Concrete Conjunctive Complexity of Symmetric Functions

Research Report YALEU/DCS/TR1219

November 2001

Joan Boyar *[1] René Peralta **[2] Denis Pochuev ***[3]

[1] Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark (joan@imada.sdu.dk)

[2] Department of Computer Science
Yale University (peralta-rene@cs.yale.edu)

[3] (denis_pochuev@hp.com)

Abstract. The conjunctive complexity of a Boolean function f is defined as the minimum number of 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 conjunctive complexity of symmetric Boolean functions. New techniques that allow such exploration are introduced. Several upper and lower bounds on the conjunctive complexity of symmetric functions are given. In many cases, the upper and lower bounds meet, giving the exact conjunctive complexity of the function.

1 Introduction

Much research in Boolean circuit complexity is devoted to the following problem: Given a Boolean function and a supply of gates that perform certain basic operations, construct a circuit which computes the function and is optimal according to some criteria. Motivated by cryptographic applications [1], we consider Boolean circuits which contain only binary XOR and AND gates, plus unary NOT gates, and we attempt to minimize the number of AND gates. In [1], it is shown that so called "discreet" proofs of knowledge of circuit satisfiability exist the size of which (the proofs) is proportional to the number of AND gates in the circuit and independent of the number of XORs.

Thus, this work aims at determining the number of conjunctions necessary and sufficient to build a circuit for a function $f : GF_2^n \to GF_2$ over the basis $(\oplus, 1, \wedge)$. This complexity measure is called the "conjunctive complexity" of f, and is denoted by $c_\wedge(f)$. We focus mainly on symmetric functions, which are functions dependent only on the Hamming weight $H(\mathbf{x})$ of the input $\mathbf{x} \in GF_2^n$.

* Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT), and by the Danish Natural Science Research Council (SNF).

** Partially supported by NSF grant CCR-0081823

*** Partially supported by NSF grant CCR-9712109

1.1 Previous work

Not much is known about the conjunctive complexity of symmetric functions. In an earlier paper ([2]), we show the following results :

- An upper bound on the complexity $c_\wedge(H^n)$ of computing the binary representation of the Hamming weight of a string of n-bits: Let $\theta = (n \bmod 4)$, and let λ be the Hamming weight of the binary representation of $n-\theta$. Then $c_\wedge(H^n) \leq n - \lambda - \lceil \frac{\theta}{2} \rceil$;
- A general upper bound of $n + 3\sqrt{n}$ for any symmetric function f. This establishes a separation between Boolean and conjunctive complexity for symmetric functions. Paul [6] and Stockmeyer [7] have shown lower bounds of the form $2.5n - O(1)$ for the Boolean complexity of infinite families of symmetric functions;
- Let $\Sigma^n_{\cdot}$ be the set of symmetric predicates on n bits. We showed an upper bound of $2n - \log_2 n$ for the complexity $c_\wedge(\Sigma^n_{\cdot})$ of simultaneously computing all symmetric functions on n bits.[1]

1.2 Our results

Several new upper and lower bounds on the conjunctive complexity of symmetric functions are obtained.

A new technique, using a normal form for these circuits and elementary linear algebra, is used to show that any non-linear symmetric function on n variables has conjunctive complexity at least $\lfloor \frac{n}{2} \rfloor$. A completely different technique is used to prove a lower bound for the threshold function, $c_\wedge(T^n_k) \geq \max(2^{\lfloor \log_2 n \rfloor} - 1, k-1, n-k)$, where the first term is clearly the most interesting and is close to the general upper bound for any symmetric function.

More constructively, general techniques are found for proving upper bounds for elementary symmetric functions, and these, plus Pascal's triangle modulo 2 are used to prove upper bounds for the counting functions, $E^n_k(\mathbf{x})$, and the threshold functions, $T^n_k(\mathbf{x})$. These general techniques are shown to give many tight results.

The exact conjunctive complexities of the elementary symmetric functions, the counting functions, and the threshold functions are determined for $n \leq 5$.

2 Some simple observations and a normal form

There is a bijection between Boolean functions on n variables and polynomials over GF^n_2. A Boolean function $f(\mathbf{x})$ whose polynomial has degree d has conjunctive complexity at least $d-1$. This we call the *degree lower bound.*

We say that a circuit is optimal for f if it has $c_\wedge(f)$ conjunctions. 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

[1] The asymptotic result $c_\wedge(\Sigma^n_{\cdot}) = O(n)$ was obtained earlier by Mihaĭljuk [4]).

