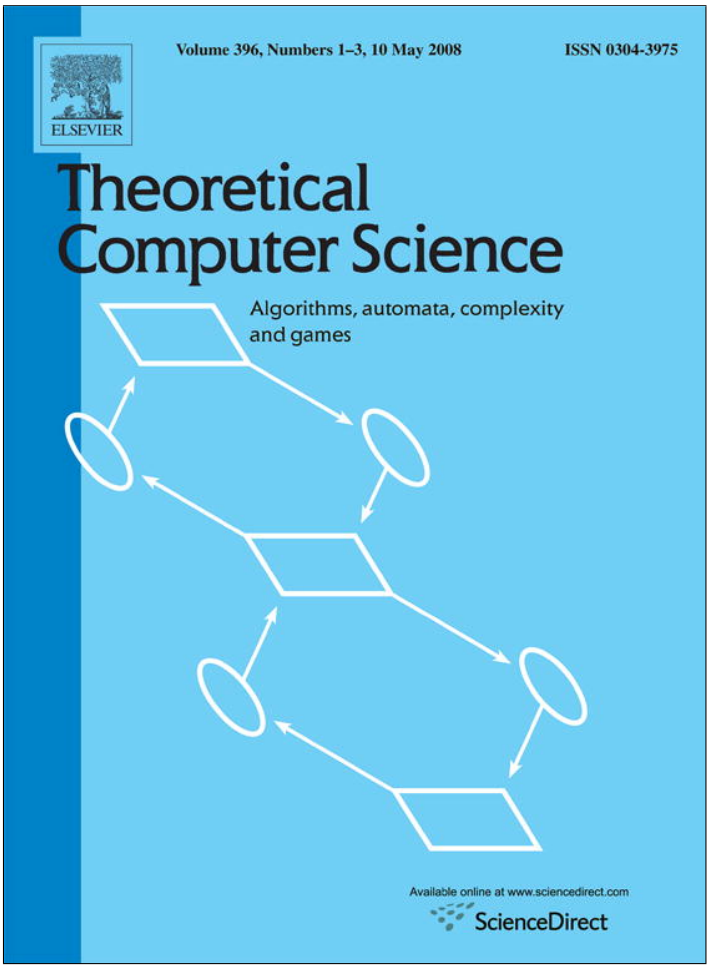

Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 396 (2008) 223–246
Theoretical Computer Science
www.elsevier.com/locate/tcs

Tight bounds for the multiplicative complexity of symmetric functions☆

Joan Boyar[a], René Peralta[b,*]

[a] *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*
[b] *Information Technology Laboratory, National Institute of Standards and Technology, USA*

Received 31 May 2007; received in revised form 5 January 2008; accepted 17 January 2008

Communicated by F. Cucker

Abstract

The multiplicative complexity of a Boolean function f is defined as the minimum number of binary conjunction (AND) gates required to construct a circuit representing f, when only exclusive-or, conjunction and negation gates may be used. This article explores in detail the multiplicative complexity of symmetric Boolean functions. New techniques that allow such exploration are introduced. They are powerful enough to give exact multiplicative complexities for several classes of symmetric functions. In particular, the multiplicative complexity of computing the Hamming weight of n bits is shown to be exactly $n - H^{\mathbb{N}}(n)$, where $H^{\mathbb{N}}(n)$ is the Hamming weight of the binary representation of n. We also show a close relationship between the complexities of basic symmetric functions and the fractal known as Sierpinski's gasket.

Keywords: Circuit complexity; Multiplicative complexity; Symmetric functions; Multi-party computation; Cryptographic proofs

1. Introduction

Much research in circuit complexity is devoted to the following problem: Given a Boolean function and a supply of gate types, construct a circuit which computes the function and is optimal according to some criteria. It seems to be very difficult in general to obtain exact bounds for specific functions. The *multiplicative complexity* $c_\wedge(f)$ of a Boolean function f is the number of conjunctions necessary and sufficient to implement a circuit which computes f over the basis $(\wedge, \oplus, 1)$ (alternatively, the number of multiplications necessary and sufficient to calculate a function over GF_2 via a straight-line program).

Our initial motivation for studying multiplicative complexity came from cryptography. Many cryptographic protocols involve proving predicates about a string X that is available in *committed* form only, i.e. the bits of X

☆ A preliminary version of this work appeared in Mathematical Foundations of Computer Science (MFCS 2006), volume 4162 of Lecture Notes in Computer Science, pages 179–189, Springer-Verlag, 2006.

* Corresponding author. Tel.: +1 (301) 975 8702.
E-mail address: peralta@nist.gov (R. Peralta).

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2008.01.030

are individually encrypted using a *bit-commitment scheme* satisfying standard cryptographic properties (see [2,5, 6]). In [4] a construction is given for a noninteractive cryptographic proof of an arbitrary predicate F on X. The predicate F is defined by a verification circuit C containing AND, NOT, and XOR gates only. For example, X could be a commitment to a bidding price in a sealed-bid auction. A predicate of interest in this scenario might include $F(X) =$"$X \geq 100$", meaning that the offer is at least \$100. The construction in [4] is called a *discreet proof*, and it reveals no information about X other than what is inferable from the value of $F(X)$. Discreet proofs are useful in a wide variety of applications, e.g. electronic voting, online sealed-bid auctions, contract signing, telemedicine, etc. The length of these *discreet proofs* is linear in the number of AND gates in C and is unaffected by the number of NOT or XOR gates.

Perhaps even more important, though, are applications to the communication complexity of secure multi-party computation. In general, for these protocols, multiplications require communication, but linear operations do not. This holds for very different paradigms for building protocols, those based on secret sharing were introduced in [7,9] and those based on threshold homomorphic encryption were introduced in [10]. For more recent results, see [14].

We focus on symmetric functions, which are functions dependent only on the Hamming weight $\overrightarrow{H}(\mathbf{x})$ of the input $\mathbf{x} \in GF_2^n$. Obtaining tight bounds is important because symmetric functions can be building blocks for arithmetic circuits, some of which involve recursive use of simple symmetric functions. Suboptimal implementations of the latter, even by an additive constant factor, translate into multiplicative extra costs when building arithmetic circuits. In cryptographic applications, whether or not a circuit is of practical use often depends on constant multiplicative factors in the number of AND gates used.

The study of multiplicative complexity may prove useful in obtaining upper bounds on the computational complexity of functions. If a function f has multiplicative complexity $O(\log(n))$, then, for all $\mathbf{x}$ in the domain of f, an element of the pre-image of $y = f(\mathbf{x})$ can be found in polynomial time as follows: Guess the values of inputs to the AND gates in a circuit for f, reducing the circuit to a collection of linear circuits. Then, find an $\mathbf{x}$ such that $y = f(\mathbf{x})$ using Gaussian elimination over GF_2. Therefore, one-way functions, if they exist, have super-logarithmic multiplicative complexity. On the other hand, low multiplicative complexity circuits may lead to better algorithms for inverting functions of importance in cryptology. There is no known satisfactory classification of functions with low multiplicative complexity. A step in this direction is the work of Fischer and Peralta [12]. They show that the number of predicates on n bits which can be computed with only one AND the gate is exactly $2^n(2^n-1)(2^n-2)/3$ for $n \geq 2$.

1.1. Previous work

Multiplicative complexity has been investigated previously by Aleksanyan [1], Schnorr [20], and Mirwald and Schnorr [17]. Their work was exclusively concerned with quadratic Boolean forms. Multiplicative complexity has more often been used to refer to more general algebraic computations. This subject has an extensive history (see, for example, [3]), since multiplication is often the dominating operation in this context.

Very little is known about multiplicative complexity of specific functions. In this paper we concentrate on the concrete (as opposed to asymptotic) multiplicative complexity of symmetric functions. In an earlier paper ([8]), we showed the following asymptotic results:

- A general upper bound of $n + 3\sqrt{n}$ for any symmetric function f. This establishes a separation between Boolean and multiplicative complexity for symmetric functions. Paul [18] and Stockmeyer [21] have shown lower bounds of the form $2.5n - O(1)$ for the Boolean complexity of infinite families of symmetric functions;
- Let Σ^n be the set of symmetric predicates on n bits. We showed an upper bound of $2n - \log_2 n$ for the complexity $c_\wedge(\Sigma^n)$ of simultaneously computing all symmetric functions on n bits (the asymptotic result $c_\wedge(\Sigma^n) = O(n)$ was obtained earlier by Mihaĭljuk [16]).

1.2. Our results

Several new upper and lower bounds on the multiplicative complexity of symmetric functions are obtained. In particular, it is shown that the multiplicative complexity of computing the Hamming weight is exactly $n - H^{\mathbb{N}}(n)$, where $H^{\mathbb{N}}(n)$ is the Hamming weight of the binary representation of n. This is a rather surprising result, given the sparsity of exact computational complexity bounds known. The construction also proves to be a powerful tool in obtaining other exact results for symmetric functions.

A new technique, using a normal form for $(\oplus, 1, \wedge)$ circuits and elementary linear algebra, is used to show that any nonlinear symmetric function on n variables has multiplicative complexity at least $\left\lfloor \frac{n}{2} \right\rfloor$. Properties of binomial coefficients are shown to yield the following lower bounds for the counting (exactly-k) and threshold-k functions on n variables:

$$c_\wedge(E_k^n) \geq \max\{k-1, n-k-1, 2^{\lfloor \log_2 n \rfloor} - 2, l_{n,k} - 1\}$$
$$c_\wedge(T_k^n) \geq \max\{k-1, n-k, 2^{\lfloor \log_2 n \rfloor} - 1, l_{n-1,k-1}\}$$

where $l_{n,k}$ is the bitwise OR of $n-k$ and k. Tighter bounds for several families of symmetric functions are obtained by considering the multiplicative complexity of such functions when restricted to hyperplanes in GF_2^n. In particular, this technique yields the exact complexities of the elementary symmetric functions Σ_2^n, Σ_3^n, Σ_{n-1}^n, Σ_{n-2}^n, Σ_{n-3}^n. Yet another application of hyperplane restrictions yields new general lower bounds for infinite subclasses of symmetric functions. Intriguingly, these subclasses are defined by fractals on the Cartesian plane.

More constructively, general techniques are developed for proving upper bounds for elementary symmetric functions. These, plus Pascal's triangle modulo 2 (known in the fractals literature as Sierpinski's gasket, see Fig. 3) are used to prove upper bounds for the counting functions, $E_k^n(\mathbf{x})$, and the threshold functions, $T_k^n(\mathbf{x})$. These general techniques are shown to give many tight results. In addition, a general upper bound on the threshold-k functions, T_k^n, is found: $c_\wedge(T_k^n) \leq n - H^{\mathbb{N}}(n) + \lceil \log_2(n+1) \rceil - 1$ for all $k \geq 1$.

The exact multiplicative complexities of the elementary symmetric functions, the counting functions, and the threshold functions are determined for $n \leq 7$. For elementary symmetric functions, the exact complexities are found for the kth elementary symmetric function, Σ_k^n, on n variables, for $k = 2, 3, n-3, n-2$, and $n-1$.

2. Some simple observations and a normal form

Each Boolean function f on n variables has a unique representation as a multilinear (i.e. square-free) polynomial over GF_2. Since $x^i = x$ over GF_2, we assume throughout the following that all polynomials are multilinear. By the "degree of f", we will mean the degree of its unique representing polynomial. It is known that a Boolean function of degree d has multiplicative complexity at least $d-1$ [20]. This we call the *degree lower bound*.

We say that a circuit is optimal for f if it has $c_\wedge(f)$ AND gates. Since $y \wedge (x \oplus 1) = (y \wedge x) \oplus y$, optimal circuits need not have more than one negation. If present, we may assume this negation is the last gate in the circuit. It is easy to see that a Boolean function $f(\mathbf{x})$ requires a negation if and only if $f(\mathbf{0}) = 1$. This in turn is true if and only if the polynomial of f has a constant term. Thus we may divide Boolean functions into "positive" functions (those for which $f(\mathbf{0}) = 0$) and "negative" functions. There is a bijection $\sigma(f) = f \oplus 1$ between positive and negative functions. The bijection preserves multiplicative complexity. Therefore we may restrict our study of multiplicative complexity to functions over the basis $(\oplus, \wedge)$. For technical reasons, and without affecting the multiplicative complexity of functions, we will allow $\oplus$ gates to contain any number of inputs (at least one). AND gates, though, are restricted to fan-in exactly 2.

A function $f : GF_2^n \to GF_2$ is "linear" (sometimes called "affine") if it is of the form $a_0 + \sum_{i=1}^n a_i x_i$ with each $a_i \in GF_2$. That is, linear functions are precisely those functions having multiplicative complexity 0. We call a gate *internal* if its output is not the output to the circuit. We say a circuit is in *Layered Normal Form* (LNF) if

- all inputs go only to $\oplus$ gates;
- outputs of all internal $\oplus$ gates are inputs only to $\wedge$ gates.

It is not hard to see that all positive functions have optimal circuits in Layered Normal Form.

Logical expressions over the basis $(\wedge, \oplus)$ correspond to arithmetic expressions over GF_2. We will use the latter notation for the most part of this paper: $a \oplus b$, $a \wedge b$, $\bar{a}$ will be written $a \oplus b$, ab, $a \oplus 1$, respectively.

The kth elementary symmetric function on n variables $x_1, x_2, \ldots, x_n$ is defined by

$$\Sigma_k^n(x_1, x_2, \ldots, x_n) = \bigoplus_{S \subseteq \{1,\ldots,n\}, |S|=k} \prod_{i \in S} x_i \qquad (1 \leq k \leq n).$$

For readability we will also use the alternative notations $\Sigma_k^n(\mathbf{x})$ or simply Σ_k^n. It will prove convenient as well to define $\Sigma_0^n = 1$.

