Autosymmetric and D-reducible Functions: Theory and Application to Security

Anna Bernasconi[1], Valentina Ciriani[2], and Licia Monfrini[2]

No Institute Given

Summary. In this paper we study Boolean functions that exhibit two different XOR-based regularities (i.e., autosymmetry and D-reducibility) at the same time. XOR-based regularities can be exploited for the efficient computation of multiplicative complexity of a Boolean function f (i.e., the minimum number of AND gates that are necessary and sufficient to represent f over the basis {AND, XOR, NOT}). The multiplicative complexity is crucial in cryptography protocols such as zero-knowledge protocols and secure two-party computation, where processing AND gates is more expensive than processing XOR gates.

1 Introduction

The multiplicative complexity of a Boolean function f is defined as the minimum number of AND gates that are necessary and sufficient to represent f with a circuit, using the 2-input Boolean operators AND and XOR, and the negation (NOT). The basis {AND, XOR, NOT} is widely used to represent Boolean functions in cryptographic applications [7, 8, 14, 15, 16], where the multiplicative complexity plays a crucial role. In particular, the minimization of the number of AND gates is important for high-level cryptography protocols such as zero-knowledge protocols and secure two-party computation, where processing AND gates is more expensive than processing XOR gates [1]. Moreover, the multiplicative complexity is an indicator of the degree of vulnerability of the circuits, as a small number of AND gates in an {AND, XOR, NOT} circuit indicates a high vulnerability to algebraic attacks [8, 10, 16]. However, determining the multiplicative complexity of a Boolean function f is a computationally intractable problem [8]. Therefore, the minimization of the number of AND gates, in circuits composed by the gates {AND, XOR, NOT}, is important in order to estimate the multiplicative complexity of the function. For this purpose, Boolean functions can be represented exploiting *Xor-And-Inverter Graphs* (XAGs) [11, 14, 15], and the multiplicative complexity of an XAG implementation of a Boolean function can be used to provide an upper bound for its real multiplicative complexity.

The "regularities" of Boolean functions are often exploited for deriving, in shorter synthesis time, more compact circuits. In the literature, some structural regularities of Boolean functions have been studied, i.e., autosymmetry [5, 6, 13] and D-reducibility [4]. These regularities are based on the notion of affine spaces and are easily expressed using XOR gates. Thus, both these structural regularities can be exploited for decreasing the multiplicative complexity of an XAG, and to better estimate the multiplicative complexity of the function. In the literature [3] a study of the multiplicative complexity of autosymmetric functions and a study of the multiplicative complexity of D-reducible functions are proposed. Moreover, experimental results show that about the 9% of these regular functions are both autosymmetric and D-reducible.

In this paper, we further investigate on regular functions that are both autosymmetric and D-reducible. In particular, we give a formal characterization of completely specified autosymmetric and D-reducible functions. Moreover, we study the case of non-completely specified functions. Finally, we discuss the multiplicative complexity of functions that are both autosymmetric and D-reducible. The experimental results show that, for functions that are both autosymmetric and D-reducible, we get a better estimate of the multiplicative complexity in about 27% of the cases with respect to exploiting autosymmetry or D-reducibility only, with an average reduction of the number of ANDs of about 27%.

2 Preliminaries

In this section, we review the definitions and properties of autosymmetric and D-reducible functions and we introduce our running example. Finally, at the end of the section, we give a very brief introduction to multiplicative complexity and XOR-AND Graphs (XAG). Hereafter, we will consider Boolean functions over n variables (i.e., described in the Boolean space $\{0,1\}^n$).

2.1 Autosymmetric Functions

In this section, we introduce a particular regularity, i.e., autosymmetry [5, 6, 13], based on affine spaces.

Intuitively, a Boolean function f over n variables is *k-autosymmetric* if it can be projected onto a smaller function f_k that depends on $n-k$ variables. The regularity of a Boolean function f is then measured computing its *autosymmetry degree* k, with $0 \leq k \leq n$, where $k = 0$ means no regularity. For $k \geq 1$ the Boolean function f is said to be *autosymmetric*, and a new function f_k depending on $n-k$ variables only, called the *restriction* of f, is identified. Moreover, an expression for f can be simply built from f_k: $f(x_1, x_2, \ldots, x_n) = f_k(y_1, y_2, \ldots, y_{n-k})$, where f_k is a Boolean function on $n-k$ variables $y_1 = \oplus(X_1), y_2 = \oplus(X_2), \ldots, y_{n-k} = \oplus(X_{n-k})$ and each $\oplus(X_i)$

$x_1 = 0$

x_2x_3 \ x_4x_5	00	01	11	10
00	0	1	0	0
01	1	0	0	1
11	0	1	0	0
10	1	0	0	1

$x_1 = 1$

x_2x_3 \ x_4x_5	00	01	11	10
00	0	1	1	0
01	1	0	0	0
11	0	1	1	0
10	1	0	0	0

Fig. 1. Karnaugh map for the running example (function f), the colors highlight the autosymmetry regularity.

is a XOR whose input is a set of variables X_i with $X_i \subseteq \{x_1, x_2, \ldots, x_n\}$. Note that $\oplus(X_i)$ can be a single variable, i.e., $X_i = \{x_j\}$ and $\oplus(X_i) = x_j$. The autosymmetry test consists of finding the value of k, the restriction f_k, and each single XOR with its input variables X_i (reduction equations). Note that a degenerate function, i.e., a function that does not depend on all the variables, is autosymmetric. The computational time of the autosymmetry test is polynomial in the size of the ROBDD representation of f [5].