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 conjunctive complexity. Therefore we may restrict our study of conjunctive complexity to functions over the basis $(\oplus, \wedge)$. For technical reasons, and without affecting the conjunctive complexity of functions, we will allow

- unbounded fan-in of $\oplus$ gates;
- the operation $x \oplus 0$.

A function $f : GF_2^n \to GF_2$ is "linear" if it is either constant or a sum (over GF_2) of variables. That is, linear functions are precisely those functions having conjunctive complexity 0.[2] 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 $a + b, ab, a + 1$, respectively.

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

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

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 [8]). Consider, for example, the MAJORITY function $T_2^3(x_1, x_2, x_3)$. We have

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

The last equality establishes $c_\wedge(T_2^3) = 1$, and also serves to show that optimal circuits may not be easily derivable via algebraic manipulation.

[2] Linear functions are sometimes called "affine" in the literature.

The following lemma appears in [2]:

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

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

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 and f an *extension* of g.

3 A tight lower bound on the conjunctive complexity of symmetric functions

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

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

Proof.

Consider an optimal LNF circuit for f. Since the circuit has at most $\lfloor \frac{n-1}{2} \rfloor - 1$ conjunctions, the number of $\oplus$ gates is at most $k = 2(\lfloor \frac{n-1}{2} \rfloor - 1) + 1 \leq n - 2$ (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:

$$a_{ij} = 1 \text{ iff } x_i \text{ is an input to } \gamma_j.$$

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. $\square$

A slightly stronger result can be obtained for even n. The proof is analogous to the proof of lemma 2.

Lemma 3. *If n is even and the Boolean function f over GF_2^n has conjunctive complexity less than $\frac{n}{2}$, then f has a non-empty 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 4. *If a symmetric Boolean function f has a proper complementable set S, then f must be linear (i.e. $c_\wedge(f) = 0$).*

Suppose a non-linear symmetric function f has a non-empty complementable set S. Then, by lemma 4, S must be the set of all inputs, i.e. $f(x) = f(\bar{x})$. Symmetric functions which have the property $f(x) = f(\bar{x})$ are called "palindromic" (since the bit string $b_0, \ldots, b_n$ where b_i is the value of $f(\mathbf{x})$ when $\mathbf{x}$ has Hamming weight i, 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 $\mathbf{x}$, $f_{\bar{S}}(\mathbf{x}) = f(\mathbf{x}) + 1$. A symmetric function $f : GF_2^n \to GF_2$ is anti-palindromic if $H(\mathbf{x}) + H(\mathbf{y}) = n$ implies $f(\mathbf{x}) = f(\mathbf{y}) + 1$. 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 4:

Lemma 5. *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 non-linear f, let $g : GF_2^{n+1} \to GF_2$ be defined by $g(x_1, \ldots, x_{n+1}) = f(x_1, \ldots, x_n) + x_{n+1}$. Clearly, $c_\wedge(g) < \frac{n+1}{2}$. By lemma 3, g has a non-empty 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 non-linear, S cannot be a proper subset of $\{x_1, \ldots, x_n\}$ (by lemma 4). Therefore f must be palindromic.
- S contains x_{n+1}: then $f(\mathbf{x}) + x_{n+1} = f_{\bar{S}}(\mathbf{x}) + \bar{x}_{n+1} = f_{\bar{S}}(\mathbf{x}) + x_{n+1} + 1$. Therefore, $f(\mathbf{x}) = f_{\bar{S}}(\mathbf{x}) + 1$. Thus, $S - \{x_{n+1}\}$ is an anti-complementable set for f. Since f is non-linear, lemma 5 implies $S - \{x_{n+1}\} = \{x_1, \ldots, x_n\}$. Therefore f is anti-palindromic.

□

Theorem 1 is not vacuous: lemma 6 shows infinite classes of palindromic and anti-palindromic functions with conjunctive complexity $\frac{n-1}{2}$.

We can now prove the main result of this section.