A classical result states that every symmetric function can be represented as a sum of elementary symmetric functions (see [22]). Consider, for example, the MAJORITY function on three variables (i.e. the threshold function T_2^3). We have

$$\begin{aligned} T_2^3(x_1, x_2, x_3) &= \Sigma_2^3(x_1, x_2, x_3) \\ &= x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \\ &= x_1(x_2 \oplus x_3) \oplus x_2x_3 \\ &= (x_1 \oplus x_2)(x_1 \oplus x_3) \oplus x_1. \end{aligned}$$

The last equality establishes $c_\wedge(T_2^3) = 1$, and also serves to show that the algebraic manipulations necessary to obtain optimal circuits may not be obvious.

The following lemmas appear in [8]:

Lemma 1. *Represent the positive integer k as a sum of powers of 2: $k = 2^{i_0} + 2^{i_1} + \cdots + 2^{i_j}$. Each i is a position of a nonzero bit in the binary representation of k. Then for any $n \geq k$,*

$$\Sigma_k^n = \Sigma_{2^{i_0}}^n \Sigma_{2^{i_1}}^n \cdots \Sigma_{2^{i_j}}^n.$$

For example, $\Sigma_{11}^n = \Sigma_8^n \Sigma_2^n \Sigma_1^n$ for $n \geq 11$.

Lemma 2. *Let $\mathbf{y} = y_k y_{k-1} \ldots y_0$ be the Hamming weight, in binary representation, of the n-bit string $\mathbf{x}$. Then $y_i = \Sigma_{2^i}^n(\mathbf{x})$ for $i = 0, \ldots, k$.*[1]

For example, the Hamming weight of a 10 bit string $\mathbf{x}$ is given by the 4 bit string

$$\Sigma_8^{10}(\mathbf{x})\Sigma_4^{10}(\mathbf{x})\Sigma_2^{10}(\mathbf{x})\Sigma_1^{10}(\mathbf{x}).$$

Finally, we observe that if $g : GF_2^k \to GF_2$ is derived from $f : GF_2^n \to GF_2$ by fixing the values of $n - k$ variables of f, then $c_\wedge(g) \leq c_\wedge(f)$. We call g a *restriction* of f.

3. A tight lower bound on the multiplicative complexity of symmetric functions

In this section we prove a lower bound of $\lfloor \frac{n}{2} \rfloor$ for nonlinear symmetric functions. This bound is met by infinitely many functions. We go to some length to prove this tight bound in part because we expect symmetric functions to be building blocks for arithmetic circuits, some of which involve recursive use of simple symmetric functions. Suboptimal implementations of the latter, even by an additive constant factor, translate into multiplicative extra costs when building arithmetic circuits. In cryptographic applications of this theory, whether or not a circuit is of practical use often depends on constant multiplicative factors in the number of AND gates used.

Given a Boolean function f over GF_2^n and a subset S of $\{x_1, \ldots, x_n\}$, we denote by $f_{\bar{S}}$ the function obtained from f by complementing the inputs in S. If $f_{\bar{S}} = f$, we say S is *complementable*. We say S is "proper" if $0 < |S| < n$.

Lemma 3. *If a Boolean function f over GF_2^n has multiplicative complexity less than $\left\lfloor \frac{n-1}{2} \right\rfloor$, then it has a proper complementable set.*

Proof. Consider an optimal LNF circuit for f. Since the circuit has at most $\left\lfloor \frac{n-1}{2} \right\rfloor - 1$ AND gates, the number of $\oplus$ gates is at most $k = 2(\left\lfloor \frac{n-1}{2} \right\rfloor - 1) + 1 \leq n - 2$ (recall that a circuit in LNF form may have at most one $\oplus$ gate which is not the input to an $\wedge$ gate). Label these gates $\gamma_1, \ldots, \gamma_k$. Define an $n \times k$ matrix $A = (a_{ij})$ over GF_2 as follows:

$a_{ij} = 1$ iff x_i is an input to γ_j.

Rows of the matrix correspond to inputs of the circuit. Columns correspond to $\oplus$ gates. Since $rank(A) \leq k \leq n - 2$, there is a subset S (with $0 < |S| \leq n - 1$) of the rows whose sum over GF_2^k is $\mathbf{0}$. Since in a LNF circuit all inputs go only to $\oplus$ gates, and each $\oplus$ gate has an even number of inputs from S, S is a complementable set of inputs. □

[1] See also [19].

A slightly stronger result can be obtained for even n. The proof is analogous to the proof of Lemma 3.

Lemma 4. *If n is even and the Boolean function f over GF_2^n has multiplicative complexity less than $\frac{n}{2}$, then f has a nonempty complementable set.*

In the case of a symmetric function f, if a proper set S of cardinality k is complementable, then *every* set of cardinality k is complementable. In particular the sets $\{x_1, \ldots, x_k\}$ and $\{x_2, \ldots, x_{k+1}\}$ are complementable. This means $\{x_1, x_{k+1}\}$ is also complementable, and therefore any two inputs are complementable. Thus if the Hamming weights of $\mathbf{x}$ and $\mathbf{y}$ have the same parity, then $f(\mathbf{x}) = f(\mathbf{y})$. But this means f is linear. We have shown

Lemma 5. *If a symmetric Boolean function f has a proper complementable set S, then f must be linear (i.e. $c_\wedge(f) = 0$).*

Lemmas 3 and 5 yield a general bound of $\left\lfloor \frac{n-1}{2} \right\rfloor$ for nonlinear symmetric functions. We now marginally improve this bound. This is of practical interest, and the techniques developed to prove this result are of use later in this paper.

Let b_i be the value of $f(\mathbf{x})$ when $\mathbf{x}$ has Hamming weight i. The bit string $b_0, \ldots, b_n$ is called the *spectrum* of f. Suppose a nonlinear symmetric function f has a non-empty complementable set S. Then, by Lemma 5, S must be the set of all inputs, i.e. $f(\mathbf{x}) = f(\bar{\mathbf{x}})$ for all $\mathbf{x}$. A symmetric function which has this property is called *palindromic* (since its spectrum is a palindrome).

The notions of complementable set and palindromic function have duals which are also useful: given a Boolean function f, we say a set of inputs $S = \{x_{i_1}, \ldots x_{i_k}\}$ is *anti-complementable* if, for all $\mathbf{x}$, $f_{\bar{S}}(\mathbf{x}) = f(\mathbf{x}) \oplus 1$. Note that if the set of all inputs is anti-complementable, then the function is *self-dual* [11]. A symmetric function $f : GF_2^n \to GF_2$ is *anti-palindromic* if $f(\mathbf{x}) = f(\bar{\mathbf{x}}) \oplus 1$ for all $\mathbf{x}$. Note that f may be anti-palindromic only for odd n, and that linear symmetric functions are either palindromic or anti-palindromic.

We leave to the reader the proof of the dual to Lemma 5:

Lemma 6. *If a symmetric Boolean function f has a proper anti-complementable set S, then f must be linear (i.e. $c_\wedge(f) = 0$).*

We can now show the following:

Theorem 1. *If n is odd, $f : GF_2^n \to GF_2$ is symmetric, and $c_\wedge(f) < \frac{n+1}{2}$, then f is either palindromic or anti-palindromic.*

Proof. We have already noted that if f is linear then it is palindromic or anti-palindromic. For nonlinear f, let $g : GF_2^{n+1} \to GF_2$ be defined by $g(x_1, \ldots, x_{n+1}) = f(x_1, \ldots, x_n) \oplus x_{n+1}$. Clearly, $c_\wedge(g) < \frac{n+1}{2}$. By Lemma 4, g has a nonempty complementable set S. There are two cases to consider.

- S does not contain x_{n+1}: then S is a complementable set for f. Since f is nonlinear, S cannot be a proper subset of $\{x_1, \ldots, x_n\}$ (by Lemma 5). Thus $S = \{x_1, \ldots x_n\}$ and therefore f is palindromic.
- S contains x_{n+1}: then $f(\mathbf{x}) \oplus x_{n+1} = f_{\bar{S}}(\mathbf{x}) \oplus \bar{x}_{n+1} = f_{\bar{S}}(\mathbf{x}) \oplus x_{n+1} \oplus 1$. Therefore, $f(\mathbf{x}) = f_{\bar{S}}(\mathbf{x}) \oplus 1$. Thus, $S - \{x_{n+1}\}$ is an anti-complementable set for f. Since f is non-linear, Lemma 6 implies $S - \{x_{n+1}\} = \{x_1, \ldots, x_n\}$. Therefore f is anti-palindromic. □

Theorem 1 is not vacuous. In particular Σ_2^{4k+1} is palindromic with complexity $2k$, and Σ_2^{4k+3} is anti-palindromic with complexity $2k + 1$ (see Corollary 5).

We can now prove the main result of this section.

Theorem 2. *Let $f : GF_2^n \to GF_2$ be a nonlinear symmetric function. Then f has multiplicative complexity at least $\left\lfloor \frac{n}{2} \right\rfloor$.*

Proof. Since f is non-linear, by Lemma 5, f may not have a proper complementable set. For odd n, Lemma 3 gives the stated bound. Now suppose, for a contradiction, that n is even and $c_\wedge(f) < \frac{n}{2}$. Then f has a nonempty complementable set S by Lemma 4. Therefore, by Lemma 5, S must be the set of all inputs to f, i.e. $f(\mathbf{x}) = f(\bar{\mathbf{x}})$. Let $b_0, \ldots, b_n$ be the palindrome defined by letting b_i be the value of $f(\mathbf{x})$ when $\mathbf{x}$ has Hamming weight i. Consider the function g defined as a restriction of f by setting x_n to zero. The function g is a symmetric function of $n - 1$

step 1: let $t = Max(I_E)$;
step 2: for each XOR gate with x_t as one of its inputs
  step 2.1: add inputs x_i for all $i \in I_E$;
  step 2.2: if two inputs are the same, delete both (this deletes input x_t);
step 3: if an XOR gate has no inputs, remove the gate and set the wire that carried its output to 0;
step 4: if any input to an AND gate is 0, remove the gate and set the wire that carried its output to 0;
step 5: remove all 0 inputs from XOR gates;
step 6: repeat steps $3-5$ until inputs to all gates are non-zero.

Fig. 1. Hyperplane restriction of a circuit in Layered Normal Form.

variables, and $c_\wedge(g) \leq c_\wedge(f) < \frac{(n-1)+1}{2} = \frac{n}{2}$, so Theorem 1 implies that g is either palindromic or anti-palindromic. First, assume that g is palindromic. Since f is palindromic, $b_0 = b_n$. Since g is palindromic $b_0 = b_{n-1}$. Thus, $b_{n-1} = b_n$. Similarly, $b_1 = b_{n-1}$ and $b_1 = b_{n-2}$, giving that $b_0 = b_1 = b_{n-2} = b_{n-1} = b_n$. Continuing in this manner, one sees that f is a constant function, contradicting the assumption that it is nonlinear. Assuming that g is anti-palindromic, one sees similarly that $b_i \neq b_{i+1}$ for $0 \leq i \leq n-1$, so f is one of the two parity functions. This again contradicts the assumption that it is non-linear. Thus, f must have multiplicative complexity at least $\lfloor \frac{n}{2} \rfloor$. □

This bound is tight: the multiplicative complexity of $\Sigma_2^n(\mathbf{x})$ is $\lfloor \frac{n}{2} \rfloor$ (see Theorem 9).

4. Hyperplane restrictions yield fractal lower bounds

In the following we investigate a new technique which uses the degree lower bound, but often achieves stronger lower bounds than that given by the degree lower bound alone. A plane E in GF_2^n can be specified by an equation $\bigoplus_{i\in I_E} x_i = 0$, where I_E is a subset of $\{1, \ldots, n\}$. For notational simplicity, if the index set is empty, we define $\bigoplus_{i\in\phi} x_i = 0$. Given a Boolean function f on n-bits, we denote the restriction of f to the plane E by $f_{\downarrow E}$. Letting $t = \mathrm{Max}(I_E)$, we view $f_{\downarrow E}$ as a function on $n-1$ variables obtained by substituting $\bigoplus_{i\in I_E-\{t\}} x_i$ for x_t in the polynomial for f. There are many ways to obtain a circuit for $f_{\downarrow E}$ from a circuit for f. For C in Layered Normal Form, $C_{\downarrow E}$ will denote the circuit constructed in the straightforward manner specified in Fig. 1. Note that $C_{\downarrow E}$ is also in Layered Normal Form.

We now proceed to prove lower bounds by choosing planes which will decrease the number of AND gates in a circuit without decreasing the degree of the function which is computed. The degree lower bound is then applied to the function resulting from the restriction.

Lemma 7. *Suppose f is an n-variate function of degree $k > 1$. If $c_\wedge(f) = k - 1 + e$, where $e \geq 0$, then there exist $u \leq e + 1$ planes $E_1, E_2, \ldots, E_u$ such that the degree of $(\ldots((p_{\downarrow E_1})_{\downarrow E_2})\ldots)_{\downarrow E_u}$ is at most $k-1$.*

Proof. Consider an optimal circuit C for f in Layered Normal Form. An AND gate of minimal depth in this circuit has two distinct linear inputs. Let one be $\bigoplus_{i\in I_E} x_i$. Then the circuit $C_{\downarrow E}$ contains at least one fewer AND gate than does C. Repeating this at most e more times yields a circuit computing a function with at least $e+1$ fewer AND gates than C. This function has multiplicative complexity at most $k-2$ and therefore has degree at most $k-1$. □

For a symmetric function, Lemma 7 yields

Corollary 1. *Suppose f is an n-variate symmetric function of degree $k > 1$. If $c_\wedge(f) = k-1$, then $\deg(f_{\downarrow E}) \leq k-1$ for at least two distinct planes E_1, E_2 where E_1 can be specified by $x_n = \bigoplus_{i=1}^{t_1} x_i$ $(t_1 < n)$, and E_2 can be specified using an equation with at most $n-2$ terms in the sum.*

Proof. The first plane is found by using the proof of Lemma 7 and relabelling the variables (since f is symmetric). Since the two inputs to the AND gate used in the proof are different, at least one does not contain all n variables; this gives the second plane. □

This technique of hyperplane restrictions yields lower bounds on multiplicative complexity which are better than the straightforward degree lower bound for almost all symmetric functions. We next state some of these bounds. In Section 7.1, the bound given by the following theorem is shown to be tight for Σ^n_{n-2} and Σ^n_{n-3}:

Theorem 3. *Let f be an n-variate symmetric polynomial of degree m with $1 < m < n-1$ and $n > 3$. Then $c_\wedge(f) \geq m$.*

Proof. We first show the bound for Σ^n_m. By the degree lower bound, $c_\wedge(\Sigma^n_m) \geq m-1$. Suppose, for the sake of contradiction, that $c_\wedge(\Sigma^n_m) = m-1$. Note

$$\Sigma^n_m = x_n \Sigma^{n-1}_{m-1} \oplus \Sigma^{n-1}_m.$$

By Corollary 1 there is some $t < n$ such that the restriction

$$R = \left(\bigoplus_{i=1}^{t} x_i\right)(\Sigma^{n-1}_{m-1}) \oplus \Sigma^{n-1}_m = \bigoplus_{i=1}^{t} (x_i \Sigma^{n-1}_{m-1}) \oplus \Sigma^{n-1}_m$$

has degree at most $m-1$. This is clearly not the case for $t = 0$, so assume that $t > 0$.