The restriction f_k is "equivalent" to, but smaller than f, and has $|S(f)|/2^k$ minterms only, where $S(f)$ denotes the support of f, and thus $|S(f)|$ is the number of minterms of f. Each point of f_k in $\{0,1\}^{n-k}$ corresponds to a set of 2^k points in $\{0,1\}^n$ where f assumes the same value. The function f can be synthesized through the synthesis of its restriction f_k. As the new $n-k$ variables are XOR combinations of some of the original ones, the reconstruction of f from f_k can be obtained with an additional logic level of XOR gates, whose inputs are the original variables, and the outputs are the new $n-k$ variables given as inputs to a circuit for f_k. In general, the restricted function f_k can be synthesized in any framework of logic minimization. In this paper we derive an XAG representation of it.

We now recall some properties of autosymmetric functions and of their restrictions, that will be useful for the analysis of their multiplicative complexity. As shown in [5, 6], any k-autosymmetric function f is associated to a k-dimensional vector space L_f, defined as the set of all minterms α s.t. $f(x) = f(x \oplus \alpha)$ for all $x \in \{0,1\}^n$. Let L_f be sorted in increasing binary order, with the vectors indexed from 0 to $2^k - 1$. The set of vectors of L_f with indices $2^0, 2^1, \ldots, 2^{k-1}$ is called the *canonical basis* B_L of L_f. The k variables that are truly independent onto L_f are called *canonical variables*, while the other variables are called *non-canonical*. Informally, the canonical variables are the ones that assume all the possible combinations of $\{0,1\}$ values in the vectors of the vector space L_f, meanwhile the non-canonical variables are the variables that, on L_f, have a constant value or are a linear combination of the canonical ones.

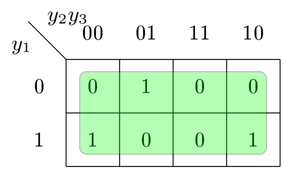

Fig. 2. Karnaugh map for the reduced function f_2 of the 2-autosymmetric function shown in Figure 1.

The canonical variables can be easily computed from the canonical basis $v_1, \ldots, v_k$, in the following way: for each v_i, let x be the variable corresponding to the first 1-component from left of v_i. The variable x is a *canonical variable*.

Finally, the restriction f_k corresponds to the projection of f onto the subspace $\{0,1\}^{n-k}$ where all the canonical variables assume value 0, while the reduction equations correspond to the linear combinations that define each non-canonical variable in terms of the canonical ones (see [5, 6] for more details).

Example 1. Given an arbitrary function f, the vector space L_f provides the essential information to compute the autosymmetry degree, the restriction f_k, and the reduction equations of f. Consider, for instance, the completely specified Boolean function $f(x_1, \ldots, x_5)$ described by its minterms as follows: $f = \{00001, 00100, 00110, 01000, 01010, 01101, 10001, 10011, 10100, 11000, 11101, 11111\}$. The function f can be represented by the Karnaugh map depicted in Figure 1. The "regularity" of the function is highlighted by the colors in the figure. The computation of the vector space L_f and of the reduction equations is not straightforward, we refer the reader to [5] for the complete algorithm. The vector space L_f associated to f is $L_f = \{00000, 01100, 10101, 11001\}$. In fact, for any element $\alpha \in L_f$ we have that $f(x) = f(x \oplus \alpha)$ for all $x \in \{0,1\}^n$. We have that $k = \log_2 |L_f| = 2$, thus f is 2-*autosymmetric*. The canonical basis is $B_V = \{01100, 10101\}$. The canonical variables are x_1 and x_2 (i.e., the variables that correspond to the first ones from left in the two vectors of the canonical base). The remaining variables x_3, x_4, and x_5 are non-canonical. The restriction f_2, depicted in Figure 2, can be computed starting from the subset of minterms $\{00001, 00100, 00110\}$ of f, where all the canonical variables are equal to 0. In fact, if we project these points in the space $\{0,1\}^3$, corresponding to the non-canonical variables x_3, x_4, and x_5, we get $f_2(y_1, y_2, y_3) = \{001, 100, 110\}$. Finally, the reduction equations for reconstructing the original function f are [5]: $y_1 = x_1 \oplus x_2 \oplus x_3; y_2 = x_4; y_3 = x_1 \oplus x_5$.

Autosymmetric functions are just a subset of all Boolean functions. Indeed, while the number of the Boolean functions of n variables is 2^{2^n}, the number of autosymmetric ones is $(2^n - 1)2^{2^{n-1}}$ [6]. Therefore, the set of autosymmetric functions is much smaller than the one containing all the Boolean functions. Nevertheless, a considerable amount of standard Boolean functions of practical interest falls in this class. Indeed, about 24% of the functions in the classical ESPRESSO benchmark suite [17] have at least one truly (i.e., non degenerate) autosymmetric output [5, 6]. Thus, the interest on autosymmetric functions is motivated by 1) their compact (in term of number of AND gates) representation, which consists of an XOR layer that is the input to an XAG for the restriction; 2) the frequency of autosymmetric functions in the set of benchmark functions.

