	2.55	3.49	2.64
$n^{0.9}$	3.50	2.67	3.65	2.78
$n/2$	3.64	2.77	3.81	2.91

Table 3: Upper bounds for indices of complexity and depth of functions T_n^m

where each index i_j is chosen equiprobably from the set $\{1, \ldots, n\}$, as well as constant 0, and

$$\tau = \mathbf{P}(F \equiv 0 | F \in \Delta_0) = 1 - \frac{\alpha}{1 - \left(1 - \frac{k}{n}\right)^m}.$$

When $d \geq 1$, the distribution Δ_d includes all possible formulae of the form $(G_1 \vee G_2)(G_3 \vee G_4)$, where the subformulas G_i are chosen from Δ_{d-1} independently.

Consider arbitrary vectors of variables X_0, X_1 with weights $k - t$ and $k + t$, respectively. Let us introduce notations for the probability of formula error:

$$p_d = \mathbf{P}(G(X_0) = 1 \mid G \in \Delta_d), \qquad q_d = \mathbf{P}(G(X_1) = 0 \mid G \in \Delta_d).$$

By construction,

$$p_{d+1} = (1 - (1 - p_d)^2)^2, \qquad q_{d+1} = 1 - (1 - q_d^2)^2. \tag{23}$$

It is easy to verify that the sequences $\{p_d\}$ and $\{q_d\}$ monotonically decrease on the intervals $(0, \alpha)$ and $(0, 1 - \alpha)$, respectively. The ends of the intervals are the roots of equations $x = (1 - (1 - x)^2)^2$ and $x = 1 - (1 - x^2)^2$.

Lemma 4. *For some constant $\gamma > 0$ and some $s \leq \log_{4\alpha}(k/t) + O(1)$,*

$$p_s < \alpha - \gamma, \qquad q_s < 1 - \alpha - \gamma.$$

Note that a similar statement in [36] is proved more simply, but in a weakened formulation: with the base $\beta < 4\alpha$ of the logarithm in the bound for s. A more precise estimate requires a more careful argument.

Proof. **1.** For $d = 0$ the error probabilities are estimated as

$$p_0 = (1-\tau)\left(1-\left(1-\frac{k-t}{n}\right)^m\right) = \alpha\cdot\frac{1-\left(1-\frac{k-t}{n}\right)^m}{1-\left(1-\frac{k}{n}\right)^m},$$
$$q_0 = \tau+(1-\tau)\left(1-\frac{k+t}{n}\right)^m = 1-\alpha\cdot\frac{1-\left(1-\frac{k+t}{n}\right)^m}{1-\left(1-\frac{k}{n}\right)^m}$$

due to the inequality $(1-1/x)^x \le \frac{1}{e} \le (1-1/x)^{x-1}$, which is valid for $x \ge 1$. Denoting $a = 1-\frac{k}{n}$ and $b = \frac{t}{n}$, we derive

$$(a+b)^m a^{-m} = (1+b/a)^m \ge 1+\frac{mb}{a} = 1+\Theta(t/k)$$
$$(a-b)^m a^{-m} = (1-b/a)^m \le 1-\frac{mb}{3a} = 1+\Theta(t/k)$$

due to the simple relations $(1+x)^y \ge 1+xy$, $(1-x)^y \le 1-xy/3$, valid for $x, y \ge 0$ and $xy \le 2$. Consequently,

$$p_0 = \alpha-\varepsilon_0, \qquad q_0 = 1-\alpha-\delta_0,$$

where $\varepsilon_0, \delta_0 \in \Theta(t/k)$.

2. Let us show that for any $s \le \log_{4\alpha}\frac{\alpha}{8\max(\varepsilon_0,\delta_0)}$,

$$p_s < \alpha-(4\alpha)^s\varepsilon_0/2, \qquad q_s < 1-\alpha-(4\alpha)^s\delta_0. \tag{24}$$

This will finish the proof of the lemma.