Consider a term $x_{i_1} x_{i_2} \ldots x_{i_m}$ in Σ^{n-1}_m. For each variable x_k in $S = \{x_1, x_2, \ldots, x_t\}$, which is equal to some variable x_{i_j} in this term, there is exactly one term, y, in Σ^{n-1}_{m-1} such that $x_k y = x_{i_1} x_{i_2} \ldots x_{i_m}$. Thus, before collapsing equal terms, a term $x_{i_1} x_{i_2} \ldots x_{i_m}$ will occur $s+1$ times in R, where $s = |\{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\} \cap \{x_1, x_2, \ldots, x_t\}|$. For the degree of R to be at most $m-1$, each term of degree m must occur an even number of times. This implies s must be odd. In particular, $|\{x_1, x_2, \ldots, x_m\} \cap \{x_1, x_2, \ldots, x_t\}| = \min\{m, t\}$ is odd. Since $m < n-1$ the set $\{x_{n-1}, x_2, \ldots, x_m\}$ has cardinality m. Thus, if $t < n-1$, then $|\{x_{n-1}, x_2, \ldots, x_m\} \cap \{x_1, x_2, \ldots, x_t\}| = \min\{m-1, t-1\}$ is also odd. This is clearly a contradiction, so t must be equal to $n-1$. Therefore, the only plane that will reduce the degree of Σ^n_m is $x_n = \bigoplus_{i=1}^{n-1} x_i$. By Corollary 1, the restricted functions defined by at least two distinct planes will have degree at most $m-1$. This is a contradiction, so $c_\wedge(\Sigma^n_m) \geq m$.

Finally, we note that the lower bound applies to any symmetric function on n variables with degree m, where $1 < m < n-1$, since, for such a function, the exclusive-or of all terms of degree m is Σ^n_m. □

The proof of Theorem 3 involved removing only one AND gate. This idea can be extended using Lemma 7. In the following, two AND gates are removed. We will need the following observation: for $1 \leq i, j \leq t-1$ and $s < t$, the polynomial $x_i \Sigma^t_s$, has exactly $\binom{t-1}{s}$ terms of degree $s+1$ and $x_i x_j \Sigma^t_s$ has $\binom{t-2}{s}$ terms of degree $s+2$.

Theorem 4. *Let f be a $n-$variate symmetric function of degree m. Suppose $1 < m \leq n-2$ and $n > 4$. Then, if $\binom{n-4}{m-2}$ is even, $\binom{n-3}{m-1}$ is even, and $\binom{n-2}{m}$ is odd, then $c_\wedge(f) \geq m+1$.*

Proof. As was noted in the proof of the previous theorem, it is enough to prove the bound for Σ^n_m.

Suppose, for the sake of contradiction, that $\binom{n-4}{m-2}$ is even, $\binom{n-3}{m-1}$ is even, $\binom{n-2}{m}$ is odd, and $c_\wedge(\Sigma^n_m) \leq m$. Note

$$\Sigma^n_m = x_n(x_{n-1}\Sigma^{n-2}_{m-2} \oplus \Sigma^{n-2}_{m-1}) \oplus x_{n-1}\Sigma^{n-2}_{m-1} \oplus \Sigma^{n-2}_m.$$

Let E_0 and E_1 be the two planes promised by Lemma 7, and let E'_0 be the restriction of E_0 to E_1. E'_0 can be written as $x_n = \bigoplus_{i \in I_{E'_0}} x_i$ and E_1 as $x_{n-1} = \bigoplus_{j \in I_{E_1}} x_j$, where the sets $I_{E'_0}$ and I_{E_1} only contain the first $n-2$ variables. Then, by Lemma 7, the polynomial

$$\left(\bigoplus_{i \in I_{E'_0}} x_i\right)\left(\left(\bigoplus_{j \in I_{E_1}} x_j\right)\Sigma^{n-2}_{m-2} \oplus \Sigma^{n-2}_{m-1}\right) \oplus \left(\bigoplus_{j \in I_{E_1}} x_j\right)\Sigma^{n-2}_{m-1} \oplus \Sigma^{n-2}_m$$

has degree at most $m-1$. The terms of degree m are as follows:

- Those from $(\bigoplus_{i \in I_{E'_0}} x_i)(\bigoplus_{j \in I_{E_1}} x_j)\Sigma^{n-2}_{m-2}$ where neither the x_i nor the x_j are in the term from Σ^{n-2}_{m-2}. The number of these is $\binom{n-4}{m-2}$ for each pair (i, j) with $i \neq j$.
- Those from $(\bigoplus_{i \in I_{E'_0}} x_i)\Sigma^{n-2}_{m-1}$ where the x_i is not in the term from Σ^{n-2}_{m-1}. The number of these is $\binom{n-3}{m-1}$ for each i.

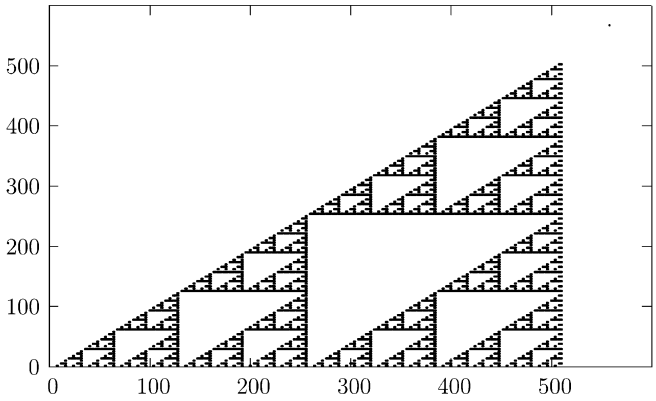

Fig. 2. Points (n,m) for which $c_\wedge(\Sigma_m^n) \geq m+1, m < n < 512$.

- Those from $(\bigoplus_{j\in I_{E_1}} x_i)\Sigma_{m-1}^{n-2}$ where the x_j is not in the term from Σ_{m-1}^{n-2}. The number of these is $\binom{n-3}{m-1}$ for each j.
- Those from Σ_m^{n-2}. The number of these is $\binom{n-2}{m}$.

As in the previous proof, the number of terms of degree m must be even for the polynomial to have degree $m-1$. This cannot be the case if $\binom{n-4}{m-2}$ and $\binom{n-3}{m-1}$ are even, while $\binom{n-2}{m}$ is odd. This gives a contradiction and the result follows. □

Theorem 4 gives the nontrivial lower bound $c_\wedge(\Sigma_4^8) \geq 5$.

Further use of these techniques yields the following theorem (the proof, which we omit, is analogous to the proof of Theorem 4):

Theorem 5. *Let f be a $n-$variate symmetric function of degree m, where $3 \leq m \leq n-3$ and $n > 6$. If $\binom{n-6}{m-3}$, $\binom{n-5}{m-2}$, and $\binom{n-4}{m-1}$ are even, while $\binom{n-3}{m}$ is odd, then $c_\wedge(f) \geq m+2$.*

The set of points in the plane that satisfy the conditions of either Theorem 4 or Theorem 5 form fractals. Fig. 2 plots these points for Theorem 4. The hyperplane restriction technique is a general tool for relating combinatorial constraints to multiplicative complexity. The combinatorial constraints thus derived seem to always yield fractals. An interesting question is whether this is solely a result of the bounding technique or the exact complexity of the elementary symmetric functions is in fact fractal in nature.

5. The exact multiplicative complexity of the Hamming weight function

The result of computing a symmetric function on some inputs is determined completely by the Hamming weight of those inputs. In this section, we investigate the multiplicative complexity of computing the Hamming weight. Let $\overrightarrow{H}(\mathbf{x})$ denote the binary representation of the Hamming weight of a bit string $\mathbf{x} \in GF_2^n$. $\overrightarrow{H}(\mathbf{x})$ has fixed length $\lceil \log_2(n+1) \rceil$ and may contain leading zeros. The function $\overrightarrow{H}()$ will be denoted by H^n when the parameter n needs to be explicitly stated. Let $H^{\mathbb{N}}(n)$ denote the Hamming weight of the binary representation of the integer n. We will show

$$c_\wedge(H^n) = n - H^{\mathbb{N}}(n).$$

It will prove useful to define the Hamming weight of the empty string λ to be 0, i.e. $\overrightarrow{H}(\lambda) = H^{\mathbb{N}}(0) = 0$. We now make some simple observations.

- If $0 \leq i < 2^k$ then $H^{\mathbb{N}}(2^k + i) = 1 + H^{\mathbb{N}}(i)$. (5.1)
- If $0 \leq k$ then $H^{\mathbb{N}}(2^k - 1) = k$. (5.2)
- If $0 \leq a, b, k$ and $n = 2^k - 1 = a + b$ then

$$H^{\mathbb{N}}(n) = H^{\mathbb{N}}(a) + H^{\mathbb{N}}(b). \quad (5.3)$$

- $\overrightarrow{H}(\mathbf{x})$ is the integer sum of the bits of $\mathbf{x}$.

- For all $n \geq m > 0$, there exists a circuit which adds an n-bit number to an m-bit number – plus an optional carry-in bit c – using n AND gates. This is a standard addition circuit using a chain of full adders. A full adder computes the two-bit sum $w_1 w_0$ of three bits b_1, b_2, b_3. Only one AND gate is needed because $w_0 = (b_1 + b_2 + b_3) \bmod 2$ and $w_1 = ((b_1+b_2)(b_2+b_3)+b_2) \bmod 2$. We will refer to this circuit as the *standard addition circuit* (with carry-in c).

Denote by $c_\wedge(ADD(n, m))$ the multiplicative complexity of adding an n-bit number to an m-bit number. An immediate application of the degree lower bound is that $c_\wedge(ADD(n, m)) \geq \text{Max}(n, m)$. This is because $c_\wedge(ADD(n, m)) \geq c_\wedge(ADD(n, 1))$, and the most significant bit of this sum is the product of all $n+1$ input variables. We have already observed that $c_\wedge(ADD(n, m)) \leq \text{Max}(n, m)$. Thus we have shown

Lemma 8. *The multiplicative complexity of adding two integer inputs, of lengths n and m in radix-2 representation, is* $\text{Max}(n, m)$.

We construct a circuit for H^n that uses $n - H^{\mathbb{N}}(n)$ AND gates. Our construction is essentially a recursive version of a construction that appeared in [8]. First we show a circuit for the case $n = 2^k - 1$.

Lemma 9. *Let* $n = 2^k - 1$ *for* $k \geq 0$. *Then* $c_\wedge(H^n) \leq n - H^{\mathbb{N}}(n) = 2^k - (k + 1)$.

Proof. The proof is by induction on k. The cases $k = 0, 1$ are easily verifiable. For $k > 1$, a string $\mathbf{x}$ of length $2^k - 1$ can be split into two strings $\mathbf{u}, \mathbf{v}$, of length $2^{k-1} - 1$ each, plus one string c of length 1. We recursively compute $\overrightarrow{H}(\mathbf{u})$ and $\overrightarrow{H}(\mathbf{v})$. Then we use the standard addition circuit with carry-in c to compute $c + \overrightarrow{H}(\mathbf{u}) + \overrightarrow{H}(\mathbf{v})$. The result is $\overrightarrow{H}(\mathbf{x})$. By induction, and the fact that $\overrightarrow{H}(\mathbf{u})$, $\overrightarrow{H}(\mathbf{v})$ are of length $k - 1$, the number of multiplications used is $2(2^{k-1} - k) + k - 1 = 2^k - (k + 1)$. □

We now consider the general case

Theorem 6. $c_\wedge(H^n) \leq n - H^{\mathbb{N}}(n)$, *for all* $n \geq 1$.