2.2 D-Reducible Functions

In this section, we summarize the definitions and the major properties of Dimension Reducible Boolean functions, i.e., D-reducible functions. We recall that the Boolean space $\{0,1\}^n$ is a vector space with respect to the exclusive sum $\oplus$ and the multiplication with the scalars 0 and 1. Moreover, an affine space is a vector space or a translation of a vector space [4], more precisely: let V be vector subspace of the Boolean vector space $(\{0,1\}^n, \oplus)$ and w be a point in $\{0,1\}^n$, then the set $A = w \oplus V = \{w \oplus v \mid v \in V\}$ is an *affine space* over V with *translation point* w. The space V is called the *vector space associated* to A. Finally, a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *D-reducible* if $f \subseteq A$, where $A \subset \{0,1\}^n$ is an affine space of dimension strictly smaller than n.

The minimal affine space A containing a D-reducible function f is unique and it is called the *associated affine space* of f. The function f can be represented as $f = \chi_A \cdot f_A$, where $f_A \subseteq \{0,1\}^{\dim A}$ is the projection of f onto A and χ_A is the characteristic function of A. Observe that the smallest affine space contains the whole on-set of a function f. Thus, this regularity is different from autosymmetry, since the numbers of minterms of the original function f and of the projected function f_A are equal to each other. Moreover, as shown in [9], an affine space can be represented by a simple expression, consisting of an AND of XORs or literals. In particular, an affine space of dimension $\dim A$ can be represented by an expression containing $(n - \dim A)$ XOR factors.

The D-reducibility of a function f can be exploited in the minimization process. The projection f_A is minimized instead of f. This approach requires two steps: first, deriving the affine space A and the projection f_A, and then minimizing f_A in any logic framework (e.g., XAG). The D-reducibility test [4], which establishes whether a function f is D-reducible, and the computation of A can be performed efficiently exploiting the Gauss-Jordan elimination procedure [12], which is used to find the on-set minterms of f that are linearly independent.

Example 2. Let us consider the running example, analyzed for autosymmetry, i.e., the function f shown in Figure 3. The minimal affine space A containing all the minterms the function f is highlighted by the color cyan in the figure.

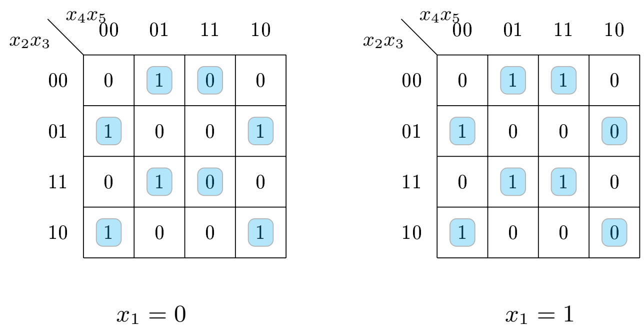

Fig. 3. Karnaugh map for the D-reducible function f. The space A of f is highlighted.

Thus, A is a 4-dimension affine space. The canonical basis of the vector space V associated to A is $\{00010, 00101, 01001, 10000\}$, its canonical variables are: x_1, x_2, x_3, and x_4, while x_5 is non-canonical. The representation, as an AND of XORs, of A is: $x_2 \oplus x_3 \oplus x_5$. Moreover, the projection of f onto the affine space A is $f_A = \{0000,\ 0010, 0011,\ 0100, 0101,\ 0110, 1000,\ 1001,\ 1010, 1100, 1110, 1111\}$. The projection f_A is represented in the Karnaugh map in Figure 4.

2.3 Multiplicative Complexity and XOR-AND Graphs

The *multiplicative complexity* $M(f)$ *of a Boolean function* f is a complexity measure defined as the number of AND gates, with fan-in 2, that are necessary and sufficient to implement f with a circuit over the basis {AND, XOR, NOT}. Moreover, the *multiplicative complexity* $M_C(f)$ *of a circuit* C implementing a Boolean function f over the basis {AND, XOR, NOT} is the actual number of AND gates in C. Therefore, the multiplicative complexity of a circuit for f only provides an upper bound for the multiplicative complexity of f, i.e., $M(f) \leq M_C(f)$. In this work, we consider Boolean functions represented in *XOR-AND graphs* (XAGs) form [11, 14, 15], which are logic networks that contain only binary XOR nodes, binary AND nodes, and inverters. In particular, we refer to the XAG model described in [14], where regular and complemented edges are used to connect the gates. Complemented edges indicate the inversion of the signals and replace inverters in the network.

3 Completely Specified Autosymmetric and D-reducible Functions

A Boolean function f, which is D-reducible and autosymmetric at the same time, can be decomposed in two different ways. The first possibility is to apply the D-reducibility decomposition, and represent f as $f = \chi_A f_A$, and then to

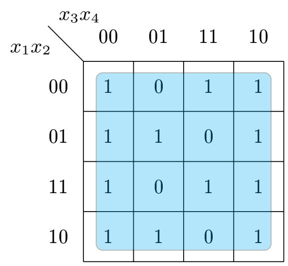

Fig. 4. Karnaugh map for the projection f_A of the D-reducible function f shown in Figure 3.

apply the autosymmetry reduction to f_A. The second possibility consists in decomposing the function f applying the autosymmetry test and deriving the restriction f_k, and then applying the D-reducibility decomposition to f_k. In this section, we prove that if f is a completely specified function, these two strategies provide the same final representation of the function f.

We first recall from [3] a theoretical result contained in the proof of a theorem, used to prove our results. For this reason, we report it as a lemma, and we recall here its proof.