2.1. Assuming $p_d = \alpha-\varepsilon_d$, and taking into account $(1-\alpha)^2 = \alpha$ we deduce from (23) that

$$p_{d+1} = (1-(1-\alpha+\varepsilon_d)^2)^2 = \alpha-4\alpha\varepsilon_d+\left((2(1-\alpha)+\varepsilon_d)^2-2(1-\alpha)\right)\varepsilon_d^2.$$

Hence, since $\varepsilon_d \le \alpha$ we have

$$4\alpha\varepsilon_d \ge \varepsilon_{d+1} \ge 4\alpha\varepsilon_d-\left((2-\alpha)^2-2(1-\alpha)\right)\varepsilon_d^2 > 4\alpha(1-\varepsilon_d)\varepsilon_d.$$

Therefore, $\varepsilon_d \le (4\alpha)^d\varepsilon_0$. Further, we obtain

$$\varepsilon_s \ge 4\alpha(1-\varepsilon_{s-1})\varepsilon_{s-1} \ge (4\alpha)^s\varepsilon_0\prod_{d=0}^{s-1}(1-\varepsilon_d) \ge (4\alpha)^s\varepsilon_0\prod_{d=0}^{s-1}(1-(4\alpha)^d\varepsilon_0). \tag{25}$$

With the use of inequality $\ln(1-x) \geq -2x$, valid for $0 \leq x \leq 1/2$, and taking into account $(4\alpha)^s \leq \alpha/(8\varepsilon_0)$, the product in (25) can be estimated as

$$\prod_{d=0}^{s-1}(1-(4\alpha)^d\varepsilon_0) \geq e^{-2\varepsilon_0(1+(4\alpha)+\ldots+(4\alpha)^{s-1})} \geq e^{-2\varepsilon_0(4\alpha)^s/(4\alpha-1)} \geq e^{-\alpha/(16\alpha-4)} > \frac{1}{2}.$$

Thus we obtain the first inequality in (24).

2.2. Similarly, denoting $q_d = 1-\alpha-\delta_d$, we have

$$q_{d+1} = 1-(1-(1-\alpha-\delta_d)^2)^2 = 1-\alpha-4\alpha\delta_d-\big((2(1-\alpha)-\delta_d)^2-2(1-\alpha)\big)\,\delta_d^2,$$

following by

$$\delta_{d+1} \leq 4\alpha\delta_d + \big((2(1-\alpha))^2-2(1-\alpha)\big)\,\delta_d^2 < 4\alpha(1+\delta_d/4)\delta_d, \tag{26}$$

and under the additional assumption $\delta_d \leq \alpha/4$ also

$$\delta_{d+1} \geq 4\alpha\delta_d + \big((2(1-\alpha)-\alpha/4)^2-2(1-\alpha)\big)\,\delta_d^2 \geq 4\alpha\delta_d.$$

Thus, the second inequality in (24) holds, $\delta_s \geq (4\alpha)^s\delta_0$, if only $\delta_{s-1} \leq \alpha/4$.

2.3. Let us prove by induction that $\delta_d < 2(4\alpha)^d\delta_0$ for $d \leq s$. For $d = 0$ there is nothing to prove. Let us verify the inductive step from $d-1$ to d. According to (26), the induction hypothesis, the inequalities $1+x \leq e^x$ and $(4\alpha)^s \leq \alpha/(8\delta_0)$, we have

$$\begin{aligned}\delta_d &\leq 4\alpha(1+\delta_{d-1}/4)\delta_{d-1} \leq (4\alpha)^d\delta_0\prod_{i=0}^{d-1}(1+\delta_i/4) \leq \\ &\leq (4\alpha)^d\delta_0\prod_{i=0}^{d-1}(1+(4\alpha)^i\delta_0/2) \leq (4\alpha)^d\delta_0 e^{(1+4\alpha+\ldots+(4\alpha)^{d-1})\delta_0/2} \leq \\ &\leq (4\alpha)^d\delta_0 e^{\delta_0(4\alpha)^d/(8\alpha-2)} \leq (4\alpha)^d\delta_0 e^{\alpha/(64\alpha-16)} < 2(4\alpha)^d\delta_0.\end{aligned}$$

As a consequence, we obtain $\delta_s \leq \alpha/4$, which is what was required. □

Lemma 5. *Let $p_s < \alpha-\gamma$ and $q_s < 1-\alpha-\gamma$ for a constant $\gamma > 0$. Then for any r and some $u = \log_2 r + O(1)$, we have $p_{s+u}, q_{s+u} < 1/2^{r+1}$.*

Proof. **1.** First, we show that $p_{s+u'}, q_{s+u'} \leq 1/8$ for a suitable $u' = O(1)$. We can assume that γ is sufficiently small, say, $\gamma \leq 0.3$. From (23) it follows

$$p_{d+1} = p_d^2(2-p_d)^2, \qquad q_{d+1} = q_d^2(2-q_d^2). \tag{27}$$

It is easy to verify that the functions $p(x) = x - x^2(2-x)^2$ and $q(x) = x - x^2(2-x^2)$ on the intervals $I_p = [1/8,\, \alpha-\gamma]$ and, respectively, $I_q =$

$[1/8,\, 1-\alpha-\gamma]$ are convex upwards, therefore they take minimum values at the ends (these values are positive). Hence, there exists $\lambda > 0$ such that $\lambda \le \min_{x\in I_p} p(x), \min_{x\in I_q} q(x)$. Then $p_{d+1} \le p_d - \lambda$ as soon as $p_d \in I_p$ and $q_{d+1} \le q_d - \lambda$ as soon as $q_d \in I_q$. This means that we can choose $u' = (1-\alpha-\gamma-1/8)/\lambda$.