Proof. We have already shown this for the cases $n = 0, 1, 3, 7, 15, 31, \ldots$. We prove the remaining cases by induction on n. Let $\mathbf{x}$ be a string of length $2^k + i$ with $k > 0$ and $0 \leq i < 2^k - 1$. Assume the theorem holds for all values $0 \leq n' < 2^k + i$. As in Lemma 9 we split $\mathbf{x}$ into three strings $\mathbf{u}, \mathbf{v}, c$ of lengths $2^k - 1$, i, and 1 respectively (note that $\mathbf{v}$ may be the empty string). We recursively compute $\overrightarrow{H}(\mathbf{u})$ and $\overrightarrow{H}(\mathbf{v})$. Then we compute the sum $c + \overrightarrow{H}(\mathbf{u}) + \overrightarrow{H}(\mathbf{v})$. The result is $\overrightarrow{H}(\mathbf{x})$. By induction, using Lemma 9 and the fact that $\overrightarrow{H}(\mathbf{u})$, $\overrightarrow{H}(\mathbf{v})$ are of maximum length k, the number of multiplications used is

$$2^k - (k + 1) + (i - H^{\mathbb{N}}(i)) + k = 2^k + i - (1 + H^{\mathbb{N}}(i)) = (2^k + i) - H^{\mathbb{N}}(2^k + i).$$

The last equality is due to observation (5.1). □

Suppose $\mathbf{x}$ is a bit of string of length 2^k. By Lemma 2, the $k + 1$st bit of $\overrightarrow{H}(\mathbf{x})$ is $\Sigma^{2^k}_{2^k}(\mathbf{x})$, which is a polynomial of degree 2^k. Thus, by the degree lower bound, it is not possible to compute the Hamming weight of a string of length 2^k bits using less than $2^k - 1$ multiplications. Since Theorem 6 gives a matching upper bound, we have

Corollary 2. $c_\wedge(H^{2^k}) = 2^k - H^{\mathbb{N}}(2^k) = 2^k - 1$ *for all* $k \geq 0$.

We proceed to show that the bound in Theorem 6 is tight, and hence the construction is optimal for all n.[2] The proof uses the known value of $c_\wedge(H^{2^k})$ to compute a lower bound on $c_\wedge(H^{2^k-i})$. For notational brevity, we will denote $c_\wedge(H^n)$ by h_n.

Theorem 7. $c_\wedge(H^n) = n - H^{\mathbb{N}}(n)$, *for all* $n \geq 1$.

Proof. By Corollary 2, we only need to consider the cases where n is strictly between consecutive powers of 2, i.e. $2^{k-1} < n < 2^k$. Our proof is by induction on k with base $k = 1$. Let $k > 1$ and assume the theorem holds for all $n' \leq 2^{k-1}$. Let $n = 2^k - i$ for some integer $1 \leq i < 2^{k-1}$. Then $n + (i - 1) = 2^k - 1$ implies, by observation (5.3),

[2] This is quite surprising. In fact, we mistakenly stated in [8] that this bound was not tight.

that $k - H^{\mathbb{N}}(i-1) = H^{\mathbb{N}}(n)$. We design a circuit for the Hamming weight of a string $\mathbf{x}$ of length $2^k = n + (i-1) + 1$ as follows. We split $\mathbf{x}$ into three strings $\mathbf{u}, \mathbf{v}, c$ of lengths $n, i-1$, and 1, respectively. We use optimal circuits to compute $\overrightarrow{H}(\mathbf{u})$ and $\overrightarrow{H}(\mathbf{v})$. Note that the longest of these two strings is $\overrightarrow{H}(\mathbf{u})$, which has length k. Then we use the standard addition circuit with carry-in c to compute $c + \overrightarrow{H}(\mathbf{u}) + \overrightarrow{H}(\mathbf{v})$. The result is $\overrightarrow{H}(\mathbf{x})$. By the inductive hypothesis, the circuit for $\overrightarrow{H}(\mathbf{v})$ contains $h_{i-1} = (i-1) - H^{\mathbb{N}}(i-1)$ multiplications. Thus the circuit for $\overrightarrow{H}(\mathbf{x})$ contains $h_n + (i-1) - H^{\mathbb{N}}(i-1) + k$ multiplications. By Corollary 2, this quantity must be at least $2^k - 1$, i.e.

$$h_n + (i-1) - H^{\mathbb{N}}(i-1) + k \geq 2^k - 1.$$

Substituting $H^{\mathbb{N}}(n)$ for $k - H^{\mathbb{N}}(i-1)$, n for $2^k - i$, and rearranging terms, we obtain $h_n \geq n - H^{\mathbb{N}}(n)$. This proves the theorem since the lower bound matches the upper bound of Theorem 6. □

6. Truncated Hamming weight

Lemmas 1 and 2 together imply that one can compute $\Sigma_k^n(\mathbf{x})$ by computing the low-order $\lceil \log_2(k+1) \rceil$ bits of the Hamming weight of $\mathbf{x}$ and using AND gates to combine the appropriate resulting outputs, those corresponding to bits where the binary representation of k has a one.

Let H_r^n be the function which computes the r low-order bits of the Hamming weight of a vector of length $n \geq 2^{r-1}$. The complexity of this function is 0 when $r = 1$ and $n - H^{\mathbb{N}}(n)$ when $n \leq 2^r - 1$. In this section we show a recursive construction for the case $n \geq 2^r - 1$.

Lemma 10. *For $j \geq r \geq 1$, we have*

$$c_{\wedge}(H_r^{2^j-1}) \leq \left(\frac{2^{r-1}-1}{2^{r-1}}\right) 2^j - r + 1.$$

Proof. The proof is by induction on j. If $j < r$, only j bits of the Hamming weight need actually be computed; the high-order $r - j$ bits will be zero.

The result clearly holds for $j = 1$. For the inductive step, we consider $j > 1$, $n = 2^j - 1$, and the input bits $x_1, \ldots, x_n$. We split this input into three parts $(x_1, \ldots, x_{(n-1)/2})$, $(x_{(n+1)/2}, \ldots, x_{n-1})$, and the last bit x_n. The total Hamming weight of the input is the sum of the Hamming weights of all three parts. The r low order bits of the Hamming weights of the first two parts are computed recursively. Then, the sum is computed using a chain of $r-1$ *full adders*. A full adder is a circuit with three inputs a, b, c, and two outputs $(a + b + c)$ and $T_2^3(a, b, c)$. Instead of a full adder for the last bit, only $(a + b + c)$ is computed, since the carry is unnecessary. Clearly, each of the full adders requires exactly one AND gate. Let $h(n, r)$ be the multiplicative complexity of our construction. Then $c_{\wedge}(H_r^n) \leq h(n, r)$.

We feed x_n in as an external carry-bit to the chain of adders. The conjunctive complexity of our construction satisfies the following recurrence equation:

$$h(2^j - 1, r) = 2h(2^{j-1} - 1, r) + (r - 1),$$

giving

$$h(2^j - 1, r) \leq \left(\frac{2^{r-1}-1}{2^{r-1}}\right) 2^j - r + 1. \quad \square$$

We now obtain a similar result for arbitrary n.

Lemma 11. *Let $r \geq 1$ and $\gamma = n \bmod 2^r$. Then,*

$$c_{\wedge}(H_r^n) \leq \left(\frac{2^{r-1}-1}{2^{r-1}}\right)(n - \gamma) + \gamma - H^{\mathbb{N}}(\gamma).$$

Proof. Note that if $n \leq 2^r$, then the result follows from Theorem 6, so assume $n = 2^r m + \gamma$ and $2^r m = \sum_{i=1}^{b} 2^{u_i}$ with $u_i < u_{i+1}$ for $1 \leq i \leq b-1$. Then

$$n = \gamma + \sum_{i=1}^{b} 2^{u_i} = \gamma + b + \sum_{i=1}^{b} (2^{u_i} - 1).$$

Thus we can split the string of n bits into $b+2$ substrings of lengths $\gamma, b, 2^{u_1}-1, \ldots, 2^{u_b}-1$. Let $c_1, \ldots, c_b$ be the bits of the string of length b. We calculate the Hamming weight of the string of length γ using $\gamma - H^{\mathbb{N}}(\gamma)$ AND gates. Call the resulting string v. Then, for each string of length $2^{u_i}-1$, we use the circuit of Lemma 10 to calculate the low-order r bits of its Hamming weight. Call the resulting string s_i. Now, for i equal 1 through b, use $r-1$ full adders to compute the low-order r bits of $v = v + s_i + c_i$. The last value of v is H_r^n.

The number of AND gates used is

$$\gamma - H^{\mathbb{N}}(\gamma) + \sum_{i=1}^{b} \left(\left(\frac{2^{r-1}-1}{2^{r-1}} \right) 2^{u_i} - r + 1 + (r-1) \right),$$

which is

$$\gamma - H^{\mathbb{N}}(\gamma) + \left(\frac{2^{r-1}-1}{2^{r-1}} \right) \sum_{i=1}^{b} 2^{u_i}$$

or

$$\gamma - H^{\mathbb{N}}(\gamma) + \left(\frac{2^{r-1}-1}{2^{r-1}} \right) (n - \gamma). \quad \square$$

7. Building blocks

We now discuss subclasses of symmetric functions. The idea is to bound, as tightly as possible, the multiplicative complexity of classes of functions which can be used to construct arbitrary symmetric functions. We focus on three classes of functions:

- The elementary symmetric functions $\Sigma_k^n(\mathbf{x})$.
- The "counting" function $E_k^n(\mathbf{x})$, which is 1 if and only if the Hamming weight of $\mathbf{x}$ is k.
- The "threshold" function $T_k^n(\mathbf{x})$, which is 1 if and only if the Hamming weight of $\mathbf{x}$ is k or more.

It turns out Sierpinski's gasket (Pascal's Triangle modulo 2; see Fig. 3), from the fractals literature, is useful in this context.

7.1. The elementary symmetric functions: Σ_k^n

We first derive a general upper bound for the multiplicative complexity of Σ_k^n. Let $c_\wedge(f_1, \ldots, f_k)$ denote the multiplicative complexity of simultaneously computing $f_1, \ldots, f_k$. An immediate corollary of Lemmas 11 and 2 is the following:

Corollary 3. *Let $r \geq 1$, $n \geq 2^{r-1}$, and $\gamma = (n \bmod 2^r)$. Then*

$$c_\wedge(\Sigma_{2^0}^n, \ldots, \Sigma_{2^{r-1}}^n) \leq \left(\frac{2^{r-1}-1}{2^{r-1}} \right) (n - \gamma) + \gamma - H^{\mathbb{N}}(\gamma).$$

By Lemmas 1 and 2, the value of $\Sigma_k^n(\mathbf{x})$ is simply the GF_2 product of a subset of the low-order $\lceil \log_2(k+1) \rceil$ bits of the Hamming weight of $\mathbf{x}$. The number of terms in this product is $H^{\mathbb{N}}(k)$. Therefore Corollary 3 yields the following upper bound.

Theorem 8. *Let $n \geq k \geq 1$, and $r = \lceil \log_2(k+1) \rceil$. Let $\gamma = (n \bmod 2^r)$. Then,*

$$c_\wedge(\Sigma_k^n) \leq \left(\frac{2^{r-1}-1}{2^{r-1}} \right) (n - \gamma) + \gamma - H^{\mathbb{N}}(\gamma) + H^{\mathbb{N}}(k) - 1.$$

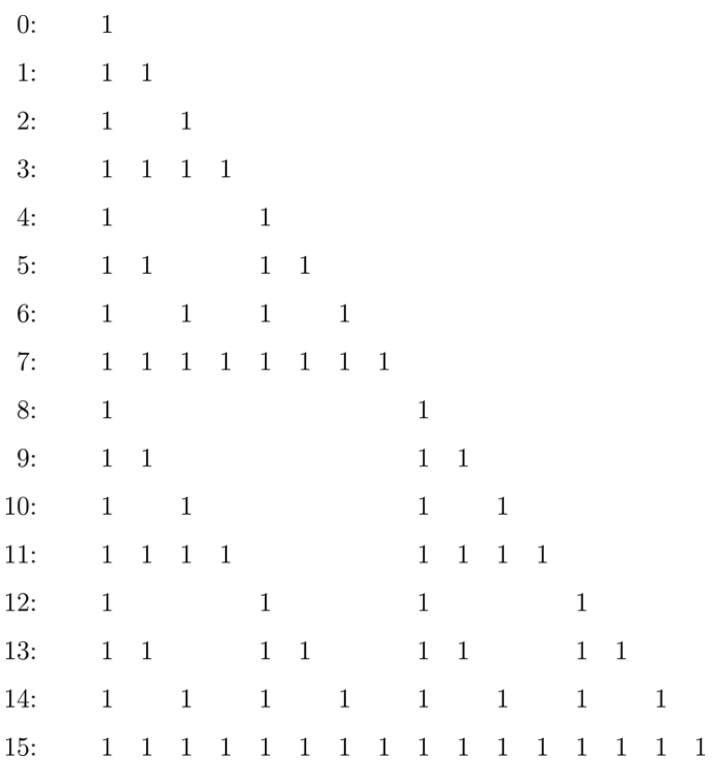

Fig. 3. Sierpinski's gasket.

Corollary 3 also yields the following:

Corollary 4. *For $n \geq 4$,*

$$c_\wedge(\Sigma_4^n) \leq c_\wedge(\Sigma_2^n, \Sigma_4^n) \leq \begin{cases} \frac{3}{4}n & : \quad n = 0 \bmod 4 \\ \frac{3}{4}(n-1) & : \quad n = 1 \bmod 4 \\ \frac{3}{4}\left(n - \frac{2}{3}\right) & : \quad n = 2 \bmod 4 \\ \frac{3}{4}\left(n - \frac{5}{3}\right) & : \quad n = 3 \bmod 4 \end{cases}.$$

For example, Corollary 4 yields the result $c_\wedge(\Sigma_4^5) = 3$, though this upper bound also follows from Theorem 6, since $\Sigma_4^5(\mathbf{x})$ is the high-order bit of $\overrightarrow{H}(\mathbf{x})$. We now consider the complexity of Σ_2^n. Recall (Theorem 2) that any nonlinear symmetric function has multiplicative complexity at least $\lfloor \frac{n}{2} \rfloor$. We also know $\Sigma_2^n(\mathbf{x})$ is the second least significant bit of $\overrightarrow{H}(\mathbf{x})$. This means we can use the truncated Hamming weight construction of Section 6, Lemma 11. The reader can verify that this construction yields a circuit with $\lfloor \frac{n}{2} \rfloor$ multiplications. We have shown

Theorem 9. *The multiplicative complexity of $\Sigma_2^n(\mathbf{x})$ is $\lfloor \frac{n}{2} \rfloor$.*

We now describe a second optimal circuit for Σ_2^n. For $i \leq j$, denote by $f[i..j]$ the function f evaluated on input $x_i, x_{i+1}, \ldots x_j$. The following recurrence is easy to verify:

$$\Sigma_2^n[1..n] = \Sigma_1^{n-1}[1..n-1] \cdot \Sigma_1^{n-1}[2..n] \oplus \Sigma_1^{n-2}[2..n-1] \oplus \Sigma_2^{n-2}[2..n-1].$$

Thus for $n \geq 4$, $c_\wedge(\Sigma_2^n) \leq c_\wedge(\Sigma_2^{n-2}) + 1$. Since $c_\wedge(\Sigma_2^2) = c_\wedge(\Sigma_2^3) = 1$, the above recurrence implies $c_\wedge(\Sigma_2^n) \leq \frac{n}{2}$. Since multiplicative complexities are integral, we have $c_\wedge(\Sigma_2^n) \leq \lfloor \frac{n}{2} \rfloor$.