Lemma 1. *[3] . Let f be an autosymmetric function with associated linear space L_f. Let f also be a D-reducible function contained in the affine space A. Then, $L_f \subseteq V$, where V is the vector space associated to A.*

Proof. First of all, we observe that the vector space L_f is a subspace of the vector space V associated to A. Let $\alpha \in L_f$, and let x be any on-set minterm of f. Then, $f(x \oplus \alpha) = f(x) = 1$, and therefore both x and $x \oplus \alpha \in A$. This in turns implies that $\alpha \in (x \oplus A)$, i.e., $\alpha \in V$, since $x \oplus A = V$ for any $x \in A$ (we refer the reader to [9] for more details on affine spaces and their properties). □

Example 3. Let us consider the function f described in Figures 1 and 3. In the previous examples we have shown that f is both autosymmetric and D-reducible. Example 1 shows that $L_f = \{00000, 01100, 10101, 11001\}$, and from the Figure 3 of Example 2 we have that $A = \{00001, 00011, 00100, 00110, 01000, 01010, 01101, 01111, 10001, 10011, 10100, 10110, 11000, 11010, 11101, 11111\}$. The corresponding vector space is computed as $V = v \oplus A$ where v is any vector contained in A. Thus, if we pick $v = 00001$ and computing $V = 00001 \oplus A$ we obtain: $V = \{00000, 00010, 00101, 00111, 01001, 01011, 01100, 01110, 10000, 10010, 10101, 10111, 11001, 11011, 11100, 11110\}$. (Notice that we can use any v in A and we would obtain the same associated vector V.) We can easily verify that $L_f \subseteq V$.

Let k denote the dimension of L_f and a be the dimension of the vector space V associated to A. The dimension of an affine space A is defined as the dimension of the corresponding vector space V.

Proposition 1. *The dimension of L_f is less or equal to the dimension of A, and the canonical variables of V include all the canonical variables of L_f.*

Proof. The first part of the proposition immediately follows from Lemma 1.

For the second part, observe that, since $L_f \in V$, we can construct a basis for V extending a basis for L_f. Each vector in a basis for L_f corresponds to a canonical variable of L_f, and consequently to a canonical variable of V. The remaining $a-k$ canonical variables of V can be derived from the remaining $a-k$ linearly independent vectors in the basis of V.

□

As a consequence, we have the following corollary.

Corollary 1. *The $n-k$ non-canonical variables of L_f include the $n-a$ non-canonical variables of V.*

Example 4. Let us consider the running example. Example 1 shows that the canonical variables of L_f are x_1 and x_2, and Example 2 shows that the canonical variables of the vector space V associated to A are x_1, x_2, x_3, and x_4. In this running example we have that the function is k-autosymmetric with $k = 2$, and that $a = 4$. Moreover, the non-canonical variables of L_f are x_3, x_4, and x_5. The non-canonical variable of the vector space V associated to A is x_5. We can verify that the $n - k = 5 - 2 = 3$ non-canonical variables of L_f contains the $n - a = 5 - 4 = 1$ non-canonical variable of V.

For completeness, we recall from [3] a theorem stating that if we first apply the D-reducibility decomposition, we do not loose the autosymmetry property of the function.

Theorem 1. *[3] Let f be a completely specified k-autosymmetric Boolean function depending on n binary variables. If f is D-reducible with associate affine space A, then the projection f_A of f onto A is k-autosymmetric.*

In order to prove that the two decomposition strategies provide the same final representation of f, we need to prove that the restriction f_k of an autosymmetric function preserves the D-reducibility property, as shown in the following theorem.

Theorem 2. *Let f be a D-reducible completely specified Boolean function depending on n binary variables, and with associate affine space A. If f is k-autosymmetric, then the restriction f_k of f is D-reducible with respect to the same affine space A.*

Proof. First of all, we notice that the reduction f_k is the result of a projection of f onto a $(n-k)$-dimensional space, where each point of f_k in $\{0,1\}^{n-k}$ corresponds to a set of 2^k points in $\{0,1\}^n$ where f assumes the same value (as reviewed in Section 2.1).

We now show that f_k is D-reducible in $\{0,1\}^{n-k}$, where it is described by the variables y_i corresponding to the non-canonical variables of L_f, and defined by the reduction equations. Observe that the on-set minterms of f_k, and the corresponding minterms in the original space $\{0,1\}^n$, are obviously covered by A. Moreover, recall that f_k is derived by f assigning value 0 to all the canonical variables of L_f, and renaming the non-canonical variables with $y_1,\dots,y_{n-k}$. If we now assign value 0 to the occurrences of the k canonical variables of L_f in χ_A, and we rename the non-canonical variables of L_f as $y_1,\dots,y_{n-k}$, we obtain the characteristic function of an $a-k$ dimensional subspace A' of A that covers f_k in $\{0,1\}^{n-k}$. Therefore, f_k is D-reducible and can be studied in a subspace of dimension $a-k$ represented by a product of $(n-k)-(a-k)=n-a$ EXOR factors, i.e.,

$$f_k = \chi_{A'} f_{kA'} ,$$

where $f_{kA'}$ depends on $a-k$ variables.

Replacing the variables $y_1,\dots,y_{n-k}$ in both $\chi_{A'}$ and $f_{kA'}$ with the corresponding reduction equations, we derive a representation of f as

$$f = \chi_A \, f_{kA} .$$

Observe that the affine space associated to f and f_k is the same.