2. From (27) it is clear that $p_{d+1} \le 4p_d^2$ and $q_{d+1} \le 2q_d^2$. Therefore, starting from $p_{s+u'}, q_{s+u'} \le 1/8$, we derive

$$p_{s+u'+i} \le 2^{-(2^i+2)}, \qquad q_{s+u'+i} \le 2^{-(2^{i+1}+1)}.$$

Thus, setting $i = \lceil \log_2 r \rceil$ we obtain the required estimates. □

Applying Lemma 5 with $r = \log\left(C_n^{k-t} + C_n^{k+t}\right)$, we find that with probability $< 1/2^{r+1}$ a formula from Δ_{s+u} incorrectly computes the function $\Psi_n^{k,t}$ on input X_0, and the same goes with X_1. As a consequence, with probability $> 1/2$ a formula from Δ_{s+u} (its depth $\lceil \log_2 m \rceil + 2(s+u)$) correctly computes the function $\Psi_n^{k,t}$ on the boundary of the uncertainty domain, and hence, due to monotonicity, on the entire definitional domain. ■

8 Some open problems

1. We have seen that threshold symmetric functions can be computed slightly more simply than general symmetric functions and even than the counting operator C_n. Is it possible to propose a special economical way to compute elementary periodic functions (MOD-functions) ($\sum x_i \equiv r \bmod q$) for all values of the period q? For the model of switching circuits, such a method is proposed in [35], but it relies on the property inherent in switching circuits to efficiently compute periodic functions with small periods.

2. The approximate summation method §3 is based on the monotone Valiant's procedure. Is it possible to propose a more economical approximate summation procedure that fully utilizes the completeness of the basis?

3. Is it possible to specify a finite basis in which the formula complexity of symmetric functions, in particular C_n or Maj_n, is $n^{1+o(1)}$?

References

[1] Avizienis A. *Signed-digit number representations for fast parallel arithmetic.* IRE Trans. on Electr. Comp. 1961. EC-**10**, 389–400.

[2] Beame P. W., Cook S. A., Hoover H. J. *Log depth circuits for division and related problems.* SIAM J. Comput. 1986. **15**(4), 994–1003.

[3] Boppana R. B. *Amplification of probabilistic Boolean formulas.* Proc. FOCS (Portland, 1985). IEEE, 1985, 20–29.

[4] Cherukhin D. Yu. *Lower bounds for the formula complexity of symmetric Boolean functions.* Diskr. Analiz i Issled. Oper., Ser. 1. 2000. **7**(3), 86–98. (in Russian)

[5] Chin A. *On the depth complexity of the counting functions.* Inform. Proc. Letters. 1990. **35**, 325–328.

[6] Chiu A., Davida G., Litow B. *Division in logspace-uniform NC^1.* Theoret. Informatics Appl. 2001. **35**, 259–275.

[7] Demenkov E., Kojevnikov A., Kulikov A., Yaroslavtsev G. *New upper bounds on the Boolean circuit complexity of symmetric functions.* Inform. Proc. Letters. 2010. **110**(7), 264–267.

[8] Fischer M. J., Meyer A. R., Paterson M. S. $\Omega(n \log n)$ *lower bounds on length of Boolean formulas.* SIAM J. Comput. 1982. **11**(3), 416–427.

[9] Gashkov S. B. *Entertaining computer arithmetic. Fast algorithms for operations with numbers and polynomials.* Librokom, Moscow, 2012. (in Russian)

[10] Grove E. *Proofs with potential.* Ph.D. thesis. Univ. California, Berkeley, 1993.