Theorem 2. *Let $f : GF_2^n \to GF_2$ be a non-linear symmetric function. Then f has conjunctive complexity at least $\lfloor \frac{n}{2} \rfloor$.*

Proof. Since f is non-linear, by lemma 4, f may not have a proper complementable set. For odd n, lemma 2 gives the stated bound. Now suppose, for a contradiction, that n is even and $c_\wedge(f) < \frac{n}{2}$.

Then f has a non-empty complementable set S by lemma 3. Therefore, by lemma 4, S must be the set of all inputs to f, i.e. $f(x) = f(\bar{x})$. 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 $n-1$ 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 non-linear. 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 conjunctive complexity at least $\lfloor \frac{n}{2} \rfloor$. □

Theorem 3 shows that the bound $c_\wedge(f) \geq \lfloor \frac{n}{2} \rfloor$ is tight for infinitely many functions.

4 Building blocks

We now discuss subclasses of symmetric functions. The idea is to bound, as tightly as possible, the conjunctive 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).

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:

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

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 conjunctive complexity is integral, we have shown $c_\wedge(\Sigma_2^n) \leq \lfloor \frac{n}{2} \rfloor$. [3] Since this matches the lower bound of theorem 2, we have shown

Theorem 3. *The conjunctive complexity of $\Sigma_2^n(\mathbf{x})$ is $\lfloor \frac{n}{2} \rfloor$.*

An alternative proof of theorem 3 can be constructed directly using the techniques of Mirwald and Schnorr 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 (see [5]).

[3] This upper bound is also implied by Schnorr et. al.'s upper bound of $\lfloor \frac{n}{2} \rfloor$ for the conjunctive complexity of binary quadratic forms [5].

By lemma 1, $\Sigma_3^n(\mathbf{x}) = \Sigma_2^n(\mathbf{x}) \cdot \Sigma_1^n(\mathbf{x})$. Thus, theorem 3 implies the upper bound

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

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

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

As evidence of the interrelation between the three classes of functions Σ_k^n, E_k^n, and T_k^n, we show how the formula for Σ_3^4 above can be arrived at by examining E_3^4. It is easy to verify that

$$E_{n-1}^n = \Sigma_{n-1}^n \qquad n \text{ even} \tag{1}$$

$$= \Sigma_{n-1}^n + \Sigma_n^n \qquad n \text{ odd}. \tag{2}$$

Now, a four-bit string $\mathbf{x}$ has exactly three 1's if and only if it has an odd number of 1's AND it has at least three 1's. But if it has an odd number of 1's, then it has at least three 1's if and only if at least two of x_1, x_2, x_3 are 1. Thus $\Sigma_3^4[1..4] = E_3^4[1..4] = \Sigma_1^4[1..4] \cdot T_2^3[1..3]$, which corresponds to the formula given.

Describing a circuit by a formula can become unwieldy. The conjunctive complexity of the function f corresponds to the minimum number of multiplications necessary to write a "straight-line program" that computes f. For example, an optimal circuit for E_2^4 is given by the program

$$\boxed{\begin{aligned} t &= x_1 + (x_3 + x_4)(x_1 + x_2 + x_3 + x_4) \\ E_2^4 &= (t + x_2)(x_1 + x_4) + x_3 + t \end{aligned}}$$

where t is a temporary variable. Thus $c_\wedge(E_2^4) = 2$ and not 3 as one is tempted to believe given

$$E_2^4 = \Sigma_2^4 + \Sigma_3^4 = \Sigma_2^4 + \Sigma_2^4\Sigma_1^4 = \Sigma_2^4(1 + \Sigma_1^4).$$

The following subsections contain some of our other findings regarding the conjunctive complexity of Σ, T and E.

4.1 Σ_k^n

We have already shown $c_\wedge(\Sigma_2^n) = \lfloor \frac{n}{2} \rfloor$. Theorem 1 then predicts that, for odd n, Σ_2^n is either palindromic or anti-palindromic. It is easy to verify

Lemma 6. *Σ_2^{4k+1} is palindromic and Σ_2^{4k+3} is anti-palindromic.*

Theorem 1 implies

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

Proof. If x has Hamming weight less than 3 then $\Sigma_3^n(x) = 0$. If x has Hamming weight 3 then $\Sigma_3^n(x) = 1$. If x has Hamming weight 4 then $\Sigma_3^n(x) = 0$. Thus Σ_3^n is neither palindromic nor anti-palindromic. □