□

In summary, we have shown how to decompose the function f with two different strategies. If we first apply the D-reducibility decomposition, and then exploit the autosymmetry property on f_A, we obtain $f=\chi_A f_{Ak}$. If, vice-versa, we first exploit the autosymmetry of f, and then we decompose the restriction f_k using the D-reducibility property, we get $f=\chi_A f_{kA}$. Observe that both functions f_{Ak} and f_{kA} depend on the same $a-k$ variables. Finally, we have the following theorem, which immediately follows from Theorems 1 and 2, and from the fact that $f=\chi_A f_{Ak}=\chi_A f_{kA}$.

Theorem 3. *The two decompositions are equivalent, i.e., $f_{Ak}=f_{kA}$.*

The following examples show the two possible strategies implemented on the running example.

Example 5 (Autosymmetry - D-reducibility). Let us consider the running example. Now, we first apply autosymmetry and then D-reducibility to the given function f. Let us consider the function f described in Figure 1. Example 1 shows that f is 2-autosymmetric and it computes the restriction f_2 as the set of minterms $f_2(y_1,y_2,y_3)=\{001,100,110\}$ in $\{0,1\}^3$. We now compute the

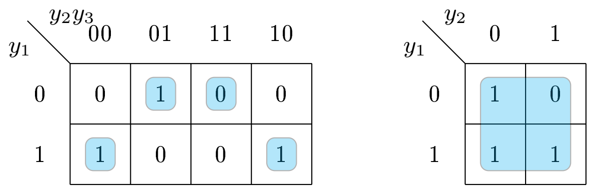

Fig. 5. Left side: Karnaugh map for $f_2(y_1, y_2, y_3)$. The space A of f is highlighted in cyan. Right side: Karnaugh map for $f_{2A}(y_1, y_2)$.

D-reducibility decomposition of f_2. The Karnaugh map for f_2 is shown on the left side of Figure 5 where the affine space A, which entirely contains f_2, is highlighted in cyan. The function f_2 can be projected in A obtaining the Boolean function $f_{2A}(y_1, y_2) = \{00, 01, 11\}$ depicted in the Karnaugh map on the right side of Figure 5. The characteristic function of A is $(y_1 \oplus y_3)$. In order to simply describe our example, we represent the function f_{2A} in SOP form (i.e., $f_{2A} = (\overline{y}_2 + y_1)$). Recall, that f_{2A} can be represented in any form, and that we will use the XAG representation in the experimental section. In summary, we have that $f_2(y_1, y_2, y_3) = \chi_A \cdot f_{2A} = (y_1 \oplus y_3)(\overline{y}_2 + y_1)$. In order to reconstruct the original function f we replace the variables y_1, y_2, and y_3 with the corresponding reduction equations computed in Example 1. We have $f(x_1, \ldots, x_5) = [(x_1 \oplus x_2 \oplus x_3) \oplus (x_1 \oplus x_5)] \cdot [\overline{x}_4 + (x_1 \oplus x_2 \oplus x_3)]$, which can be simplified. We finally obtain:
$f(x_1, \ldots, x_5) = \chi_A \cdot f_{2A} = (x_2 \oplus x_3 \oplus x_5) \cdot [\overline{x}_4 + (x_1 \oplus x_2 \oplus x_3)]$.

Example 6 (D-reducibility-Autosymmetry). Let us consider again the running example. In this case, we first apply D-reducibility and then autosymmetry to the given function f. Let us consider the function f described in Figure 3. Example 2 shows that f is D-reducible, and the projection $f_A(x_2, x_3, x_4, x_5)$ is shown in Figure 4: $f_A = \{0000, 0010, 0011, 0100, 0101, 0110, 1000, 1001, 1010, 1100, 1110, 1111\}$. We now compute the autosymmetry decomposition of f_A. The Karnaugh map for f_A is depicted on the left side of Figure 6. The projection f_A is autosymmetric, and its associated vector space is $L_{f_A} = \{0000, 0110, 1010, 1100\}$. This space has dimension $k = \log_2 |L_{f_A}| = 2$, thus f_A in Figure 6 is 2-*autosymmetric*. The canonical basis is $\{0110, 1010\}$ and the canonical variables are: x_1 and x_2. Thus, the non-canonical variables are x_3 and x_4. We can now compute the restriction f_{A2} using the subset $\{0000, 0010, 0011\}$ of the minterms of f_A that have the canonical variables set to 0. If we project such minterms into the Boolean space $\{0, 1\}^2$ of the variable x_3 and x_4, we obtain the function $f_{A2}(y_1, y_2) = \{00, 10, 11\}$ depicted in the Karnaugh map on the right-hand side of Figure 6. The corresponding reduction equations are: $y_1 = x_1 \oplus x_2 \oplus x_3; y_2 = x_4$. A SOP form for the function f_{A2} is: $SOP(f_{A2}) = \overline{y}_2 + y_1$. Applying the reduction equations, we have that $\overline{y}_2 + y_1 = \overline{x}_4 + (x_1 \oplus x_2 \oplus x_3)$. Recalling that the characteristic