Yet another optimal circuit for Σ_2^n can be constructed using the techniques of Aleksanyan [1] and of Mirwald and Schnorr [17] (after noting that the rank of the $n \times n$ matrix $\bar{I}$ over GF_2 is $n-1$ for n odd and n for even n).

Theorems 1 and 9 predict that, for odd n, Σ_2^n is either palindromic or anti-palindromic. It is easy to verify

Corollary 5. *Σ_2^{4k+1} is palindromic and Σ_2^{4k+3} is anti-palindromic.*

We now turn our attention to Σ_3^n. By Lemma 1, $\Sigma_3^n(\mathbf{x}) = \Sigma_2^n(\mathbf{x}) \cdot \Sigma_1^n(\mathbf{x})$. Thus, Theorem 9 implies the upper bound

$$c_\wedge(\Sigma_3^n) \leq \left\lfloor \frac{n}{2} \right\rfloor + 1.$$

However, this bound is not optimal. In fact, $c_\wedge(\Sigma_3^4) = 2$, as can be verified from

$$\Sigma_3^4(\mathbf{x}) = (x_1 \oplus x_2 \oplus x_3 \oplus x_4)((x_1 \oplus x_2)(x_1 \oplus x_3) \oplus x_1).$$

Theorem 1 implies

Corollary 6. *If $n \geq 3$ is odd, then $c_\wedge(\Sigma_3^n) \geq \frac{n+1}{2}$.*

Proof. The spectrum of $\Sigma_3^n(\mathbf{x})$ is the bit sequence $b_0 b_1 \ldots b_n$ where $b_i = 0$ for $i < 3$ and $b_i = \binom{i}{3} \bmod 2$ for $3 \leq i \leq n$. The spectrum of $\Sigma_3^n(\mathbf{x})$ starts with 000. If n is congruent to 1 modulo 4 the spectrum ends with 100. If n is congruent to 3 modulo 4 the spectrum ends with 001. Thus Σ_3^n is neither palindromic nor anti-palindromic. □

Now, for $n \geq 3$,

$$\begin{aligned}\Sigma_3^n &= x_n \Sigma_2^{n-1} \oplus \Sigma_3^{n-1} \\ &= x_n \Sigma_2^{n-1} \oplus \Sigma_1^{n-1} \Sigma_2^{n-1} \\ &= \Sigma_1^n \Sigma_2^{n-1}\end{aligned}$$

implies the upper bound $c_\wedge(\Sigma_3^n) \leq 1 + \left\lfloor \frac{n-1}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil$. For odd n, this upper bound matches the lower bound of Corollary 6. Note also that Σ_3^{n-1} is a restriction of Σ_3^n (set $x_n = 0$). Therefore $c_\wedge(\Sigma_3^{n-1}) \leq c_\wedge(\Sigma_3^n)$. For even n, this implies the lower bound $\left\lceil \frac{n-1}{2} \right\rceil = \frac{n}{2} \leq c_\wedge(\Sigma_3^n)$, which matches the previous upper bound of $\left\lceil \frac{n}{2} \right\rceil$. Thus we have shown

Theorem 10. $c_\wedge(\Sigma_3^n) = \left\lceil \frac{n}{2} \right\rceil$.

We now turn to Σ_m^n, where m is larger.
Lemma 1 implies

$$\Sigma_m^n = \Sigma_1^n \Sigma_{m-1}^n \qquad (m \text{ odd } 1 \leq m \leq n).$$

A simple lemma follows:

Lemma 12. *If m is odd and $1 \leq m \leq n$, then*

$$\Sigma_m^n = \Sigma_{m-1}^{n-1} \Sigma_1^n$$

and therefore $c_\wedge(\Sigma_m^n) \leq c_\wedge(\Sigma_{m-1}^{n-1}) + 1$.

Proof.

$$\begin{aligned}\Sigma_m^n &= x_n \Sigma_{m-1}^{n-1} \oplus \Sigma_m^{n-1} \\ &= x_n \Sigma_{m-1}^{n-1} \oplus \Sigma_{m-1}^{n-1} \Sigma_1^{n-1} \\ &= \Sigma_{m-1}^{n-1}(x_n \oplus \Sigma_1^{n-1}) \\ &= \Sigma_{m-1}^{n-1} \Sigma_1^n. \quad \square\end{aligned}$$

We can apply Lemma 12 to show $c_\wedge(\Sigma_5^8) = 5$ as follows. By Theorem 3, $c_\wedge(\Sigma_5^8) \geq 5$ and $c_\wedge(\Sigma_4^7) \geq 4$. Thus, by Corollary 4, $c_\wedge(\Sigma_4^7) = 4$. Therefore, $c_\wedge(\Sigma_5^8) = c_\wedge(\Sigma_4^7 \Sigma_1^8) \leq c_\wedge(\Sigma_4^7) + 1 = 5$.

In the following lemmas, we use our Hamming weight circuit construction to derive the exact multiplicative complexities of Σ_{n-1}^n, Σ_{n-2}^n, and Σ_{n-3}^n. The proofs will rely heavily on Lemmas 1, 2 and 12. In particular, Lemmas 1

and 2 imply that only $H^{\mathbb{N}}(m) - 1$ additional AND gates are needed to compute Σ^t_m from $\{\Sigma^t_{2^i} \mid 2^i \leq t\}$. We will also rely on the following simple observations regarding the Hamming weight of binary representations on integers:

$$\begin{aligned} H^{\mathbb{N}}(n) - H^{\mathbb{N}}(n-2) &= 1 \quad \text{when } (n \equiv 2 \bmod 4)\\ H^{\mathbb{N}}(n-1) - H^{\mathbb{N}}(n-4) &= 2 \quad \text{when } (n \equiv 0 \bmod 4)\\ H^{\mathbb{N}}(n) - H^{\mathbb{N}}(n-3) &= 2 \quad \text{when } (n \equiv 3 \bmod 4) \end{aligned}$$

The reader is also warned that the correctness of the algebraic manipulation we will be doing on elementary symmetric functions is dependent on the number of variables. For example, by Lemma 1, $\Sigma^n_{n-2} = \Sigma^n_{n-4}\Sigma^n_2$ is correct when $n \equiv 0 \bmod 4$, but not when $n \equiv 2 \bmod 4$.

Lemma 13. $c_\wedge(\Sigma^n_{n-1}) = n - 2$.

Proof. The degree lower bound implies $c_\wedge(\Sigma^n_{n-1}) \geq n - 2$. We will prove, by induction on n, that this is an upper bound as well. The theorem is trivially true for $n = 2$. For $n = 3$, the claim follows from $\Sigma^3_2 = (x_1 \oplus x_2)(x_1 \oplus x_3) \oplus x_1$.

We will consider separately the cases n even and n odd. For even n we have

$$\Sigma^n_{n-1} = \Sigma^{n-1}_{n-2}\Sigma^n_1.$$

By induction $c_\wedge(\Sigma^n_{n-1}) \leq n - 3 + 1 = n - 2$.

In the case of n odd we have

$$\begin{aligned} \Sigma^n_{n-1} &= x_n\Sigma^{n-1}_{n-2} \oplus \Sigma^{n-1}_{n-1}\\ &= x_n\Sigma^{n-2}_{n-3}\Sigma^{n-1}_1 \oplus x_{n-1}\Sigma^{n-2}_{n-2}\\ &= \Sigma^{n-2}_{n-3}(x_n\Sigma^{n-1}_1 \oplus x_{n-1}\Sigma^{n-2}_1)\\ &= \Sigma^{n-2}_{n-3}(x_n\Sigma^{n-1}_1 \oplus x_{n-1}\Sigma^{n-1}_1 \oplus x_{n-1})\\ &= \Sigma^{n-2}_{n-3}((x_n \oplus x_{n-1})\Sigma^{n-1}_1 \oplus x_{n-1}). \end{aligned}$$

By induction $c_\wedge(\Sigma^n_{n-1}) \leq n - 4 + 2 = n - 2$. □

Lemma 14. $c_\wedge(\Sigma^n_{n-2}) = n - 2$ *for* $n > 3$.

Proof. The lower bound follows from Theorem 3. For the upper bound, we first consider the case $n \equiv 2 \bmod 4$. In this case the circuit for $\Sigma^n_{n-2}(\mathbf{x})$ can be constructed as follows:

- compute the Hamming weight of $\mathbf{x}$. This uses $n - H^{\mathbb{N}}(n)$ AND gates;
- use the values $\{\Sigma^n_{2^i}(\mathbf{x}) \mid 2^i \leq n\}$ to compute $\Sigma^n_{n-2}(\mathbf{x})$. This uses $H^{\mathbb{N}}(n-2) - 1$ AND gates.

The number of AND gates of this circuit is

$$\begin{aligned} n - H^{\mathbb{N}}(n) + H^{\mathbb{N}}(n-2) - 1 &= n - (H^{\mathbb{N}}(n) - H^{\mathbb{N}}(n-2)) - 1\\ &= n - 2 \qquad (n \equiv 2 \bmod 4). \end{aligned}$$

We now consider the case $n \equiv 0 \bmod 4$. In this case we have

$$\begin{aligned} \Sigma^n_{n-2} &= x_n\Sigma^{n-1}_{n-3} \oplus \Sigma^{n-1}_{n-2}\\ &= x_n\Sigma^{n-1}_{n-4}\Sigma^{n-1}_1 \oplus \Sigma^{n-1}_{n-2}\\ &= \Sigma^{n-1}_{n-4}(x_n\Sigma^{n-1}_1 \oplus \Sigma^{n-1}_2). \end{aligned}$$

We can therefore construct the circuit as follows:

- Compute the Hamming weight of $(x_1, \ldots, x_{n-1})$. This uses $n - 1 - H^{\mathbb{N}}(n-1)$ AND gates.
- Use the values $\{\Sigma^{n-1}_{2^i}(\mathbf{x}) \mid 2^i \leq n-1\}$ to compute Σ^{n-1}_{n-4}. This uses $H^{\mathbb{N}}(n-4) - 1$ AND gates.
- Use two more AND gates to compute $\Sigma^{n-1}_{n-4}(x_n\Sigma^{n-1}_1 \oplus \Sigma^{n-1}_2)$.

The number of AND gates of this circuit is

$$\begin{aligned} n-1-H^{\mathbb{N}}(n-1)+H^{\mathbb{N}}(n-4)-1+2 &= n-(H^{\mathbb{N}}(n-1)-H^{\mathbb{N}}(n-4)) \\ &= n-2 \qquad (n\equiv 0 \bmod 4). \end{aligned}$$

Finally, if n is 1 or 3 modulo 4 then $n-2$ is odd and $n-3$ is either 0 or 2 modulo 4. Therefore

$$c_\wedge(\Sigma^n_{n-2}) = c_\wedge(\Sigma^{n-1}_{n-3}\Sigma^n_1) \le c_\wedge(\Sigma^{n-1}_{n-3})+1 \le n-3+1 = n-2. \quad \square$$

Lemma 15. *$c_\wedge(\Sigma^n_{n-3}) = n-3$ for $n>4$.*

Proof. The lower bound follows from Theorem 3. If n is even, then

$$c_\wedge(\Sigma^n_{n-3}) = c_\wedge(\Sigma^{n-1}_{n-4}\Sigma^n_1) \le c_\wedge(\Sigma^{n-1}_{n-4})+1$$

reduces the problem to the case of n odd.

If $n\equiv 3 \bmod 4$ then the Hamming weight circuit construction yields

$$c_\wedge(\Sigma^n_{n-3}) \le n-H^{\mathbb{N}}(n)+H^{\mathbb{N}}(n-3)-1 = n-(H^{\mathbb{N}}(n)+H^{\mathbb{N}}(n-3))-1 = n-3.$$

Finally, if $n\equiv 1 \bmod 4$,

$$\begin{aligned} \Sigma^n_{n-3} &= x_n\Sigma^{n-1}_{n-4}\oplus\Sigma^{n-1}_{n-3} \\ &= x_n\Sigma^{n-2}_{n-5}\Sigma^{n-1}_1\oplus x_{n-1}\Sigma^{n-2}_{n-4}\oplus\Sigma^{n-2}_{n-3} \\ &= \Sigma^{n-2}_{n-5}(x_n\Sigma^{n-1}_1\oplus x_{n-1}\Sigma^{n-2}_1\oplus\Sigma^{n-2}_2) \\ &= \Sigma^{n-2}_{n-5}(x_n\Sigma^{n-1}_1\oplus x_{n-1}\Sigma^{n-1}_1\oplus x_{n-1}\oplus\Sigma^{n-2}_2) \\ &= \Sigma^{n-2}_{n-5}((x_n\oplus x_{n-1})\Sigma^{n-1}_1\oplus x_{n-1}\oplus\Sigma^{n-2}_2). \end{aligned}$$

Observe that Σ^{n-1}_1 is linear and that a Hamming weight circuit construction for $(x_1,\ldots,x_{n-2})$ yields Σ^{n-2}_2. Therefore

$$\begin{aligned} c_\wedge(\Sigma^n_{n-3}) &\le (n-2)-H^{\mathbb{N}}(n-2)+H^{\mathbb{N}}(n-5)-1+2 \\ &= n-(H^{\mathbb{N}}(n-2)-H^{\mathbb{N}}(n-5))-1 \\ &= n-3. \quad \square \end{aligned}$$

7.2. *The exactly-k, $E^n_k(\mathbf{x})$, and threshold, $T^n_k(\mathbf{x})$, functions*

In this section, general results for the exactly-k and threshold-k functions are proved, expressing these functions in terms of the elementary symmetric functions and using these expressions to derive degree lower bounds. Exact complexities are derived for certain infinite subclasses.