Now, for $n \geq 3$,

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

implies the upper bound $c_\wedge(\Sigma_3^n) \leq 1 + \lfloor \frac{n-1}{2} \rfloor = \lceil \frac{n}{2} \rceil$. For odd n, this upper bound matches the lower bound of corollary 1. 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 $\lceil \frac{n-1}{2} \rceil = \frac{n}{2} \leq c_\wedge(\Sigma_3^n)$, which matches the previous upper bound of $\lceil \frac{n}{2} \rceil$. Thus we have shown

Theorem 4. $c_\wedge(\Sigma_3^n) = \lceil \frac{n}{2} \rceil$.

Using techniques and results from [2], one can prove upper bounds on the conjunctive complexity of Σ_k^n for any fixed k. It is shown in Lemma 11 in [2] that when $k = 2^i$ for some i, then the value of $\Sigma_k^n(\mathbf{x})$ is the $i+1$st bit of the Hamming weight of $\mathbf{x}$. Then, Lemma 1 above implies 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 1. Lemma 7 in [2] says that the Hamming weight of $2^i - 1$ bits can be computed using at most $2^i - (i+1)$ AND gates, and the proof shows how to do the computation. Here we modify the lemma and proof somewhat since we only need the low-order $r = \lceil \log_2(k+1) \rceil$ bits of the Hamming weight. Let H_r^n be the function which computes the r low-order bits of the Hamming weight of a vector of length n.

Lemma 7. *For $j \geq 1$ and $r \geq 1$, we have*

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

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 $x_1, ..., x_n$. 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 low order $\min(r,j)$ bits of the Hamming weights of the first two parts are computed recursively. Then, the sum is computed using a chain of $\min(r,j) - 1$ *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

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

giving

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

□

Modifying the proof of Theorem 8 in [2] somewhat, one can get a similar result for general n.

Lemma 8. *Let $n, r \geq 1$ and $\gamma = n \bmod 2^r$. Then,*

$$c_\wedge(H_r^n) \leq \left(\frac{2^{r-1} - 1}{2^{r-1}}\right)(n - \gamma) + s(n,r),$$

where $s(n,r)$ is the conjunctive complexity of computing the Hamming weight of a bit string of length $\gamma = n \bmod 2^r$, i.e. $s(n,r) = c_\wedge(H^\gamma)$.

Proof. This clearly holds for $r = 1$ and for $n < 2^r$, so we assume that $r \geq 2$ and $n \geq 2^r$. Let $n = 2^r m + \gamma$, where $0 \leq \gamma < 2^r$. Let $j = \lceil \log_2(n+1) \rceil$. There exists $b \in [1..j-r]$ such that, using the binary representation of n, we get

$$n = \gamma + \sum_{i=1}^{b} 2^{u_i} = \gamma + b + \sum_{i=1}^{b} (2^{u_i} - 1) \quad \text{with} \quad u_i < u_{i+1} \quad \text{for} \quad 1 \leq i \leq b-1.$$

Notice that u_1 is at least r. Let the string of n bits be $\boldsymbol{x} = (x_1, \ldots, x_n)$. Denote by c_1 through c_b the last b variables in $\boldsymbol{x}$. Our proof is constructive:

Truncated Hamming weight construction

1. Compute s_0, the bits of the Hamming weight of the first γ bits of $\boldsymbol{x}$.
2. For i equal 1 through b, compute s_i, the low-order r bits of $\sum_{\alpha_{i-1}}^{\alpha_i - 1} x_i$, where $\alpha_0 = \gamma + 1$ and $\alpha_i = \alpha_{i-1} + 2^{u_i} - 1$. Note that this is the low-order r bits of the Hamming weight of $2^{u_i} - 1$ variables.
3. Let $v = s_0$. For i equal 1 through b use $r-1$ binary adders to compute the low-order $r-1$ bits of $v = v + s_i + c_i$ and the carry from bit $r-1$. Then, compute bit r of v using $\oplus$ gates.

The algorithm clearly computes the low-order r bits of the Hamming weight of $\boldsymbol{x}$. The number of AND gates used is at most

- $s(n,r)$ at step 1.

- $\sum_{i=1}^{b}\left(\left(\frac{2^{r-1}-1}{2^{r-1}}\right)2^{u_i}-r+1\right)$ at step 2 (see Lemma 7).
- $\sum_{i=1}^{b}(r-1)$ at step 3.