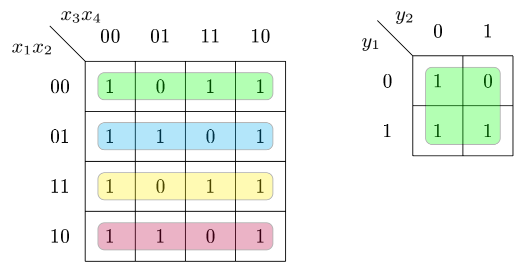

Fig. 6. Left side: Karnaugh map for the function $f_A(x_2, x_3, x_4, x_5)$. Right side: Karnaugh map for $f_{A2}(y_1, y_2)$.

function of A is $\chi_A = (x_2 \oplus x_3 \oplus x_5)$, we have:

$$f(x_1, \ldots, x_5) = \chi_A \cdot f_{A2} = (x_2 \oplus x_3 \oplus x_5) \cdot [\overline{x}_4 + (x_1 \oplus x_2 \oplus x_3)] \,.$$

We finally notice that this decomposition is identical to the one obtained with the other strategy in the previous example.

4 Incompletely Specified Autosymmetric and D-reducible Functions

In this section, we discuss the case where an incompletely specified Boolean function f is D-reducible and autosymmetric at the same time.

The autosymmetry test of an incompletely specified Boolean function specifies the don't cares to a 0 or a 1, in order to obtain a completely specified function, whose degree of autosymmetry is maximum [2]. Therefore, after the autosymmetry test, the reduced function f_k is completely specified.

Meanwhile, the D-reducibility reduction of an incompletely specified Boolean function f has the objective to find the smallest affine space A that contains the minterms of f, the points of A that are not minterms of f can be 0 or don't cares. Thus, the projected function f_A remains an incompletely specified Boolean function. In any case, if we consider a function f that is both D-reducible and autosymmetric, the resulting decomposed functions f_{kA} and f_{Ak} are completely specified, because of the autosymmetry test.

When the initial function is incompletely specified, the properties proved in Section 3 do not hold. In this case, we have that the completely specified functions f_{kA} and f_{Ak} can be different. We show this through an example from the ESPRESSO benchmark suite [17].

Example 7. Consider the function f that is the first output of the *bench* benchmark defined as follows: f^{on} = {010001, 011010, 011110, 101001, 101110}, f^{off} = {000110, 001000, 001001, 001010, 001110, 001111, 100010, 100101,

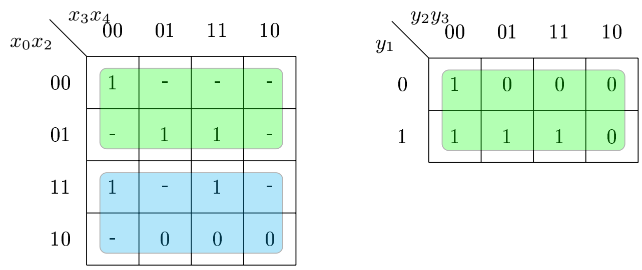

Fig. 7. Left-hand side: Karnaugh of the projection f_A for *bench_0*. Right-hand side: Karnaugh map of the restriction f_{A1} for *bench_0*

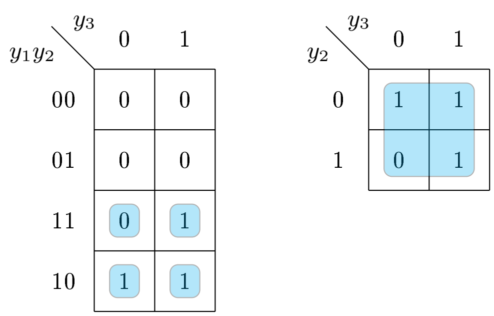

Fig. 8. Left-hand side: Karnaugh map of the restriction f_3 for *bench_0*. Right-hand side: Karnaugh map of the projection f_{3A} for *bench_0*

100110}, all the other points are in f^{dc}. If we first apply D-reducibility and then autosymmetry, we obtain the Karnaugh maps shown in Figure 7. On the left side of the figure, we have the Karnaugh map of the projection f_A, which is a 1-autosymmetric function. Thus, on the right, we have the Karnaugh map of the restriction f_{A1}. Notice that the Karnaugh map on the left contains don't cares, since the D-reducibility test does not specify the don't care conditions. If we first apply autosymmetry and then D-reducibility, we have the Karnaugh maps shown in Figure 8. The incompletely specified function f is 3-autosymmetric. Thus, on the left of Figure 8, we have the Karnaugh map of the restriction f_3. On the right side, we have Karnaugh map of the projection f_{3A}. Notice that the Karnaugh map on the left does not contain don't cares, since the autosymmetry test specifies the don't care conditions in order to obtain the best degree of autosymmetry. From this example, we can observe that in presence of don't care conditions we can have two different final results, on changing the test ordering.

Finally, considering the results obtained in Sections 3 and 4, we can define the following strategy:

- If the function is completely specified, we can use one of the two approaches (actually, the experiments in Section 6 show that performing the D-reducibility and then the autosymmetry seems to be the more efficient approach).
- If the function is incompletely specified, we should use both approaches and take the best solution (the experimental results in Section 6 show that the running time cost for performing both approaches is affordable).

5 Multiplicative Complexity

In this section we discuss the multiplicative complexity of a completely specified autosymmetric and D-reducible function.

Since f is autosymmetric and D-reducible, we can upper bound its multiplicative complexity by first projecting f onto A, and then by estimating the multiplicative complexity of the restriction f_{Ak} of f_A, as proved in [3], in the following way $M(f) \leq (n - \dim A) + M(f_{Ak})$.