The degree of $E^n_k = a_0\Sigma^n_0\oplus\cdots\oplus a_n\Sigma^n_n$ is the largest i such that a_i is nonzero. It is clear that $a_i=0$ for $i<k$. It turns out there is a simple formula for the remaining a_i.

Lemma 16. *$E^n_k = \bigoplus_{i=k}^n a_i\Sigma^n_i$, where $a_i = \binom{i}{k} \bmod 2$.*

Proof. By induction on i. If $\mathbf{x}$ has Hamming weight k, then $1 = E^n_k(\mathbf{x}) = a_k\Sigma^n_k(\mathbf{x})$ since higher-degree terms are 0. Thus $1 = a_k\Sigma^n_k(\mathbf{x}) = a_k$, which provides the basis for the induction. We use the identity $\binom{i}{k}\binom{l}{i} = \binom{l}{k}\binom{l-k}{i-k}$ (equation 5.21 in [13]). Assume that $a_i = \binom{i}{k} \bmod 2$ for $i=k,\ldots,l-1$. (In the following all binomial coefficients are reduced

modulo 2.) Consider $\mathbf{x}$ with Hamming weight $l > k$. Then $\Sigma_i^n(\mathbf{x}) = \binom{l}{i}$ for $k \le i \le l$ and $\Sigma_i^n(\mathbf{x}) = 0$ for $i > l$. Thus

$$\begin{aligned}
0 &= E_k^n(\mathbf{x}) \\
&= a_k \Sigma_k^n(\mathbf{x}) \oplus \cdots \oplus a_l \Sigma_l^n(\mathbf{x}) \\
&= \bigoplus_{i=k}^{l-1} \binom{i}{k}\binom{l}{i} \oplus a_l \qquad \text{(by the induction hypothesis)} \\
&= \bigoplus_{i=k}^{l-1} \binom{l}{k}\binom{l-k}{i-k} \oplus a_l \\
&= \binom{l}{k} \bigoplus_{i=k}^{l-1} \binom{l-k}{i-k} \oplus a_l \\
&= \binom{l}{k} \bigoplus_{i=0}^{l-1-k} \binom{l-k}{i} \oplus a_l \\
&= \binom{l}{k}(2^{l-k} - 1) \oplus a_l \\
&= \binom{l}{k} \oplus a_l \qquad \text{(since arithmetic is modulo 2)}
\end{aligned}$$

Thus $a_l = \binom{l}{k} \bmod 2$, completing the induction step. □

Lemma 16 implies that we can use Sierpinski's gasket (see Fig. 3) to compute the expansion of E_k^n. We have replaced 0's by blanks to highlight the fractal-like structure of the triangle. Rows and columns are numbered from 0 to 15. To use the figure, say, to compute E_6^{13} simply look at column 6 (the seventh actual column since column 0 is included), rows 6 to 13. The corresponding bit-array is 1 1 0 0 0 0 0 0. Therefore $E_6^{13} = \Sigma_6^{13} \oplus \Sigma_7^{13}$. Now, $\Sigma_6^{13} \oplus \Sigma_7^{13} = \Sigma_4^{13} \cdot \Sigma_2^{13} \cdot (1 \oplus \Sigma_1^{13})$. Thus $c_\wedge(E_6^{13}) \le c_\wedge(\Sigma_4^{13}, \Sigma_2^{13}) + 2$. By Corollary 4, $c_\wedge(\Sigma_4^{13}, \Sigma_2^{13}) \le 9$. Therefore $c_\wedge(E_6^{13}) \le 11$. This is quite remarkable given the general upper bound of $13 + 3\sqrt{13} > 23$ from [8] (or if one considers that the associated polynomial has over 18 thousand multiplications).

A similar lemma holds for the threshold functions since T_k^n can be expressed recursively using $T_k^n = x_n E_{k-1}^{n-1} \oplus T_k^{n-1}$, which says that at least k out of $x_1, \ldots, x_n$ are ones if and only if at least k out of $x_1, \ldots, x_{n-1}$ are ones or (exclusive) x_n is ones and exactly $k-1$ out of $x_1, \ldots, x_{n-1}$ are ones. This leads to the following characterization of the expansion of T_k^n based on Sierpinski's gasket:

Lemma 17. *$T_k^n = \bigoplus_{i=k}^n b_i \Sigma_i^n$ where $b_i = \binom{i-1}{k-1} \pmod 2$.*

Proof. Recall that all binomial coefficients below should be considered as being reduced modulo 2. Our proof is by induction on $n - k$. The base case is

$$T_n^n = \Sigma_n^n = \bigoplus_{i=n}^{n} \binom{i-1}{n-1} \Sigma_i^n.$$

Using the expansion of E_k^n, the proof is straightforward:

$$\begin{aligned}
T_k^n &= x_n E_{k-1}^{n-1} \oplus T_k^{n-1} \\
&= x_n \bigoplus_{i=k-1}^{n-1} \binom{i}{k-1} \Sigma_i^{n-1} \oplus \bigoplus_{i=k}^{n-1} \binom{i-1}{k-1} \Sigma_i^{n-1} \qquad \text{by induction} \\
&= \bigoplus_{i=k}^{n-1} \binom{i-1}{k-1} \left(x_n \Sigma_{i-1}^{n-1} \oplus \Sigma_i^{n-1}\right) \oplus \binom{n-1}{k-1} x_n \Sigma_{n-1}^{n-1} \\
&= \bigoplus_{i=k}^{n} \binom{i-1}{k-1} \Sigma_i^n. \quad \square
\end{aligned}$$

7.2.1. Lower bounds

Since $E_k^n(\mathbf{x}) = E_{n-k}^n(\bar{\mathbf{x}})$ $(1 \le k \le n)$, we have

$$c_\wedge(E_k^n) = c_\wedge(E_{n-k}^n) \quad (0 \le k \le n\,). \tag{7.1}$$

Then the degree lower bound yields $c_\wedge(E_k^n) \ge \max\{k-1, n-k-1\}$. Similarly, since $T_k^n(\mathbf{x}) = 1 \oplus T_{n-k+1}^n(\bar{\mathbf{x}})$ $(1 \le k \le n)$, we have

$$c_\wedge(T_k^n) = c_\wedge(T_{n-k+1}^n) \quad (1 \le k \le n\,), \tag{7.2}$$

and the degree lower bound yields $c_\wedge(T_k^n) \ge \max\{k-1, n-k\}$. Since $T_n^n = \Sigma_n^n$, we have $c_\wedge(T_1^n) = c_\wedge(T_n^n) = c_\wedge(\Sigma_n^n) = n-1$.

As mentioned above, the degree of E_k^n (or T_k^n) will be the largest value j such that the expansion of E_k^n (T_k^n) contains the term Σ_j^n. In the case of E_k^n this will be the largest $k \le j \le n$ such that the binomial coefficient $a_j = \binom{j}{k}$ is odd, and the case of T_k^n this will be the largest $k \le j \le n$ such that $b_j = \binom{j-1}{k-1}$ is odd. Thus, the degree of T_k^n is one more than the degree of E_{k-1}^{n-1}. Given this relation, we will only consider the degree of E_k^n.

A theorem by Kummer [15] shows that the binomial coefficient $\binom{j}{k}$ is odd if and only if $k \sqsubseteq j$, where the notation $k \sqsubseteq j$ means that if the binary representations of k and j are $k_s k_{s-1} \ldots k_1$ and $j_s j_{s-1} \ldots j_1$, respectively, then for each i such that $k_i = 1$, it also the case that $j_i = 1$. Hence the degree of E_k^n is the largest $k \le j \le n$ such that $k \sqsubseteq j$. For any $n \ge 2$, the binary representation of $j' = 2^{\lfloor \log_2 n \rfloor} - 1$ consists entirely of ones, so if $k \le j'$, then $k \sqsubseteq j'$, so j' is a lower bound on the degree of E_k^n. In addition, the value $l_{n,k}$ calculated by performing the bitwise OR of $n-k$ and k will be at most n and will have the property that $k \sqsubseteq l_{n,k}$, so $l_{n,k}$ is also a lower bound on E_k^n. This gives the following degree lower bounds on the multiplicative complexity of the exactly-k and threshold-k functions:

Theorem 11. *$c_\wedge(E_k^n) \ge \max\{k-1, n-k-1, 2^{\lfloor \log_2 n \rfloor} - 2, l_{n,k} - 1\}$ and $c_\wedge(T_k^n) \ge \max\{k-1, n-k, 2^{\lfloor \log_2 n \rfloor} - 1, l_{n-1,k-1}\}$, where $l_{n,k}$ is the number corresponding to the bitwise OR of the binary representations of $n-k$ and k.*

7.2.2. Upper bounds

We refer to a set of Boolean functions on n variables as a *basis*. We call a basis *complete* if any symmetric function can be expressed as a linear combination of functions in the basis. Examples of complete bases are $\{\Sigma_i^n \mid 0 \le i \le n\}$, and $\{E_i^n \mid 0 \le i \le n\}$. Define $A_m^q = \bigoplus_{i=m}^q \Sigma_i^n$ for $m \le q \le n$.[3] Then $\Sigma_n^n = A_n^n$ and $\Sigma_m^n = A_m^n \oplus A_{m+1}^n$ for $m < n$. Therefore, the basis $\{A_i^n \mid 0 \le i \le n\}$ is complete. We will prove upper bounds on the multiplicative complexity of several classes of functions by constructing circuits for functions in the class A_i^q with $0 \le i \le q \le n$.

Notice that the functions $A_0^0 = \Sigma_0^n = 1$ and $A_0^1 = 1 \oplus \Sigma_1^n$ are linear. We next use the truncated Hamming weight circuit of Section 6 to compute nonlinear functions of the form A_i^q.

Lemma 18. *Let $r \ge 1$ and $2^r - 1 \le n$. Assume the values of $\Sigma_{2^i}^n$ are known for $i = 0, \ldots, r-1$. Then $A_0^{2^r-1}$ can be computed using $r-1$ additional AND gates.*

Proof. The proof is by induction on r. The base case $r = 1$ follows from $A_0^1 = 1 \oplus \Sigma_1^n$: For notational convenience, let $t_r = 2^r - 1$. Note t_r satisfies the recursion $t_r = 2^{r-1} + t_{r-1}$. By Lemma 1, and the fact that $t_r < 2(2^{r-1})$, we have $A_{2^{r-1}}^{t_r} = A_0^{t_{r-1}} \Sigma_{2^{r-1}}^n$. Thus, for $r > 1$,

$$\begin{aligned} A_0^{t_r} &= A_0^{t_{r-1}} \oplus A_{2^{r-1}}^{t_r} \qquad (\text{since } 2^{r-1} = t_{r-1} + 1) \\ &= A_0^{t_{r-1}} \oplus A_0^{t_{r-1}} \Sigma_{2^{r-1}}^n \\ &= A_0^{t_{r-1}} (1 \oplus \Sigma_{2^{r-1}}^n). \end{aligned}$$

By the induction hypothesis, this can be computed using $1 + (r-2) = r-1$ additional AND gates. □

We note that the recursive construction in the proof of Lemma 18 also yields the following partial sums:

[3] Note that, in the notation A_m^q, the parameter n is implicit.

Corollary 7. *Let $r \geq 1$ and $2^r - 1 \leq n$. Assume the values of $\Sigma^n_{2^i}$ are known for $i = 0, \ldots, r-1$. Then the functions $A^{2^s-1}_0$ $(0 \leq s \leq r)$ can be simultaneously computed using at most $r-1$ additional AND gates.*

We view the set of functions $\{A^{2^s-1}_0 \mid 0 \leq s \leq r\} \cup \{\Sigma^n_{2^i} \mid i = 0, \ldots, r-1\}$ as a basis. The number of AND gates sufficient to compute any linear combination of functions in this basis is no more than $c_\wedge(H^n_r) + r - 1$.[4] We can expand the basis as follows:

Corollary 8. *Let $r \geq 0$ and $2^r - 1 \leq n$. Assume the values of $\Sigma^n_{2^i}$ are known for $i = 0, \ldots, r-1$. Then the basis $\{A^{2^s-1}_0 \mid 0 \leq s \leq r\} \cup \{A^{2^s-1}_m \mid 0 \leq s \leq r, m = 2^q, q < s\} \cup \{A^{2^s-1}_m \mid 0 \leq s \leq r, m = 2^q + 1, q < s\}$ can be computed using $r-1$ additional AND gates.*

Proof. Start with $\{A^{2^s-1}_0 \mid 0 \leq s \leq r\}$ from Corollary 7 and note

$$A^{2^s-1}_{2^q} = A^{2^q-1}_0 \oplus A^{2^s-1}_0$$
$$A^{2^s-1}_{2^q+1} = A^{2^s-1}_{2^q} \oplus \Sigma^n_{2^q}. \quad \square$$

We illustrate the power of these techniques with an example. For $n = 7$ variables, the Hamming weight construction yields the functions $\Sigma^7_{2^i}$ for $i = 0, 1, 2$. The cost of this construction is 4 AND gates. By Corollary 8, two additional AND gates are enough to compute the functions $A^7_m = \bigoplus_{i=m}^7 \Sigma^n_i$ for $m \in \{0, 1, 2, 3, 4, 5\}$. Two more AND gates are sufficient to compute $\Sigma^7_6 = \Sigma^7_4 \Sigma^7_2$ and $\Sigma^7_7 = \Sigma^7_6 \Sigma^7_1$. Since $A^7_6 = \Sigma^7_6 \oplus \Sigma^7_7$ and $A^7_7 = \Sigma^7_7$, we have constructed the complete basis $\{A^7_i \mid 0 \leq i \leq 7\}$ at a cost of (at most) 8 AND gates. This shows

Lemma 19. *Any symmetric function on 7 inputs has multiplicative complexity at most 8.*

We conclude this by pointing out one of several exact complexity results that can be derived using the above techniques. This particular result will later be used for determining the exact complexity of the majority function when the number of variables is a power of two.