Thus, the total number of AND gates is at most

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

□

Let $c_{\wedge}(f_1,\dots,f_k)$ denote the conjunctive complexity of simultaneously computing $f_1,\dots,f_k$. An immediate corollary of lemma 8 is

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

$$c_{\wedge}(\Sigma^n_{2^1},\dots,\Sigma^n_{2^{r-1}})\leq\left(\frac{2^{r-1}-1}{2^{r-1}}\right)(n-\gamma)+c_{\wedge}(H^{\gamma}).$$

As mentioned earlier, after computing the low-order $\lceil\log_2(k+1)\rceil$ bits of the Hamming weight of $\boldsymbol{x}$, the value of $\Sigma^n_k(\mathbf{x})$ is simply the conjunction of a subset of these bits. Thus, we have

Theorem 5. *Let $n,k\geq 1$, $j=\lceil\log_2(k+1)\rceil$, and l be the number of ones in the binary representation of k. Then,*

$$c_{\wedge}(\Sigma^n_k)\leq\left(\frac{2^{j-1}-1}{2^{j-1}}\right)(n-\gamma)+s(n,j)+l-1,$$

where $s(n,j)$ is the conjunctive complexity of computing the Hamming weight of $\gamma=n \bmod 2^j$. By Theorem 8 in [2], $s(n,j)\leq\gamma-\lambda-\lceil\theta/2\rceil$, where $\theta=\gamma \bmod 4$ and λ is the Hamming weight of the binary representation of $\gamma-\theta$.

As an example, since the value of $\Sigma^n_4(x_1,...,x_n)$ is the third bit of the binary representation of the Hamming weight of $\mathbf{x}$, in order to simultaneously compute $\Sigma^n_4(x_1,...,x_n)$ and $\Sigma^n_2(x_1,...,x_n)$, one can keep track of the low-order three bits of the Hamming weight of $\mathbf{x}$.

The following is an upper bound for $c_{\wedge}(\Sigma^n_4,\Sigma^n_2)$:

Corollary 3. *For $n\geq 4$,*

$$c_{\wedge}(\Sigma^n_4,\Sigma^n_2)\leq\begin{cases}\frac{3}{4}n & : & n=0 \bmod 4\\ \frac{3}{4}(n-1) & : & n=1 \bmod 4\\ \frac{3}{4}(n-\frac{2}{3}) & : & n=2 \bmod 4\\ \frac{3}{4}(n-\frac{5}{3}) & : & n=3 \bmod 4\end{cases}.$$

Proof. In this case, $j=3$, $l=1$, and $\gamma=n \bmod 8$. The cases where $n<2^j$ are $n=4,5,6,7$. Lemma 8 gives $c_{\wedge}(H^4_3)\leq 3$, $c_{\wedge}(H^5_3)\leq 3$, $c_{\wedge}(H^6_3)\leq 4$, and $c_{\wedge}(H^7_3)\leq 4$, so the result holds for $n<8$. For $n\geq 8$, Theorem 5 yields $c_{\wedge}(\Sigma^n_4)\leq\frac{3}{4}(n-\gamma)+s(n,3)=\frac{3}{4}(n-\gamma)+s(\gamma,3)$. Thus we need to bound $s(\gamma,3)$. Using Theorem 8 in [2], we have $s(0,3)=0$, $s(1,3)=0$, $s(2,3)=1$, $s(3,3)=1$, $s(4,3)=3$, $s(5,3)=3$, $s(6,3)=4$, and $s(7,3)=4$, giving the required result. □

Applying corollary 3 yields the non-trivial example $c_\wedge(\Sigma_4^5) = 3$.

The elementary symmetric functions will play an important role in the analysis of other symmetric functions. A useful observation at this point is the following

$$H(\mathbf{x}) = l \quad \Rightarrow \quad \Sigma_i^n(\mathbf{x}) = \begin{cases} 0 & i > l \\ \binom{l}{i} \bmod 2 & i \le l \end{cases} \tag{3}$$

4.2 $E_k^n(\mathbf{x})$

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

$$c_\wedge(E_k^n) = c_\wedge(E_{n-k}^n) \quad (0 \le k \le n\ ). \tag{4}$$

Then the degree lower bound yields $c_\wedge(E_k^n) \ge \max\{k-1, n-k-1\}$.