[11] Karatsuba A. A., Ofman, Yu. P. *Multiplication of multidigit numbers on automata.* Soviet Physics Doklady. 1963. **7**, 595–596. (translated from Russian)

[12] Khrapchenko V. M. *Asymptotic estimation of addition time of a parallel adder.* in Problemy Kibernetiki. Vol. 19. Nauka, Moscow, 1967, 107–120. (in Russian)

[13] Khrapchenko V. M. *Method of determining lower bounds for the complexity of P-schemes.* Math. Notes. 1971. **10**(1), 474–479. (translated from Russian)

[14] Khrapchenko V. M. *The complexity of the realization of symmetrical functions by formulae.* Math. Notes. 1972. **11**(1), 70–76. (translated from Russian)

[15] Khrapchenko V. M. *On the complexity of realization of symmetric Boolean functions by formulas over finite bases.* in Problemy Kibernetiki. Vol. 31. Nauka, Moscow, 1976, 231–234. (in Russian)

[16] Khrapchenko V. M. *Some estimates of multiplication time,* in Problemy Kibernetiki. Vol. 33. Nauka, Moscow, 1978, 221–227. (in Russian)

[17] van Leijenhorst D. C. *A note on the formula size of the "mod k" functions.* Inform. Proc. Letters. 1987. **24**, 223–224.

[18] Lupanov O. B. *On a certain approach to the synthesis of control systems — the principle of local coding*, in Problemy Kibernetiki. Vol. 14. Nauka, Moscow, 1965, 31–110. (in Russian)

[19] Lupanov O. B. *On computing symmetric functions of propositional calculus by switching networks*, in Problemy Kibernetiki. Vol. 15. Nauka, Moscow, 1965, 85–99. (in Russian)

[20] McColl W. F. *Some results on circuit depth.* Theory of computation. Report no. 18. Coventry, Univ. Warwick, 1977.

[21] Ofman, Yu. P. *On the algorithmic complexity of discrete functions.* Soviet Physics Doklady. 1963. **7**, 589–591. (translated from Russian)

[22] Orzel I. *Computing the elementary symmetric polynomials in positive characteristics.* 2025. arXiv:2509.05009.

[23] Paterson M. S. *New bounds on formula size.* Proc. 3rd GI-Conf. on Theory of Comput. Sci. (Darmstadt, 1977). LNCS. 1977. **48**, 17–26.

[24] Paterson M. S., Pippenger N., Zwick U. *Faster circuits and shorter formulae for multiple addition, multiplication and symmetric Boolean functions.* Proc. FOCS (St. Louis, 1990). IEEE, 1990, 642–650.

[25] Paterson M., Zwick U. *Shallow multiplication circuits.* Proc. 10th IEEE Symp. on Comp. Arithm. (Grenoble, 1991). IEEE, 1991, 28–34.

[26] Paterson M. S., Pippenger N., Zwick U. *Optimal carry save networks.* in LMS Lecture Notes Series. Boolean function complexity. Vol. 169. Cambridge Univ. Press, 1992, 174–201.

[27] Paterson M., Zwick U. *Shallow circuits and concise formulae for multiple addition and multiplication.* Comput. Complexity. 1993. **3**, 262–291.

[28] Peterson G. L. *An upper bound on the size of formulae for symmetric Boolean function.* Tech. Report 78–03–01. Univ. Washington, 1978.

[29] Pippenger N. *Short formulae for symmetric functions.* IBM report RC–5143. Yorktown heights, NY, 1974.

[30] Sergeev I. S. *Upper bounds for the formula size of the majority function.* 2012. arXiv:1208.3874.

[31] Sergeev I. S. *Upper bounds on the depth of symmetric Boolean functions.* Moscow Univ. Comput. Math. and Cybern. 2013. **37**(4), 195–201. (translated from Russian)

[32] Sergeev I. S. *Upper bounds for the formula size of symmetric Boolean functions.* Russian Math. 2014. **58**(5), 30–42. (translated from Russian)

[33] Sergeev I. S. *Complexity and depth of formulas for symmetric Boolean functions.* Moscow Univ. Math. Bulletin. 2016. **71**(3), 127–130. (translated from Russian)

[34] Sergeev I. S. *Upper bounds for the size and the depth of formulae for MOD-functions.* Discrete Math. and Applications. 2017. **27**(1), 15–22. (translated from Russian)

[35] Sinha R. K., Thathachar J. S. *Efficient oblivious branching programs for threshold and Mod functions.* J. Comput. and System Sci. 1997. **55**(3), 373–384.

[36] Valiant L. G. *Short monotone formulae for the majority function.* J. Algorithms. 1984. **5**, 363–366.

[37] Wallace C. S. *A suggestion for a fast multiplier.* IEEE Trans. on Electr. Comp. 1964. EC-**13**, 14–17.

[38] Wegener I. *The complexity of Boolean functions.* Wiley–Teubner, Stuttgart, 1987.

[39] Yablonskii S. V. *Introduction to discrete mathematics.* Nauka, Moscow, 1986. (in Russian)