Corollary 9. *Let $r \geq 1$, $n = 2^r - 1$, and $m = 2^{r-1}$. Then $c_\wedge(E^n_m) = n - 1$.*

Proof. The case $r = 1$ follows from $E^1_1 = x_1$. We now consider the case $r > 1$. Note that, when m is a power of 2, the binomial coefficients $\binom{i}{m}$ are odd for $i = m, \ldots, 2m-1$. Thus, by Lemma 16, we have $E^{2^r-1}_{2^{r-1}} = A^{2^r-1}_{2^{r-1}}$. Thus the lower bound $c_\wedge(E^{2^r-1}_{2^{r-1}}) \geq 2^r - 2$ follows from the degree lower bound. For the upper bound, first compute the Hamming weight of $x_1, \ldots, x_n$ using $c_\wedge(H^n) = n - H^{\mathbb{N}}(n) = (2^r - 1) - r$ AND gates. By Corollary 8, $r-1$ additional AND gates are sufficient to compute the basis $\{A^{2^s-1}_m \mid 0 \leq s \leq r, m = 2^q, q < s\}$. (The basis $\{A^{2^s-1}_m \mid 0 \leq s \leq r, m = 2^q + 1, q < s\}$ can also be computed with no additional AND gates, but it is not necessary to do so for this proof.)

Thus a total of $(2^r - 1) - r + r - 1 = 2^r - 2 = n - 1$ AND gates are sufficient to compute $E^{2^r-1}_{2^{r-1}} = A^{2^r-1}_{2^{r-1}}$. $\square$

7.3. Majority

The majority function, a special case of the threshold function, is of particular importance in applications of this theory (e.g. electronic voting protocols). The threshold-k function and the exactly-k function are related by the identity $T^n_m = x_n E^{n-1}_{m-1} \oplus T^{n-1}_m$. This will allow us to establish the exact complexity of the majority function when the number of variables is a power of two.

Theorem 12. *Let $n = 2^r$ and $m = 2^{r-1} + 1$. Then $c_\wedge(T^n_m) = n - 1$.*

[4] H^n_r is defined in Section 6.

Proof. From Lemma 17, we have $T_m^n = \bigoplus_{i=m}^n b_i \Sigma_i^n$ where $b_i = \binom{i-1}{m-1}$ (mod 2). Since $b_n = \binom{2^r-1}{2^{r-1}}$ (mod 2) = 1, the degree of T_m^n is n. By the degree lower bound, $c_\wedge(T_m^n) \geq n-1$. Similarly, $T_m^{n-1} = \bigoplus_{i=m}^{n-1} b_i \Sigma_i^{n-1}$ where $b_i = \binom{i-1}{m-1}$ (mod 2) = $\binom{i-1}{2^r-1}$ (mod 2). In the binomial coefficients $\binom{i-1}{2^r-1}$, the value of $i-1$ ranges between 2^{r-1} and 2^r-2. As observed earlier, all these coefficients are odd. Thus $T_m^{n-1} = \bigoplus_{i=m}^{n-1} \Sigma_i^{n-1} = A_m^{n-1}$.[5] For an upper bound we use the construction in the proof of Corollary 9. This construction computes E_{m-1}^{n-1} at a cost of $n-2$ AND gates. The construction first builds a basis which contains $A_m^{n-1} = T_m^{n-1}$. Therefore one additional AND gate is enough to compute $T_m^n = x_n E_{m-1}^{n-1} \oplus T_m^{n-1}$. The total number of AND gates is $n-2+1 = n-1$. □

We now consider the general case. We will prove two upper bounds for the complexity of T_m^{2m-1}, i.e. majority of $n = 2m-1$ inputs. The first result is most interesting for small m, the second is asymptotically better.

Theorem 13. $c_\wedge(T_m^{2m-1}) \leq 2^{\lceil \log_2 m \rceil} + m - \lceil \log_2 m \rceil - 2$ *for all* $m \geq 2$.

Proof. Let $r = \lceil \log_2(2m-1) \rceil = 1 + \lceil \log_2 m \rceil$. Let k be such that $2m-1+2k = 2^r-1$. Notice that $m+k = 2^{r-1}$ and $k < m$. We create $2k$ additional inputs, half of them set to 0 and the other half to 1. This converts the problem of computing T_m^{2m-1} to that of computing $T_{m+k}^{2m-1+2k}$ on the enlarged set of inputs. Notice that $T_{m+k}^{2m-1+2k} = T_{2^{r-1}}^{2^r-1} = \Sigma_{2^{r-1}}^{2^r-1}$.

The latter can be computed using our standard Hamming weight circuit using $2^r - 1 - r$ AND gates. The top level of this circuit groups the inputs in sets of three and computes the majority of each group using one AND gate per group (see the proofs of Lemma 9 and Theorem 6). In this case, however, $2k$ of the inputs have a fixed value. By ordering the inputs in such a way that k groups of three have two fixed values, we decrease the number of AND gates in the circuit by k. Thus, the complexity of this construction is $2^r - 1 - r - k = 2^{\lceil \log_2 m \rceil} + m - \lceil \log_2 m \rceil - 2$ AND gates. □

The above construction is optimal for $m = 2, 3, 4$, but is asymptotically worse than the general upper bound of $n + 3\sqrt{n}$ for any symmetric function. The following construction does worse for small m, but gives an asymptotic upper bound that is at most logarithmically higher than n.

Let $s = \lceil \log_2 m \rceil$. The key idea behind the above construction is to add some artificial inputs so that it is only necessary to calculate the high-order bit of some Hamming weight. An alternative is to ignore the artificial zero bits and simply add $k = 2^s - m$ to the Hamming weight of $\mathbf{x}$. Let $\overrightarrow{H}(\mathbf{x}) = u_s \dots u_0$ and express k in binary as $k_s \dots k_0$.[6] Let the sum of $u_s \dots u_0$ and $k_s \dots k_0$ be τ. Since $m \leq 2^s$ and τ is an integer, τ is bounded above by $2m - 1 + 2^s - m = 2^s + m - 1 < 2^{s+1}$. Thus we can write $\tau = t_s \dots t_0$ and the value of $T_m^{2m-1}(\mathbf{x})$ is simply t_s. The sum can be computed using s AND gates. The total cost of this construction is $2m - 1 - H^{\mathbb{N}}(2m-1) + \lceil \log_2 m \rceil$ AND gates. In the special case that m is a power of 2, the value k will be zero, so no gates are necessary to add it to the Hamming weight already computed. In this case, only $2m - 1 - H^{\mathbb{N}}(2m-1)$ AND gates are needed.

This technique can clearly be generalized to any threshold function: Consider T_m^n, let $s = \lceil \log_2(n+1) \rceil - 1$ and define $r = |2^s - m|$. First suppose that $m \leq 2^s$. Then, one can compute $T_m^n(\mathbf{x})$ as the high-order bit of $\overrightarrow{H}(\mathbf{x}) + r$, using $n - H^{\mathbb{N}}(n) + s$ AND gates. On the other hand, if $m > 2^s$, then $n - m + 1 \leq n - 2^s \leq 2^s$. Thus, one can use the equality $T_m^n(\mathbf{x}) = 1 \oplus T_{n-m+1}^n(\bar{\mathbf{x}})$, which holds for $1 \leq m \leq n$: Compute the Hamming weight of the complement of $\mathbf{x}$, add $r = 2^s - n + m - 1$ to the result, and return the complement of the high-order bit as the result. This gives the following:

Theorem 14. $c_\wedge(T_m^n) \leq n - H^{\mathbb{N}}(n) + \lceil \log_2(n+1) \rceil - 1$ *for all* $m \geq 1$.

As mentioned above, the result is not optimal for small n, but it is better than the known general upper bound for symmetric functions.

[5] The implicit set of variables in A_m^{n-1} is $x_1, \dots, x_{n-1}$.

[6] Do not be distracted by the fact that one or more of the most significant bits in $k_s \dots k_0$ are zero. This is irrelevant for the argument.

8. Conclusion and open problems

In general, circuit complexity is very difficult. Multiplicative complexity is another approach to circuit complexity which seems promising initially, given the number of functions for which the techniques presented here have given the exact multiplicative complexities. The paper has presented lower bound techniques based on linear algebra, the degrees of the polynomials representing the functions, and planar restrictions of functions. The upper bound techniques have been based on computing the Hamming weight optimally and on derivations based on identities involving the elementary symmetric functions.

Additional upper and lower bound techniques would be quite interesting, with constructive upper bounds being the more interesting, given their applicability to designing efficient circuits.

Appendix A.1 contains tables showing the multiplicative complexity of the elementary symmetric, threshold-k, and exactly-k functions for at most 8 variables. These tables leave two concrete open problems: Is $c_\wedge(\Sigma_4^8)$ equal to 5 or 6, and is $c_\wedge(E_4^8)$ equal to 6 or 7? If $c_\wedge(\Sigma_4^8) = 6$, then the multiplicative complexity of Σ_m^n is not monotonic in m. Thus, depending on the answer to the complexity of Σ_4^8, there may or may not be an "easy" answer to the open problem: Is the multiplicative complexity of Σ_m^n monotonic in m? The monotonic function $\left\lfloor \frac{n+m}{2} \right\rfloor - 1$ fits all known results for the multiplicative complexity of $c_\wedge(\Sigma_m^n)$ for $m \geq 2$. Thus, showing that $c_\wedge(\Sigma_4^8) = 6$ would also break this pattern. It is not obvious, however, how to compute Σ_4^8 without also computing the lower order bits of the Hamming weight and thus using 6 AND gates.

Appendix A.2 contains a list of those bounds, formulas and identities used in this paper, which may be useful in proving further results.

Acknowledgements

The very thorough review and useful suggestions of anonymous referees are gratefully acknowledged. The second author would like to thank Michael Fischer for many interesting conversations on the subject of multiplicative complexity. He would also like to thank the Department of Mathematics and Computer Science at the University of Southern Denmark (formerly Odense University) for several invitations during which some of this work was done.

The first author was partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT), and by the Danish Natural Science Research Council (SNF). Part of the second author's work was done at the Computer Science Department, Yale University, before this author joined NIST. While at Yale, this work was partially supported by NSF grant CCR-0081823.

Appendix. Tables

A.1. Results for Σ_k^n, T_k^n, E_k^n for $n \leq 8$

The tools derived in this paper are sufficient to establish the exact multiplicative complexity of a large class of symmetric functions. Tables A.1–A.3 show the multiplicative complexities of Σ_k^n, T_k^n and E_k^n for $3 \leq n \leq 8$. Note that we know the exact complexities of all the tabulated functions except for Σ_4^8 and E_4^8.

For the elementary symmetric functions, all the values of (n, k) in the table are such that $k \in \{0, 1, 2, 3, n-3, n-2, n-1\}$, except for the pair $(n = 8, k = 4)$, so the results all follow from the relevant theorems, showing the exact multiplicative complexities of Σ_2^n, Σ_3^n, Σ_{n-3}^n, Σ_{n-2}^n, Σ_{n-1}^n. The lower bound for Σ_4^8 follows from Theorem 4 and the upper bound from Corollary 4.

The degrees of the exactly-k and threshold-k functions can be calculated using Lemmas 16 and 17, respectively. Then, the lower bounds can be obtained from the degree lower bound or Theorem 3 combined with Lemma 16 or 17. The upper bounds are described below.

For the threshold-k functions with $n = 7$, Theorem 14 gives an upper bound of 6 for all m. Theorem 13 gives the exact upper bound for T_2^3, T_3^5, T_4^7, and Theorem 12 gives the exact upper bound for T_5^8. The upper bound for T_4^8 then follows from $c_\wedge(T_4^8) = c_\wedge(T_5^8)$ (see Eq. (7.2)).

The result that $c_\wedge(T_4^6) = 4$ follows from $T_4^6 = \Sigma_4^6$. Then, from Eq. (7.2), we know $c_\wedge(T_3^6) = c_\wedge(T_4^6) = 4$. Note that $T_3^6 = \Sigma_3^6 \oplus \Sigma_4^6$. If we tried to compute T_3^6 using the Hamming weight circuit, we would obtain Σ_4^6, Σ_2^6, Σ_1^6 at a

Table A.1
Complexity of Σ_i^n for $3 \le n \le 8$

Σ_i^n	i						
n	2	3	4	5	6	7	8
3	1	2	–	–	–	–	–
4	2	2	3	–	–	–	–
5	2	3	3	4	–	–	–
6	3	3	4	4	5	–	–
7	3	4	4	5	5	6	–
8	4	4	5–6	5	6	6	7

Table A.2
Complexity of the threshold function T_i^n for $3 \le n \le 8$

T_i^n	i							
n	1	2	3	4	5	6	7	8
3	2	1	2	–	–	–	–	–
4	3	3	3	3	–	–	–	–
5	4	3	3	3	4	–	–	–
6	5	5	4	4	5	5	–	–
7	6	5	6	4	6	5	6	–
8	7	7	7	7	7	7	7	7

Table A.3
Complexity of the counting function E_i^n for $3 \le n \le 8$

E_i^n	i								
n	0	1	2	3	4	5	6	7	8
3	2	2	2	2	–	–	–	–	–
4	3	2	2	2	3	–	–	–	–
5	4	4	3	3	4	4	–	–	–
6	5	4	5	3	5	4	5	–	–
7	6	6	6	6	6	6	6	6	–
8	7	6	6	6	6–7	6	6	6	7

cost of 4 AND gates. It would then be impossible to obtain $\Sigma_3^6 \oplus \Sigma_4^6$ without using an extra AND gate. This example shows that direct Hamming weight constructions do not always lead to optimal circuits.