The degree of $E_k^n = a_0\Sigma_0^n + \dots a_n\Sigma_n^n$ is the largest i such that a_i is non-zero. It is clear that $a_i = 0$ for $i < k$. It turns out there is a simple formula for the remaining a_i.

Lemma 9. $E_k^n = a_k\Sigma_k^n + \dots a_n\Sigma_n^n$ *where* $a_i = \binom{i}{k} \bmod 2 \qquad (k \le i \le n)$.

Proof. By induction on i. If $\mathbf{x}$ has Hamming weight k, then $1 = E_k^n(\mathbf{x}) = a_k\Sigma_k^n(\mathbf{x})$ since higher-degree terms are 0. Thus $1 = a_k\Sigma_k^n(\mathbf{x}) = a_k$, which provides the basis for the induction. Now assume $a_i = \binom{i}{k} \bmod 2 \qquad (i = k, \dots, l-1)$. This time let $\mathbf{x}$ have Hamming weight $l > k$. Then

$$\begin{aligned}
0 &= E_k^n(\mathbf{x}) \\
&= a_k\Sigma_k^n(\mathbf{x}) + \dots + a_l\Sigma_l^n(\mathbf{x}) \\
&= \sum_{i=k}^{l-1} \binom{i}{k}\binom{l}{i} + a_l \qquad \text{(using the induction hypothesis and (3))} \\
&= \sum_{i=k}^{l-1} \binom{l}{k}\binom{l-k}{i-k} + a_l \\
&= \binom{l}{k}\sum_{i=k}^{l-1} \binom{l-k}{i-k} + a_l \\
&= \binom{l}{k}\sum_{i=0}^{l-1-k} \binom{l-k}{i} + a_l \\
&= \binom{l}{k}(2^{l-k}-1) + a_l \\
&= \binom{l}{k} + a_l \qquad \text{(since arithmetic is modulo 2)}
\end{aligned}$$

Thus $a_l = \binom{l}{k} \bmod 2$ completing the induction step. [4] □

[4] The identity $\binom{i}{k}\binom{l}{i} = \binom{l}{k}\binom{l-k}{i-k}$ can be argued as follows. To choose k elements from a set of size l, choose first a subset of size i and then choose k elements from this subset. Note that each set of size k can be chosen in $\binom{l-k}{i-k}$ ways.