Alternatively, we can first compute the restriction f_k and then estimate the multiplicative complexity of the projection $f_{k,A}$ of f_k on the affine space A. Indeed, we have that, since f is autosymmetric, the multiplicative complexity of f (i.e., $M(f)$) is equal to the multiplicative complexity of f_k (i.e., $M(f_k)$). In fact, f can be reconstruct from f_k just replacing the y_i with XORs of literals. Moreover, as proved in [3] we have that, if f is D-reducible then $M(f) \leq (n - \dim A) + M(f_A)$. Recall that, we have proved in Section 3 that, if f is both autosymmetric and D-reducible, also f_k is D-reducible. Therefore, we can say that $M(f_k) \leq (n - \dim A) + M(f_{kA})$. Since $M(f) = M(f_k)$, we finally have that $M(f) \leq (n - \dim A) + M(f_{kA})$, as expected.

6 Experimental Results

In this section, we report and discuss the experimental results reached applying both the autosymmetry test and the D-reducible decomposition to Boolean functions in the benchmarks from Espresso, LGSynth'89 benchmark suite [17] and to some functions from cryptography benchmarks in the context of multi-party computation (MPC) and fully homomorphic encryption (FHE) [14, 15].

The experiments have been run on a Intel(R) Core(TM) i7-8565U 1.80 GHz processor with 8.00 GB RAM, on Windows 11 for D-reducibility, and on a virtual machine running OS Ubuntu 64-bit for autosymmetry.

Observe that autosymmetry and D-reducibility are properties of single outputs, e.g., different outputs of the same benchmark can have different

Table 1. Results for functions that are both autosymmetric and D-reducible. Benchmarks with "*" are incompletely specified. The last row shows the average values obtained from all the benchmarks considered.

	Autosymmetry [3]			**D-reducibility [3]**			**A + D**			**D + A**		
Benchmark_output	AND	XOR	Time (s)	AND	XOR	Time (s)	AND	XOR	Time (s)	AND	XOR	Time (s)
apla_4*	18	20	13.45	15	17	2.60	18	20	13.76	**9**	5	1.01
b10_4*	15	10	11.03	**14**	6	6.64	15	6	6.51	15	6	6.58
bench_0 *	**2**	0	0.01	5	15	0.33	**2**	0	0.01	3	2	0.01
cps_74	17	4	9.11	20	2	11.52	**16**	2	6.09	**16**	2	5.72
dk17_3*	5	3	0.39	12	5	3.19	**4**	9	0.05	6	0	0.01
duke2_12*	28	5	15.88	36	25	23.07	**23**	19	11.71	**23**	19	11.47
exam_4*	22	20	13.27	59	45	42.28	22	20	13.65	**10**	13	2.95
exep_6*	20	0	8.13	16	0	0.01	**15**	0	0.16	16	0	0.01
exp_11*	16	1	9.21	6	8	0.39	**5**	8	0.46	**5**	1	0.01
p1_15*	20	26	6.60	**17**	18	13.55	18	12	5.57	18	12	4.99
p3_7*	32	30	22.55	28	11	14.04	32	30	24.99	**18**	32	6.58
pdc_3*	45	17	32.50	213	65	105.92	45	17	32.70	**36**	23	20.22
pdc_5*	21	21	14.54	270	76	123.40	**19**	25	8.47	21	25	8.55
sao2_2*	**7**	0	5.83	27	9	11.49	**7**	0	1.66	**7**	0	1.56
spla_5*	**67**	40	53.53	142	55	92.78	70	28	56.70	70	28	52.70
spla_12*	64	17	43.72	95	43	56.89	**59**	24	37.86	95	43	49.25
t1_22	**5**	0	1.23	**5**	2	0.15	**5**	0	0.10	**5**	0	0.08
t4_3*	11	6	2.00	15	6	6.91	12	1	5.43	**5**	9	0.19
x1dn_2*	**16**	8	6.58	19	8	6.35	**16**	8	5.18	17	8	2.47
dec_untilsat_39	**6**	0	1.52	**6**	0	0.01	**6**	0	1.71	**6**	0	0.01
Average	10.43	6.09	5.13	22.20	9.82	9.50	**9.78**	5.89	4.19	11.81	5.46	**3.70**

autosymmetry degrees. Therefore, we perform the autosymmetry and D-reducibility tests on the single outputs of the considered benchmark suites. We considered each output as a separate Boolean function, and analyzed a total of 237 D-reducible and autosymmetric (non degenerate) functions. The given functions and their restrictions or projections have been synthesized in XAG form using the heuristic approach proposed in [14].

We conducted four tests each composed by the following overall strategy: 1) Regularity test (autosymmetry alone; or D-reducibility alone; or first autosymmetry and then D-reducibility; or first D-reducibility and then autosymmetry); 2) XAG construction on the projected/reduced function [14]; 3) Reconstruction of the original function in XAG form (adding XORs from the reduction equations and/or adding AND of XORs for the characteristic function of the affine space A).

We report in Table 1 a significant subset of functions as representative indicators of our experiments. The first column reports the name and the number of the considered output of each benchmark. The following triples of columns report the multiplicative complexity of the XAG (AND) and the number of XORs (XOR) for the case we are considering, obtained running the heuristic in [14], and the running time in seconds. These triples describes the results for the following four different strategies: autosymmetry alone, D-reducibility alone, first autosymmetry and then D-reducibility (A + D), and first D-reducibility and second autosymmetry (D + A).