The upper bounds for some of the threshold-k and exactly-k functions for $n = 8$ were derived by first using four AND gates to compute the Hamming weight of the first seven values $x_1, x_2, \ldots, x_7$. The three outputs from this computation are Σ_4^7, Σ_2^7, and Σ_1^7. One can use the following derivations to obtain circuits with low multiplicative complexity.

$$
\begin{aligned}
T_6^8 &= \Sigma_6^8 \oplus \Sigma_8^8 = x_8(\Sigma_7^7 \oplus \Sigma_5^7) \oplus \Sigma_6^7 = \Sigma_4^7(x_8(\Sigma_3^7 \oplus \Sigma_1^7) \oplus \Sigma_2^7) \\
&= \Sigma_4^7(x_8(\Sigma_2^7 \Sigma_1^7 \oplus \Sigma_1^7) \oplus \Sigma_2^7) \\
T_7^8 &= \Sigma_7^8 \oplus \Sigma_8^8 = x_8(\Sigma_6^7 \oplus \Sigma_7^7) \oplus \Sigma_7^7 = (\Sigma_4^7 \Sigma_2^7)(x_8(1 \oplus \Sigma_1^7) \oplus \Sigma_1^7) \\
E_6^8 &= \Sigma_6^8 \oplus \Sigma_7^8 = x_8 \Sigma_5^7 \oplus \Sigma_6^7 \oplus x_8 \Sigma_6^7 \oplus \Sigma_7^7 = \Sigma_4^7(x_8(\Sigma_1^7 \oplus \Sigma_2^7) \oplus \Sigma_2^7 \oplus \Sigma_3^7) \\
&= \Sigma_4^7((\Sigma_1^7 \oplus \Sigma_2^7)(x_8 \oplus \Sigma_2^7)) \\
E_5^8 &= \Sigma_5^8 \oplus \Sigma_7^8 = x_8 \Sigma_4^7 \oplus \Sigma_5^7 \oplus x_8 \Sigma_6^7 \oplus \Sigma_7^7 = x_8(\Sigma_4^7 \oplus \Sigma_6^7) \oplus \Sigma_1^7(\Sigma_4^7 \oplus \Sigma_6^7) \\
&= \Sigma_1^8(\Sigma_4^7 \oplus \Sigma_4^7 \Sigma_2^7) \\
E_4^8 &= \Sigma_4^8 \oplus \Sigma_5^8 \oplus \Sigma_6^8 \oplus \Sigma_7^8 = x_8(\Sigma_3^7 \oplus \Sigma_4^7 \oplus \Sigma_5^7 \oplus \Sigma_6^7) \oplus (\Sigma_4^7 \oplus \Sigma_5^7 \oplus \Sigma_6^7 \oplus \Sigma_7^7) \\
&= (x_8 \oplus \Sigma_4^7)(\Sigma_3^7 \oplus \Sigma_4^7 \oplus \Sigma_5^7 \oplus \Sigma_6^7) = (x_8 \oplus \Sigma_4^7)(\Sigma_2^7 \Sigma_1^7 \oplus \Sigma_4^7)(1 \oplus \Sigma_1^7 \oplus \Sigma_2^7).
\end{aligned}
$$

The other upper bounds on threshold-k functions which are not straightforward, or do not follow from others by the symmetry $c_\wedge(T_k^n) = c_\wedge(T_{n-k+1}^n)$, can be verified using the following derivations:

$$\begin{aligned}
T_5^7 &= \Sigma_5^7 \oplus \Sigma_6^7 \oplus \Sigma_7^7 = \Sigma_4^7(\Sigma_1^7 \oplus \Sigma_2^7 \oplus \Sigma_1^7\Sigma_2^7)\\
T_5^6 &= \Sigma_5^6 \oplus \Sigma_6^6 = \Sigma_4^6(\Sigma_1^6 \oplus \Sigma_2^6)\\
T_6^7 &= \Sigma_6^7\\
T_5^6 &= \Sigma_5^6 \oplus \Sigma_6^6 = \Sigma_4^6(\Sigma_1^6 \oplus \Sigma_2^6)\\
T_4^5 &= \Sigma_4^5\\
T_3^4 &= \Sigma_3^4 \oplus \Sigma_4^4 = x_4(\Sigma_2^3 \oplus \Sigma_3^3) \oplus \Sigma_3^3 = x_4(\Sigma_2^3 \oplus \Sigma_2^3\Sigma_1^3) \oplus \Sigma_2^3\Sigma_1^3\\
&= \Sigma_2^3(x_4 \oplus x_4\Sigma_1^3 \oplus \Sigma_1^3).
\end{aligned}$$

Finally, the remaining less trivial upper bounds on the exactly-k functions can be verified using the following:

$$\begin{aligned}
E_7^8 &= \Sigma_7^8\\
E_4^7 &= \Sigma_4^7 \oplus \Sigma_5^7 \oplus \Sigma_6^7 \oplus \Sigma_7^7 = \Sigma_4^7(1 \oplus \Sigma_1^7 \oplus \Sigma_2^7 \oplus \Sigma_1^7\Sigma_2^7)\\
E_5^7 &= \Sigma_5^7 \oplus \Sigma_7^7 = \Sigma_5^7(1 \oplus \Sigma_2^7)\\
E_6^7 &= \Sigma_6^7 \oplus \Sigma_7^7 = \Sigma_6^7(1 \oplus \Sigma_1^7)\\
E_3^6 &= \Sigma_3^6\\
E_4^6 &= \Sigma_4^6 \oplus \Sigma_5^6 \oplus \Sigma_6^6 = \Sigma_4^6(1 \oplus \Sigma_1^6 \oplus \Sigma_2^6)\\
E_5^6 &= \Sigma_5^6\\
E_3^5 &= \Sigma_3^5\\
E_4^5 &= \Sigma_4^5 \oplus \Sigma_5^5 = \Sigma_4^5(1 \oplus \Sigma_1^5)\\
E_2^4 &= (T_2^3 \oplus x_4)(1 \oplus \Sigma_1^4)\\
E_3^4 &= \Sigma_3^4\\
E_2^3 &= \Sigma_2^3 \oplus \Sigma_3^3 = \Sigma_2^3(1 \oplus \Sigma_1^3).
\end{aligned}$$

A.2. *Exact complexities, bounds and identities*

In the following

- f refers to an arbitrary symmetric function on n variables;
- S refers to an arbitrary set of symmetric functions on n variables;
- H^n is the Hamming weight function on n variables;
- $H^{\mathbb{N}}(n)$ is the Hamming weight of the binary representation of the integer n.
- $\delta(f)$ is the degree of f.

The following partial list of results is provided for easy reference.
Nontrivial exact complexities

$$\begin{aligned}
c_\wedge(H^n) &= n - H^{\mathbb{N}}(n).\\
c_\wedge(\Sigma_2^n) &= \left\lfloor \frac{n}{2} \right\rfloor.\\
c_\wedge(\Sigma_3^n) &= \left\lceil \frac{n}{2} \right\rceil.\\
c_\wedge(\Sigma_{n-1}^n) &= n - 2.\\
c_\wedge(\Sigma_{n-2}^n) &= n - 2 \quad \text{for } n > 3.
\end{aligned}$$

$$c_\wedge(\Sigma^n_{n-3}) = n-3 \quad \text{for } n > 4.$$
$$c_\wedge(E^{2^r-1}_{2^{r-1}}) = 2^r - 2.$$
$$c_\wedge(T^{2^r}_{2^{r-1}+1}) = 2^r - 1.$$

Bounds and identities

$$c_\wedge(f) \geq \delta(f) \quad \text{if } 1 < \delta(f) < n-1.$$
$$c_\wedge(f) \leq n + 3\sqrt{n}.$$
$$c_\wedge(S) \leq 2n - \log_2 n.$$
$$\Sigma^n_k = \Sigma^n_{2^{i_0}} \Sigma^n_{2^{i_1}} \cdots \Sigma^n_{2^{i_j}} \qquad (k = 2^{i_0} + 2^{i_1} + \cdots + 2^{i_j}).$$
$$\Sigma^n_{2^i}(\mathbf{x}) = \text{the } (i+1)^{th} \text{ l.s.b. of } H^n(\mathbf{x}).$$
$$\Sigma^n_m = x_n \Sigma^{n-1}_{m-1} \oplus \Sigma^{n-1}_m \quad \text{for } 1 < m \leq n-1.$$
$$\Sigma^n_m = \Sigma^{n-1}_{m-1} \Sigma^n_1 \quad \text{for } m \text{ odd}.$$
$$c_\wedge(f) \geq \left\lfloor \frac{n}{2} \right\rfloor \quad \text{if } f \text{ is non-linear}.$$
$$E^n_k = \bigoplus_{i=k}^{n} a_i \Sigma^n_i \qquad \text{where } a_i = \tbinom{i}{k} \bmod 2.$$
$$T^n_k = \bigoplus_{i=k}^{n} b_i \Sigma^n_i \qquad \text{where } b_i = \tbinom{i-1}{k-1} \pmod 2.$$

Symmetries for equality and threshold functions

$$c_\wedge(E^n_k) = c_\wedge(E^n_{n-k}) \quad (0 \leq k \leq n\ ).$$
$$c_\wedge(T^n_k) = c_\wedge(T^n_{n-k+1}) \quad (1 \leq k \leq n\ ).$$

Let $l_{n,k}$ be the bitwise OR of $n-k$ and k. Then

$$c_\wedge(E^n_k) \geq \max\{k-1, n-k-1, 2^{\lfloor \log_2 n \rfloor} - 2, l_{n,k} - 1\}; \quad \text{and}$$
$$c_\wedge(T^n_k) \geq \max\{k-1, n-k, 2^{\lfloor \log_2 n \rfloor} - 1, l_{n-1,k-1}\}.$$

References

[1] A.A. Aleksanyan, On realization of quadratic Boolean functions by systems of linear equations, Cybernetics 25 (1) (1989) 9–17.
[2] G. Brassard, D. Chaum, C. Crépeau, Minimum disclosure proofs of knowledge, Journal of Computer and System Sciences 37 (1988) 156–189.
[3] P. Bürgisser, M. Clausen, M.A. Shokrollahi, Algebraic Complexity Theory, in: Grundlehren der Mathematischen Wissenschaften, vol. 315, Springer-Verlag, 1997.
[4] J. Boyar, I. Damgård, R. Peralta, Short non-interactive cryptographic proofs, Journal of Cryptology 13 (2000) 449–472.
[5] J. Boyar, M. Krentel, S. Kurtz, A discrete logarithm implementation of zero-knowledge blobs, Journal of Cryptology 2 (2) (1990) 63–76.
[6] J. Boyar, C. Lund, R. Peralta, On the communication complexity of zero-knowledge proofs, Journal of Cryptology 6 (2) (1993) 65–85.
[7] M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, in: Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, 1988, pp. 1–10.
[8] J. Boyar, R. Peralta, D. Pochuev, On the multiplicative complexity of Boolean functions over the basis $(\wedge, \oplus, 1)$, Theoretical Computer Science 235 (2000) 43–57.
[9] D. Chaum, C. Crépeau, I. Damgård, Multi-party unconditionally secure protocols, in: Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, 1988, pp. 11–19.
[10] R. Cramer, I. Damgård, J.B. Nielsen, Multiparty computation from threshold homomorphic encryption, in: Advances in Cryptology — EUROCRYPT 2001, in: Lecture Notes in Computer Science, vol. 2045, Springer-Verlag, 2001, pp. 280–300.
[11] P. Dunne, The complexity of Boolean networks, Academic Press, 1988.
[12] M. Fischer, R. Peralta, Counting predicates of conjunctive complexity one, Technical Report YALEU/DCS/TR1222, Yale University, December 2001.
[13] R. Graham, D. Knuth, O. Patashnik, Concrete Mathematics: A Foundation for Computer Science, second edn, Addison-Wesley, 1994, p. 114.
[14] M. Hirt, J.B. Nielsen, Upper bounds on the communication complexity of optimally resistent cryptographic multiparty computation, in: Advances in Cryptology — ASIACRYPT 2005, in: Lecture Notes in Computer Science, vol. 3788, Springer-Verlag, 2005, pp. 79–99.

[15] E.E. Kummer, Über die Ergänzungssätze zu den allgemeinen Reciprocitätsgesetzen, Journal für die Riene und Angewandte Mathematik 44 (1852) 93–146.
[16] M.V. Mihaĭljuk, On the complexity of calculating the elementary symmetric functions over finite fields, Sov. Math. Dokl. 20 (1979) 170–174.
[17] R. Mirwald, C. Schnorr, The multiplicative complexity of quadratic Boolean forms, Theoretical Computer Science 102 (2) (1992) 307–328.
[18] W.J. Paul, A 2.5n lower bound on the combinational complexity of Boolean functions, in: Proceedings of the 7th Annual ACM Symposium on the Theory of Computing, 1975, pp. 27–36.
[19] R. Rueppel, J. Massey, The knapsack as a nonlinear function, in: Abstracts of papers, IEEE Int. Symp. on Information Theory, 1985, p. 46.
[20] C.P. Schnorr, The multiplicative complexity of Boolean functions, in: Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 6th International Conference, in: Lecture Notes in Computer Science, vol. 357, 1989, pp. 45–58.
[21] L. Stockmeyer, On the combinational complexity of certain symmetric Boolean functions, Mathematical Systems Theory 10 (1977) 323–336.
[22] B.L. van der Waerden, Algebra. Frederick Ungar Publishing.