Lemma 9 implies we can use Pascal's Triangle modulo 2 (see Fig. 1) 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 $E_6^{13} = \Sigma_6^{13} + \Sigma_7^{13}$. Now, $\Sigma_6^{13} + \Sigma_7^{13} = \Sigma_4^{13} \cdot \Sigma_2^{13} \cdot (1 + \Sigma_1^{13})$. Thus $c_\wedge(E_6^{13}) \leq c_\wedge(\Sigma_4^{13}, \Sigma_2^{13}) + 2$. By corollary 3, $c_\wedge(\Sigma_4^{13}, \Sigma_2^{13}) \leq 9$ Therefore $c_\wedge(E_6^{13}) \leq 11$ which is quite remarkable, given that the associated polynomial has over 18 thousand multiplications.

```
 0:  1
 1:  1 1
 2:  1   1
 3:  1 1 1 1
 4:  1       1
 5:  1 1     1 1
 6:  1   1   1   1
 7:  1 1 1 1 1 1 1 1
 8:  1               1
 9:  1 1             1 1
10:  1   1           1   1
11:  1 1 1 1         1 1 1 1
12:  1       1       1       1
13:  1 1     1 1     1 1     1 1
14:  1   1   1   1   1   1   1   1
15:  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Fig 1: Pascal's Triangle modulo 2

4.3 $T_k^n(\mathbf{x})$

Since $T_k^n(\mathbf{x}) = 1 + T_{n-k+1}^n(\bar{\mathbf{x}}) \quad (1 \leq k \leq n)$, we have

$$c_\wedge(T_k^n) = c_\wedge(T_{n-k+1}^n) \quad (1 \leq k \leq n\ ). \tag{5}$$

Then the degree lower bound yields $c_\wedge(T_k^n) \geq \max\{k-1, n-k\}$. Since $T_n^n = \Sigma_n^n$, we have $c_\wedge(T_1^n) = c_\wedge(T_n^n) = c_\wedge(\Sigma_n^n) = n-1$. We also note that T_k^n is the sum of E_i^n for $i = k$ to n. Therefore, Pascal's Triangle is also useful for finding the expansion of T_k^n.

For odd $k < n$ the decomposition of T_k^n into elementary symmetric functions includes the term Σ_{k+1}^n. This observation yields the slightly sharper lower bounds

$$c_\wedge(T_k^n) \geq k \quad (\text{ odd } \ k > \frac{n}{2}) \tag{6}$$

and

$$c_\wedge(T_k^n) \geq n-k+1 \quad (\ \textit{even} \ \ (n-k) \geq \frac{n}{2}). \tag{7}$$

When n is a power of 2 and $k > 0$, the following lemma implies a lower bound of $n-1$ for T_k^n.

Lemma 10. *If n is a power of 2, then the expansion of T_k^n contains Σ_n^n for all $k > 0$.*

Proof. Following (Graham-Knuth-Patashnik [3]) we denote by $\epsilon_p(n)$ the exponent of p in the factorization of n. It is easy to verify that (equation 4.24 in [3]) $\epsilon_2(n!) = n - H(n)$.[5] This implies $\epsilon_2(\binom{n}{i}) = H(i) + H(n-i) - H(n)$. Since n is a power of 2, the Hamming weight of n is 1. Thus $\epsilon_2(\binom{n}{i}) = H(i) + H(n-i) - 1 > 0$ for $0 < i < n$. This means $\binom{n}{i}$ is even. Evaluating T_k^n at $\mathbf{x}$ with Hamming weight n yields

$$1 = T_k^n(\mathbf{x}) = a_n + \sum_{i=k}^{n-1} a_i \binom{n}{i} \bmod 2 = a_n.$$

□

Each coefficient in the expansion of T_k^n turns out to be a sum of binomial coefficients.

Lemma 11. $T_k^n = a_k \Sigma_k^n + \ldots + a_n \Sigma_n^n$ *where* $a_i = \sum_{l=k}^{i} \binom{i}{l} \bmod 2$.

Proof. Since $T_k^n = E_k^n + \ldots + E_n^n$, it follows that a_i is the sum of the coefficients of Σ_i^n in $E_k^n, \ldots, E_i^n$. By Lemma 9, this sum is $\sum_{l=k}^{i} \binom{i}{l} \bmod 2$. □

Now it is easy to show the following

Lemma 12. *For all n, if the expansion of T_k^n contains Σ_m^n, then the expansion of T_k^{n+1} contains Σ_m^{n+1}.*

Proof. Lemma 11 shows that the coefficient of Σ_m^n in T_k^n is independent of n provided $k \le m \le n$.

□

Lemmas 10 and 12 allow us to sharpen our lower bound to

Corollary 4.

$$c_\wedge(T_k^n) \ge \max\{k-1, n-k, 2^{\lfloor \log_2 n \rfloor} - 1\} \tag{8}$$

Proof. Let $m = 2^{\lfloor \log_2 n \rfloor}$. If $k > m$, then there is nothing to prove. Otherwise it is sufficient to prove that the expansion of T_k^n contains Σ_m^n. This can be argued as follows:

- the expansion of T_k^m contains Σ_m^m (since m is a power of 2);
- by Lemma 12, the expansion of T_k^n contains Σ_m^n.