The experiments show that the functions where the XAG minimization can benefit from autosymmetry and D-reducibility are about 27%, with an average reduction of the number of ANDs of about 27.4%; the number of functions where the estimates of the multiplicative complexity are the same is about 66.7%, while for the 6.3% of the functions the method provides a worst result. The worst result could come from the fact that the approach proposed in [14], for XAG synthesis, is heuristic. Some particular benchmarks seem to highly benefit from the proposed strategies. For example, the benchmark *t4_3* can be represented using the D+A approach with the gain of 55%, in AND gates, with respect to exploiting autosymmetry alone. We finally observe that the combined methods can also provide a reduction of the number of XOR gates, due to the XOR factorization in both approaches.
In conclusion, the experiments show that:

1. Running times deeply depend on the XAG heuristic [14]. Moreover, in general, the running time for the XAG heuristic depends on the dimension of its input function. For this reason, in the cases when we perform both the testing procedures often the total running times are reduced since the input function for the XAG heuristic is smaller. In other words, the gain in running time for constructing the XAG is higher than the running times required for testing the two regularities.
2. In case of completely specified functions (where A+D and D+A give the same results), the strategy more convenient is D+A since this strategy has better running times.
3. In case of incompletely specified functions, it is convenient to test both the strategies A+D and D+A in order to find the best solution. The sum of the running times of the two approaches (A+D and D+A) is about the 50% greater than the running time of the autosymmetry approach alone (which is much more time consuming than the D-reducibility test). Therefore, testing both the strategies (A+D and D+A) is still computationally convenient.

7 Conclusion

This paper has addressed regular functions that are both autosymmetric and D-reducible. The theoretical study shows that in the case of completely specified Boolean functions, the two tests can be performed in any order, obtaining exactly the same decomposition. In the case of incompletely specified Boolean functions this property does not hold. The experimental results validate the proposed approach. Future works can include the study of other XOR-based regularities for enhancing the computation of multiplicative complexity.

References

1. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Advances in Cryptology - EUROCRYPT Proceedings, Part I. pp. 430–454 (2015)
2. Bernasconi, A., Ciriani, V.: Autosymmetry of Incompletely Specified Functions. In: Design Automation and Test in Europe (DATE) (2021)
3. Bernasconi, A., Cimato, S., Ciriani, V., Molteni, M.C.: Multiplicative Complexity of XOR Based Regular Functions. IEEE Transactions on Computers (Early Access) (2022)
4. Bernasconi, A., Ciriani, V.: Dimension-Reducible Boolean Functions Based on Affine Spaces. ACM Trans. Design Autom. Electr. Syst. **16**(2), 13:1–13:21 (2011)
5. Bernasconi, A., Ciriani, V., Luccio, F., Pagli, L.: Exploiting Regularities for Boolean Function Synthesis. Theory Comput. Syst. **39**(4), 485–501 (2006)
6. Bernasconi, A., Ciriani, V., Luccio, F., Pagli, L.: Synthesis of Autosymmetric Functions in a New Three-Level Form. Theory Comput. Syst. **42**(4), 450–464 (2008)
7. Boyar, J., Peralta, R., Pochuev, D.: On the multiplicative complexity of Boolean functions over the basis $(\wedge, +, 1)$. Theor. Comput. Sci. **235**(1), 43–57 (2000)
8. Çalik, Ç., Turan, M.S., Peralta, R.: The multiplicative complexity of 6-variable Boolean functions. Cryptography and Communications **11**(1), 93–107 (2019)
9. Ciriani, V.: Synthesis of SPP Three-Level Logic Networks using Affine Spaces. IEEE Trans. on CAD of Integrated Circuits and Systems **22**(10), 1310–1323 (2003)
10. Goudarzi, D., Rivain, M.: On the Multiplicative Complexity of Boolean Functions and Bitsliced Higher-Order Masking. In: Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, Proceedings. pp. 457–478 (2016)
11. Halecek, I., Fiser, P., Schmidt, J.: Are XORs in logic synthesis really necessary? In: 20th IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems, DDECS 2017, Dresden, Germany, April 19-21, 2017. pp. 134–139 (2017)
12. Liebler, R.: Basic Matrix Algebra with Algorithms and Applications. Chapman & Hall/CRC. (2003)
13. Luccio, F., Pagli, L.: On a New Boolean Function with Applications. IEEE Transactions on Computers **48**(3), 296–310 (1999)
14. Testa, E., Soeken, M., Riener, H., Amaru, L., Micheli, G.D.: A Logic Synthesis Toolbox for Reducing the Multiplicative Complexity in Logic Networks. In: 2020 Design, Automation Test in Europe Conference Exhibition (DATE). pp. 568–573 (2020)
15. Testa, E., Soeken, M., Amarù, L.G., Micheli, G.D.: Reducing the Multiplicative Complexity in Logic Networks for Cryptography and Security Applications. In: Proceedings of the 56th Annual Design Automation Conference 2019, DAC. p. 74 (2019)
16. Turan, M.S., Peralta, R.: The Multiplicative Complexity of Boolean Functions on Four and Five Variables. In: Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey. pp. 21–33 (2014)
17. Yang, S.: Logic synthesis and optimization benchmarks user guide version 3.0. User guide, Microelectronic Center (1991)