□

For small values of n, we know the exact conjunctive complexity of T_k^n for $2 \le k \le \lceil \frac{n}{2} \rceil$:

- We have already seen that $c_\wedge(T_2^3) = 1$.

[5] We extend the notation $H(\mathbf{x})$ to integer arguments in the natural way. That is, $H(n)$ is the Hamming weight of the binary representation of n.

– $T_2^4 = \Sigma_2^4 + \Sigma_4^4$ and therefore $c_\wedge(T_2^4) \geq 3$ (as predicted by equation 7). The following program shows that $c_\wedge(T_2^4) = 3$:

$$\begin{aligned} t &= (x_1 + x_2 + x_4) \cdot (x_1 + x_2 + x_3) + x_1 \\ T_2^4 &= t \cdot x_1 \cdot x_2 + t + x_2 \end{aligned}$$

– by 7, $c_\wedge(T_3^5) \geq 3$. The following program shows that $c_\wedge(T_3^5) = 3$:

$$\begin{aligned} t_1 &= (x_0 + x_3) + (x_3 + x_4) \cdot (x_0 + x_4) \\ t_2 &= (x_1 + x_2) \cdot (x_0 + x_1 + x_3 + x_4) + x_0 + x_2 \\ T_3^5 &= t_1 \cdot t_2 + x_0 \end{aligned}$$

– by 5, $c_\wedge(T_2^5) = c_\wedge(T_4^5)$. Since $T_4^5 = \Sigma_4^5$, we have $c_\wedge(T_2^5) \geq 3$. The following program shows that $c_\wedge(T_2^5) = 3$:

$$\begin{aligned} t_1 &= x_1 + (x_1 + x_3) \cdot (x_1 + x_4) \\ t_2 &= x_2 + (x_1 + x_2 + x_3 + x4) \cdot (x_0 + x_2) \\ T_2^5 &= t_1 \cdot t_2 + t_1 + t_2 \end{aligned}$$

4.4 Results for Σ_k^n, T_k^n, E_k^n for $n \leq 5$

$n = 3$						
i	$c_\wedge(\Sigma_i^3)$	δ	$c_\wedge(T_i^3)$	δ	$c_\wedge(E_i^3)$	δ
0	0	0	0	0	2	3
1	0	1	2	3	2	3
2	1	2	1	2	2	3
3	2	3	2	3	2	3

$n = 4$						
i	$c_\wedge(\Sigma_i^4)$	δ	$c_\wedge(T_i^4)$	δ	$c_\wedge(E_i^4)$	δ
0	0	0	0	0	3	4
1	0	1	3	4	2	3
2	2	2	3	4	2	3
3	2	3	3	4	2	3
4	3	4	3	4	3	4

$n = 5$						
i	$c_\wedge(\Sigma_i^5)$	δ	$c_\wedge(T_i^5)$	δ	$c_\wedge(E_i^5)$	δ
0	0	0	0	0	4	5
1	0	1	4	5	4	5
2	2	2	3	4	3	3
3	3	3	3	4	3	3
4	3	4	3	4	4	5
5	4	5	4	5	4	5

Table 1. Exact conjunctive complexities of Σ_k^n, T_k^n, E_k^n for $n \leq 5$

The tools derived in this paper are sufficient to establish the exact conjunctive complexity of Σ_k^n, T_k^n and E_k^n for small n. Up to $n = 5$, the theoretical upper bounds matched the theoretical lower bounds for all functions except T_2^4, T_3^5, T_2^5. For these functions, straight-line programs matching the theoretical lower bounds were found using heuristic techniques. Even for such small n, significant computational resources were required to find optimal solutions. The exact complexities are given in Table 1. The degree of the associated polynomial is given in the columns with heading "δ". Note that the degree lower bound is met for all but a few of the functions ($\Sigma_2^4, \Sigma_2^5, \Sigma_3^5, E_2^5, E_3^5$). One would expect this to not be the case for higher values of n. Exact complexity tables for higher n is the subject of work in progress.

References

1. J. Boyar, I. Damgård, and R. Peralta. Short non-interactive cryptographic proofs. *Journal of Cryptology*, 13:449–472, 2000.
2. J. Boyar, R. Peralta, and D. Pochuev. On the multiplicative complexity of boolean functions over the basis $(\wedge, \oplus, 1)$. *Theoretical Computer Science*, 235:43–57, 2000.
3. R. Graham, D. Knuth, and O. Patashnik. *Concrete mathematics: a foundation for computer science*, page 114. Addison-Wesley, second edition, 1994.
4. M. V. Mihaĭljuk. On the complexity of calculating the elementary symmetric functions over finite fields. *Sov. Math. Dokl.*, 20:170–174, 1979.
5. R. Mirwald and C. Schnorr. The multiplicative complexity of quadratic boolean forms. *Theoretical Computer Science*, 102(2):307–328, 1992.
6. 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*, pages 27–36, 1975.
7. L. Stockmeyer. On the combinational complexity of certain symmetric boolean functions. *Mathematical Systems Theory*, 10:323–336, 1977.
8. B. L. van der Waerden. *Algebra.* Frederick Ungar Publishing.